Random Oracles in the Standard Model

A Systematic Study of Random Oracle (Un-)Instantiability via Universal Computational Extractors and Obfuscation

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doctor rerum naturalium (Dr. rer. nat.)
von

Arno Andreas Mittelbach, M.Sc.
geboren in Mainz

Referenten:	Prof. Dr. Marc Fischlin
	Prof. Dr. Ran Canetti
Tag der Einreichung:	20.10.2015
Tag der mündlichen Prüfung:	07.12.2015

Darmstadt, 2016
D 17

Erklärung

Hiermit versichere ich, die vorliegende Arbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 20. Oktober 2015

Arno Mittelbach

Wissenschaftlicher Werdegang

Februar 2012 – Oktober 2015

Doktorand der Informatik an der Technischen Universität Darmstadt.

April 2009 – August 2011

Studium der Informatik an der Technischen Universität Darmstadt.
Master of Science, Note 1.1.

August 2008 – März 2009

Research Programmer an der University of Oxford,
Abteilung: Oxford University Computer Services.

Oktober 2005 – August 2008

Studium der Informatik an der Technischen Universität Darmstadt.
Bachelor of Science, Note 1.1 “mit Auszeichnung”.

List of Publications

Peer Reviewed Conference Publications

[1] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random oracle uninstantiability from indistinguishability obfuscation. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. **Part of this thesis.**

[2] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via UCEs. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 122–141, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. **Part of this thesis.**

[3] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 142–161, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. **Part of this thesis.**

[4] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 188–205, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Berlin, Germany. **Part of this thesis.**

[5] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 603–621, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany. **Part of this thesis.**

[6] Jean Paul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, Giorgia Azzurra Marson, Arno Mittelbach, and Kenneth G. Paterson. Unpicking plaid - a cryptographic analysis of an iso-standards-track authentication protocol. In Liqun Chen and Chris Mitchell, editors, *Security Standardization Research – First International Conference, SSR 2014*, volume 8893 of *Lecture Notes in Computer Science*, pages 1–25, London, UK, December 1014. Springer, Berlin, Germany.

[7] Paul Baecher, Christina Brzuska, and Arno Mittelbach. Reset indifferentiability and its consequences. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013*, volume 8269 of *Lecture Notes in Computer Science*, pages 154–173, Bengalore, India, December 1–5, 2013. Springer, Berlin, Germany.

[8] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of OPACITY - (extended abstract). In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *European Symposium on Research in Computer Security, ESORICS 2013: 18th*, volume 8134 of *Lecture Notes in Computer Science*, pages 345–362, Egham, UK, September 9–13, 2013. Springer, Berlin, Germany.

[9] Arno Mittelbach. Cryptophia's short combiner for collision-resistant hash functions. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *International Conference on Applied Cryptography and Network Security, ACNS 2013: 11th*, volume 7954 of *Lecture Notes in Computer Science*, pages 136–153, Banff, AB, Canada, June 25–28, 2013. Springer, Berlin, Germany.

[10] Arno Mittelbach. Hash combiners for second pre-image resistance, target collision resistance and pre-image resistance have long output. In Ivan Visconti and Roberto De Prisco, editors, *International Conference on Security in Communication Networks, SCN 2012: 8th*, volume 7485 of *Lecture Notes in Computer Science*, pages 522–539, Amalfi, Italy, September 5–7, 2012. Springer, Berlin, Germany.

[11] Arno Mittelbach, Lasse Lehmann, Christoph Rensing, and Ralf Steinmetz. Automatic detection of local reuse. In Martin Wolpers, PaulA. Kirschner, Maren Scheffel, Stefanie Lindstaedt, and Vania Dimitrova, editors, *Sustaining TEL: From Innovation to Learning and Practice*, volume 6383 of *Lecture Notes in Computer Science*, pages 229–244. Springer Berlin Heidelberg, 2010.

[12] Lasse Lehmann, Arno Mittelbach, James Cummings, Christoph Rensing, and Ralf Steinmetz. Automatic detection and visualisation of overlap for tracking of information flow. In *Proceedings I-Know*, 2010.

Peer Reviewed Journal Publications

[1] Jean Paul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, Giorgia Azzurra Marson, Arno Mittelbach, and Kenneth G. Paterson. Unpicking PLAID: a cryptographic analysis of an ISO-standards-track authentication protocol. *International Journal of Information Security*, pages 1–21, 2016.

[2] James Cummings and Arno Mittelbach. The holinshed project: Comparing and linking two editions of holinshed's chronicle. *International Journal of Humanities & Arts Computing (IJHAC)*, 4:39–53, 2010.

[3] Lasse Lehmann, Arno Mittelbach, Christoph Rensing, and Ralf Steinmetz. Capture of lifecycle information in office applications. *International Journal of Technology Enhanced Learning (IJTEL)*, 2(1):41–57, 2010.

Non-Refereed Articles (Articles in Submission)

[1] Christina Brzuska and Arno Mittelbach. Universal computational extractors and the superfluous padding assumption for indistinguishability obfuscation. Cryptology ePrint Archive, Report 2015/581, 2015. `http://eprint.iacr.org/2015/581`. **Part of this thesis.**

Peer Reviewed Posters

[1] James Cummings and Arno Mittelbach. The holinshed project: comparing and linking two editions of holinshed's chronicles. Poster at Digital Resources for the Humanities & Arts (DRHA 2009), Sep 2009.

Acknowledgments

2015 was an eventful year for me and one of its highlights was finishing my Ph.D (an event only topped by me marrying my longtime girlfriend Mareika). I've had a wonderful time working as a Ph.D. student in Darmstadt where I had the great fortune to be part of a fantastic research group headed by Marc Fischlin. I am deeply thankful to Marc for giving me the opportunity to pursue a Ph.D. in his group and for introducing me to the world of theoretic cryptography.

I would like to thank my co-workers and co-authors, many of which became good friends. In particular I would like to thank Paul Baecher, Christina Brzuska, Özgür Dagdelen, Jean Paul Degabriele, Pooya Farshim, Victoria Fehr, Marc Fischlin, Felix Günther, Tommaso Gagliardoni, Peter Gaži, Giorgia Marson, Sogol Mazaheri, Cristina Onete, Kenny Paterson and Daniele Venturi. I need to single out a few in this list. Thanks a lot to Cristina, Giorgia and Victoria with whom I shared the *butterfly office* over the years. Thanks to Paul and Felix for letting me procrastinate time and again in the *coffee office*. A huge thank you to Christina who introduced me to the world of research in coffee places and who taught me that the intuitively plausible is often impossible and that while intuition is a good start it always needs a proper formalization.

I would also like to thank Mihir Bellare and Ran Canetti. My work is based on the works of hundreds of fantastic researchers but Mihir's and Ran's works clearly stand out in almost every chapter of my thesis. Thank you for so much inspiration. I am particularly grateful to Ran for joining my graduation committee and agreeing to co-review my thesis.

Research is a funny type of work and inspirations come at strange moments. In that respect, I need to greatly thank Leni, my dog, who joined our family five years ago and who never got tired of telling me that extensive walks are the best strategy to organize one's thoughts and who is responsible for many inspirations.

Finally, I am deeply thankful to my wife Mareika for supporting me in everything I set out to do. A Ph.D. isn't easy for the person pursuing it and it is definitely not easy for that person's partner. Thank you for always being there for me.

Arno Mittelbach
Roßdorf, January 2016

Abstract

Provable security is a fundamental concept of modern cryptography (see, e.g., Katz and Lindell; Introduction to Modern Cryptography, Chapter 1, 2007). In order to argue about security, we first require a precise and rigorous definition of what *security* means (e.g., a definition of *secure encryption*). Such a security definition, in particular, contains a description of the capabilities of an adversary, i.e., a model of the adversary which tries to come as close to reality as possible. Given a security model, the provable security approach is to provide a mathematical proof that a given construction achieves the desired level of security. In other words, we prove that no (efficient) adversary can break the security with "good probability" given that it behaves as defined within the security model. Typically, proofs reduce the security of a construction to an unproven cryptographic hardness-assumption—we show that the existence of an adversary violates an assumption—which, preferably, is as simple as possible. Furthermore, any assumption needs to be stated precisely. Examples of cryptographic hardness assumptions can be complexity-theoretic assumptions, such as, $\mathsf{P} \neq \mathsf{NP}$, the assumed existence of objects, such as, one-way functions, or long standing open problems from number theory, such as, factoring large integers or computing the discrete log in certain groups.

A widely used technique for the construction of cryptographic schemes is the so-called *random oracle methodology*, introduced in 1993 by Bellare and Rogaway (CCS, 1993). As before, we start with a precise model of security which is extended to include a uniformly random function which may be evaluated by any party (including the construction and adversary). As a random function has a huge, if not infinite description, parties cannot be given its code but, instead, are provided with black-box access to an oracle which evaluates the function for them. This oracle is called the *random oracle*. Then, as before, a mathematical proof is given that a construction is secure as per definition relative to the random oracle. Finally, to implement the scheme in practice, the random oracle is replaced by a cryptographic hash function (such as SHA-3).

No mathematical model can fully capture reality and, thus, cast in the framework of provable security, we may consider a *random oracle security model* as being somewhat further away from reality than a *standard security model*: in addition to the abstractions of the standard security model it is assumed that adversaries do not make any use of the code of a hash function; it is assumed that they do not even evaluate the hash function on their own but use an external device for the evaluation (i.e., black-box access). Now, if we trust schemes that are designed according to the provable security approach, should we then also trust schemes devised via the random oracle methodology?

This question has lead to a huge debate within the cryptographic community and has been discussed for more than two decades. The discussion is fueled by results showing that the extension of security models to include a random oracle may produce provably secure schemes that cannot be securely implemented. In 1998, Canetti, Goldreich, and Halevi (STOC, 1998) showed that schemes exist that are inherently insecure but which should be secure according to the random oracle

methodology. In more detail, they present a public-key encryption scheme which an adversary can trivially attack when given the code of the hash function that was used to replace the random oracle. Note that this differs from, e.g., side-channel attacks: while here also the attack is successful because it is *outside the model*, implementations can, potentially, protect against these attacks and, thus, secure implementations may exist. With the scheme presented by Canetti et al., on the other hand, there is, provably, no secure implementation although the scheme is secure in the random oracle model.

Despite these negative results, many of the schemes that we trust on a daily basis—examples include the standardized public-key encryption scheme RSA-OAEP as well as the standardized signature schemes RSA-PSS and DSA[1]—only have proofs in the random oracle model. Similarly, for many "advanced" cryptographic concepts, including IND-secure deterministic public-key encryption, correlated-input secure hash functions, universal hardcore functions, and many others, we (so far) only have constructions in the random oracle model. One reason for the success of random oracles is that they allow to design very efficient and "natural" schemes. Furthermore, the power of random oracles enables us to realize concepts which we would not know how to implement without random oracles. A third, and very compelling argument in favor of the random oracle methodology is that the "random oracle heuristic" seems to be a good one: no random oracle scheme which was not designed to portrait inconsistencies of the random oracle model has been attacked due to the use of random oracles. However, if a scheme is, indeed, secure, should we then not be able to understand and pinpoint the underlying source of hardness?

In this thesis we study random oracles with the help of *program obfuscation* (in particular *indistinguishability obfuscation* and *point-function obfuscation*). The study of obfuscation has a long tradition in computer science, and specifically in cryptography, but only recent advancements gave rise to the first candidate constructions of provably secure general-purpose indistinguishability obfuscators (Garg et al.; FOCS, 2013). Intuitively, a program obfuscator takes as input a program and produces a functionally equivalent but unintelligible program, i.e., the obfuscated program "hides how it operates". Conceptually, this is very close to one of the fundamental abstractions made within the random oracle model where hash functions do not have an explicit and efficient description but can be evaluated only via black-box access to the random oracle. While an obfuscated hash function still has an explicit and efficient description, the description should hide the way the function works and, thus, intuitively, should be of no help to any adversary.

Using obfuscation-based techniques we show how to instantiate the random oracle in various cryptographic constructions. Amongst others, we obtain the first *standard model* (i.e., without random oracles) candidate construction for a universal hardcore function with long output, a q-query correlated-input secure hash function, and q-query IND-secure deterministic public-key encryption. We obtain our positive results by instantiating various forms of *universal computational extractors*. The universal computational extractor (UCE) framework was introduced by Bellare, Hoang, and Keelveedhi (CRYPTO, 2013) to provide (very strong) standard-model notions of hash functions that allow instantiating random oracles in a wide range of applications. Intriguingly, even though obfuscation allows us to show how to replace random oracles in certain situations, it also allows us to show limitations of the random oracle methodology as well as of UCEs. Using obfuscation-based techniques, we prove that several concrete UCE assumptions (including all originally proposed as-

[1]We note that for the digital signature algorithm (DSA) no security proofs are known. DSA is, however, closely related to ElGamal which has a proof of security in the random oracle model but none without random oracles.

sumptions) cannot hold in case indistinguishability obfuscation exists. (We note that these negative results inspired the weaker UCE notions that lay at the core of our positive constructions.) Assuming the existence of indistinguishability obfuscation also allows us to extend the uninstantiability techniques of Canetti et al. (STOC, 1998) and show that a large class of random-oracle transformations are not sound. This affects the Encrypt-with-Hash transformation (Bellare et al.; CRYPTO, 2007) to construct deterministic public-key encryption, as well as the widely used Fujisaki–Okamoto transformation (Fujisaki, Okamoto; CRYPTO, 1999) which transforms weak public-key encryption schemes into strong public-key encryption schemes.

An often repeated criticism of random-oracle uninstantiability results is that the schemes only fail to be secure because they are designed to do so and, furthermore, their "artificial" design conflicts "good cryptographic practice" (see, for example, Koblitz and Menezes; Journal of Cryptology, 2007). Similar criticism can be voiced also for our counterexamples to the general applicability of the above mentioned random-oracle transformations. While this does not refute the mathematical validity of such uninstantiability results, we do, however, also present a very different counterexample to the soundness of the random oracle methodology: we show that if indistinguishability obfuscation exists, then a strong variant of point-function obfuscation (which can be similarly interpreted as a strong form of symmetric encryption) cannot be achieved without the help of random oracles while at the same time there are simple and elegant constructions in the random oracle model. We note that the same holds also for our negative results for UCEs.

In summary, we develop techniques to work with obfuscation which allow us to show that the existence of indistinguishability obfuscation implies that various random oracle techniques may lead to insecure schemes. Our results suggest, once again, that we should be careful with random oracle proofs and we hope that they spark further research to overcome the necessity to use random oracles in the first place. Concerning the latter, we make first steps by proposing new and widely applicable UCE notions together with standard-model candidate constructions showing that UCEs may, indeed, be a viable alternative to the use of random oracles.

Zusammenfassung

Ein grundlegendes Prinzip moderner Kryptographie ist die sogenannte *Beweisbare Sicherheit* (siehe z.B. Katz und Lindell; Introduction to Modern Cryptography, Kapitel 1, 2007). Um die Sicherheit eines Verfahrens mathematisch zu zeigen wird zunächst eine präzise mathematische Definition davon benötigt, was *Sicherheit* bedeutet (z.B. eine Definition von *sicherer Verschlüsselung*). Eine solche Definition beinhaltet insbesondere eine Beschreibung der Fähigkeiten eines Angreifers, das heißt, ein Angreifermodell, welches so genau wie möglich die realen Gegebenheiten abbilden sollte. Gegeben eine Sicherheitsdefinition wird anschließend bewiesen, dass ein bestimmtes Verfahren die Definition erfüllt. Hierfür zeigen wir typischerweise, dass die Existenz eines (effizienten) Angreifers eine oder mehrere (meist unbewiesene) kryptographische Annahmen verletzt. Diese können beispielsweise aus der Komplexitätstheorie (z.B. $\mathsf{P} \neq \mathsf{NP}$) oder Zahlentheorie (z.B. Faktorisieren großer Zahlen oder die Berechnung des diskreten Logarithmus in bestimmten Gruppen) stammen oder es kann die Existenz kryptographischer Primitive gefordert werden (z.B. die Existenz von One-way Funktionen). Verwendete Annahmen müssen ebenfalls präzise definiert und sollten möglichst einfach und allgemein gehalten werden.

Eine weitverbreitete Methode für die Konstruktion von kryptographischen Primitiven ist die 1993 von Bellare und Rogaway eingeführte *Random-Oracle-Methodik* (CCS, 1993). Wie zuvor ist der Ausgangspunkt auch hier eine präzise Sicherheitsdefinition. Diese wird zudem um die Existenz einer zufälligen Funktion erweitert, auf die jede Partei (insbesondere auch Konstruktion und Angreifer) Zugriff haben. Da eine zufällige Funktion eine sehr große, wenn nicht sogar unendlich große Beschreibung hat, kann diese den Parteien nicht direkt gegeben werden. Um die Funktion dennoch ausführen zu können erhalten sie daher Black-Box-Zugriff auf ein Orakel, das die Funktion für sie ausführt. Dieses Orakel wird auch *Random Oracle* genannt. Anschließend wird wie zuvor ein mathematischer Beweis geführt, um die Sicherheit der Konstruktion relativ zu der Existenz eines Random Oracles zu zeigen. Um eine solche Konstruktion in der Praxis verwenden zu können wird das Random Oracle für die Implementierung durch eine kryptographische Hashfunktion (z.B. SHA-3) instanziiert.

Ein mathematisches Modell kann die Realität niemals in Gänze abbilden. Ein um ein Random Oracle erweitertes Sicherheitsmodell kann daher als etwas weiter von der Realität entfernt angesehen werden: Neben den Abstraktionen eines Standard-Sicherheitsmodells wird zusätzlich angenommen, dass Angreifer die Beschreibung der Hashfunktion ignorieren und diese nur über eine externe Instanz (per Black-Box-Zugriff) auswerten. Wenn wir nun kryptographischen Verfahren vertrauen, die *beweisbar sicher* sind, sollten wir dann auch Verfahren vertrauen, die relativ zu einem Random Oracle beweisbar sicher sind?

Über die Einordnung von Random-Oracle-Verfahren wird seit über zwei Jahrzehnten in der kryptographischen Gemeinschaft gestritten. So zeigen diverse Resultate, dass die Erweiterung von Sicherheitsmodellen um Random Oracles problematische Konstruktionen zur Folge haben können,

Konstruktionen die zwar im Random-Oracle-Modell sicher, jedoch in der Praxis immer unsicher sind, unabhängig davon wie sie implementiert werden. Canetti, Goldreich und Halevi (STOC, 1998) zeigen die Existenz von im Random-Oracle-Modell sicheren Public-key Verschlüsselungsverfahren, die jedoch trivial angreifbar sind sobald ein Angreifer Zugriff auf die Beschreibung der verwendeten Hashfunktion erhält. Ein solches Resultat ist insbesondere von z.B. Seitenkanalangriffen zu unterscheiden: Obwohl in beiden Fällen Angriffe erfolgreich sind, die durch das Sicherheitsmodell nicht abgedeckt werden, können Implementierungen potentiell gegen Seitenkanalangriffe gesichert werden; sichere Implementierungen können existieren. Im Falle des Public-key Verfahrens von Canetti et al. kann hingegen keine sichere Implementierung existieren obwohl das Verfahren im Random Oracle Modell als sicher bewiesen ist.

Trotz negativer Resultate sind viele Verfahren die wir tagtäglich einsetzen nur im Random Oracle Modell als sicher bewiesen. Beispiel hierfür sind das standardisierte Verschlüsselungsverfahren RSA-OAEP sowie die standardisierten Signaturverfahren RSA-PSS und DSA[2] genannt. Darüber hinaus kennen wir für viele komplexeren kryptographischen Primitive lediglich Konstruktionen im Random Oracle Modell. Beispiele hierfür sind IND-sichere deterministische Public-key Verschlüsselung, Correlated-input sichere Hashfunktionen oder universelle Hardcorefunktionen.

Gründe für den Erfolg von Random Oracles sind vielfältig. Zum einen sind Random-Oracle-Verfahren häufig sehr effizient und "natürlich". Darüber hinaus ermöglichen Random Oracles die Konstruktion starker kryptographischer Primitive, die wir ohne den Einsatz von Random Oracles derzeit nicht konstruieren können. Ein weiteres Argument für den Einsatz von Random Oracles ist das die "Random-Oracle-Heuristik" zu funktionieren scheint: Kein Random Oracle Verfahren, das nicht mit dem Ziel konstruiert worden ist Schwächen des Random Oracle Modells aufzuzeigen konnte bislang aufgrund der Nutzung von Random Oracles angegriffen werden. Aber sollten wir, wenn ein Verfahren in der Tat sicher ist, nicht in der Lage sein den Grund der Sicherheit, bzw. das zugrundeliegende schwere Problem zu identifizieren?

In dieser Dissertation untersuchen wir Random Oracles mit Hilfe von *Code Obfuscation* (insbesondere betrachten wir *Indistinguishability Obfuscation* sowie *Point-function Obfuscation*). Code Obfuscation hat eine lange Tradition in der Informatik (insbesondere der Software Entwicklung) wurde jedoch meist als Heuristik betrachtet. Ein wissenschaftlicher Durchbruch in kryptographischer Obfuscation (einer beweisbar sicheren Form von Obfuscation) gelang erst kürzlich Garg et al. (FOCS, 2013), die eine Konstruktion eines Indistinguishability Obfuscators vorschlagen. Ein Code Obfuscator ist vereinfacht gesagt ein Programm, welches als Eingabe den Code eines Programms erwartet und einen funktional äquivalenten, jedoch nicht mehr verständlichen, Programmcode erzeugt: Aus dem obfuszierten Programmcode kann die Art und Weise, wie das Programm Berechnungen durchführt nicht mehr rekonstruiert werden. Konzeptionell ist dies ähnlich zu den Abstraktionen des Random-Oracle-Modells bei denen davon ausgegangen wird, dass kein Programmcode der Hashfunktion existiert und diese nur mittels Black-Box-Zugriff auf das Random Oracle ausgewertet werden kann. Bei einer obfuszierten Hashfunktion liegt zwar Code vor, jedoch versteckt die Beschreibung jegliche Informationen über die Funktion und sollte daher einem Angreifer, so die Idee, nicht nützlich sein.

In der vorliegenden Arbeit zeigen wir, wie mittels Obfuscation, Random Oracles in verschiedenen Konstruktionen ersetzt werden können. Unter anderem geben wir die ersten Standardmodell-

[2]Der Digital Signature Algorithm (DSA) ist verwandt mit dem ElGamal Verfahren, welches lediglich einen Sicherheitsbeweis im Random Oracle Modell hat. Für DSA ist hingegen selbst im Random Oracle Modell kein Sicherheitsbeweis bekannt.

Konstruktionen (d.h. Konstruktionen ohne die Verwendung von Random Oracles) für universelle Hardcorefunktionen mit langer Ausgabe, für q-query Correlated-input sichere Hashfunktionen sowie für q-query IND-sichere deterministische Verschlüsselung an. Hierfür zeigen wir, wie verschiedene sogenannte *Universal Computational Extractors* mittels Obfuscation instanziiert werden können. Das Universal Computational Extractor (UCE) Framework wurde von Bellare, Hoang und Keelveedhi (CRYPTO, 2013) vorgeschlagen, um ausreichend starke Eigenschaften von Hashfunktionen im Standardmodell abbilden zu können, die es erlauben Random Oracles in unterschiedlichen Anwendungen zu instanziieren. Obfuscation erlaubt hingegen nicht nur Random Oracles in verschiedenen Kontexten zu ersetzen, sondern kann auch zur Auslotung der Grenzen von Random Oracles und UCEs verwendet werden. So zeigen wir, dass verschiedene UCE-Annahmen (dies beinhaltet insbesondere alle Originalannahmen von Bellare et al.) nicht haltbar sind, sollte Indistinguishability Obfuscation existieren. (Es sei angemerkt, dass diese negativen Resultate die schwächeren UCE Annahmen inspiriert haben, welche die Basis unserer positiven Konstruktionen bilden.) Unter der Annahme, dass Indistinguishability Obfuscation existiert, können wir darüber hinaus die Uninstanziierbarkeitsresultate von Canetti et al. (STOC, 1998) erweitern: Wir zeigen, dass diverse Random-Oracle-Transformationen uninstanziierbare Konstruktionen hervorbringen können. Dies betrifft unter anderem die Encrypt-with-Hash Transformation (Bellare et al.; CRYPTO, 2007) für die Erzeugung deterministischer Public-key Verschlüsselungssysteme sowie die weitverbreitete Fujisaki–Okamoto Transformation (Fujisaki, Okamoto; CRYPTO, 1999), die schwache Public-key Verschlüsselungsverfahren in sehr starke Verfahren transformiert.

Eine häufig vorgebrachte Kritik an Uninstanziierbarkeitsresultaten für Random Oracles ist, dass die aufgezeigten Verfahren nur deswegen unsicher werden, da diese explizit so konstruiert worden sind und dass dies nur deswegen möglich ist weil ihr "künstlicher Aufbau" "gute kryptographischer Praxis" ignoriert (siehe z.B. Koblitz und Menezes; Journal of Cryptology, 2007). Diese Kritik kann ebenso gegen unsere Gegenbeispiele für die allgemeine Anwendbarkeit der oben genannten Random-Oracle-Transformationen vorgebracht werden. Obwohl dies die mathematische Gültigkeit der Ergebnisse nicht mindert, zeigen wir dennoch weitere Gegenbeispiele für die diese Kritik nicht gilt. Unter der Annahme, dass Indistinguishability Obfuscation existiert zeigen wir, dass starke Varianten von Point-function Obfuscation (diese können auch als starke symmetrische Verschlüsselungsverfahren angesehen werden) nicht existieren können, obwohl im Random-Oracle-Modell effiziente und einfache Konstruktionen existieren.

Zusammenfassung: Wir entwickeln Techniken zur Arbeit mit Obfsucation, die uns erlauben zu zeigen, dass diverse Random-Oracle-Techniken zu unsicheren Verfahren führen können unter der Annahme, dass Indistinguishability Obfuscation existiert. Unsere Ergebnisse legen nahe, Random-Oracle-Beweisen mit einer gewissen Skepsis zu begegnen und dass die Forschung an Techniken zur Überwindung von Random Oracles ein lohnenswertes Unterfangen ist. In diesem Sinne zeigen wir neue und schwächere UCE Definitionen auf (mit zugehörigen Standardmodell-Konstruktionen), die es erlauben Random Oracles in vielfältigen Kontexten zu umgehen.

Short Contents

Contents

Chapter 1

Introduction

> The status of the random oracle model, thus, is as follows: it allows us to "prove" a whole lot of pracitcal [sic] signature schemes secure [...], as well as a lot of encryption schemes and other things, but the meaning of these proofs is uncertain (as opposed to proofs in the model without random oracles, which clearly imply that the scheme cannot be broken without violating the security assumption). It continues to be used because of its power, but it would be very nice if someone figured out how to prove these things without random oracles to give us more assurance that these are really secure. I personally view it as a way to acknowledge our failures: there are a lot of constructions that seem secure on some intuitive level, but we can't prove them secure in the standard model. So (hopefully until we have a realy [sic] proof of security) we prove the [sic] secure in this funny fake model.
>
> Leo Reyzin [Rey03]

In January 2007 the National Institute of Standards and Technology (NIST) announced the development of new hash algorithms for a revision of the Federal Information Processing Standard (FIPS) 180–2, the Secure Hash Standard [NIST07]. NIST decided that the new hash functions were to be selected via a public competition (the SHA-3 competition) similar to a competition that was started roughly ten years earlier and which led to the development of the Rijndael block cipher which is nowadays known as AES. With their first announcement NIST formulated selection criteria for candidate algorithms based on different factors ranging from *security* to *licensing* [NIST07]. Of particular interest to us is one of the security criteria: NIST suggested to judge algorithms based on

> "*[t]he extent to which the algorithm output is indistinguishable from a random oracle.*"
>
> [NIST07]

Random oracles have a long tradition in the study of complexity theory and were made popular in the context of cryptography with the highly celebrated work of Bellare and Rogaway who, in 1993, introduced the *random oracle methodology* [BR93]. In simple terms, a random oracle provides black-box access to a random function that maps bit-strings to bit-strings. Usually, we here consider functions from the space $\{0,1\}^* \to \{0,1\}^\ell$, that is, functions that take arbitrarily long bit-strings as input and output a fixed-length bit-string. Having black-box access to a function can be thought of as having access to a device that provides a query interface which allows to observe input-output behavior but anything beyond remains hidden. The *random oracle methodology* as introduced by Bellare and Rogaway [BR93] is a design paradigm for devising cryptographic constructions:

> "*We argue that the random oracle model [...] provides a bridge between cryptographic theory and cryptographic practice. In the paradigm we suggest, a practical protocol P is produced by first devising and proving correct a protocol P^R for the random oracle model, and then replacing oracle accesses by the computation of an 'appropriately chosen' function h. This paradigm yields protocols much more efficient than standard ones while retaining many of the advantages of provable security.*" Bellare and Rogaway [BR93]

The random oracle methodology found countless applications both in cryptographic theory and practice. Random oracles are used regularly in security analyses (examples are [MSW08, GMP+08, Wil11, DFG+13, Dag13]) and inspired various practical cryptographic constructions that we trust on a daily basis and that have become standardized cryptographic schemes (e.g., [BR95, BR96, RFC 3447, FIPS 186-4]). The huge success of the random oracle methodology is further exemplified by the fact that "Random Oracles are Practical" [BR93], the paper in which Bellare and Rogaway introduced the random oracle methodology is one of the most cited papers in the field of modern cryptography; according to Google Scholar it has been cited almost 4000 times as of August 2015.

When following the random oracle methodology we design a cryptographic scheme P^{RO} relative to a random oracle RO which means that when proving security for P we consider a world in which every party has black-box access to the random oracle RO. Every party includes, in particular, the scheme P itself as well as any adversary. Then, to implement the construction (for example, in hard- or software) the random oracle is replaced by an *appropriately chosen* efficiently computable function h which we call a standard-model function where the term *standard model* refers to a world without random oracles and emphasizes that h is an effectively and efficiently computable function. In practice, we usually choose h as a cryptographic hash function and it, thus, seems to be a reasonable requirement for a new hash function, that is to become SHA-3 to produce *output that is essentially indistinguishable from a random oracle.*

The Random Oracle Model – A "Funny Fake Model"

There is one caveat when using the random oracle model (ROM). In 1998, Canetti, Goldreich, and Halevi ([CGH98]; CGH) showed that the random oracle methodology is *unsound*: it allows to *prove the security* of schemes that are *inherently insecure*. To this end, CGH construct specially designed encryption and signature schemes and prove that they are secure in the ROM. They then show that there can be no *efficient* function h that can replace the random oracle in implementations of their schemes, that is, whatever the choice of h, the resulting scheme will be insecure.

From a theoretical standpoint this result has severe consequences for the random oracle methodology. Indeed, one can go so far as to argue that proofs in the random oracle model are useless as they do not provide any formal implications for the security of the scheme in practice. In other words, a scheme that has a security proof in the random oracle model may, or may not be secure in the real world. Goldreich puts it this way:

> *"The bottom-line: It should be clear that the Random Oracle Methodology is not sound; that is, the mere fact that a scheme is secure in the Random Oracle Model cannot be taken as evidence (or indication) to the security of (possible) implementations of this scheme."* Oded Goldreich, [CGH98]

Consequently, neither random oracles nor appropriate approximations do exist in the standard model and one might expect this to be the end of the story, but far from it.

Despite the negative result of CGH, the random oracle methodology is still widely used both in theory and practice and, indeed, we have several good reasons for doing so. Schemes designed based on random oracles are often much more efficient and elegant than their counter-parts that try to avoid using random oracles. Furthermore, for a large number of interesting primitives we do not have any alternatives: all known constructions are in the random oracle model. In practice we often need to make trade-offs between practicability (especially efficiency) and (provable) security and, thus, using a scheme that has a security proof in the random oracle model is of course better than using one without any proof whatsoever. Oded Goldreich puts it this way: "[The random oracle model] may be useful as a test-bed (or as a sanity check). Indeed, if the scheme does not perform well on the test-bed (resp., fails the sanity check) then it should be dumped." [CGH98]. The best argument in favor of the random oracle methodology is, however, very simple: it does (seem to) work in most interesting cases. Constructions that we use in practice and that rely on the random oracle model have not been found to contain weaknesses caused by the use of the random oracle methodology. On the other hand, most of the counter-examples to the general applicability of the random oracle model make use of *contrived* constructions that violate *good cryptographic practice* in order to highlight inconsistencies of the ROM [KM15]. The counter-example presented by CGH [CGH98] is a prototypical example. On a high level, they construct a signature scheme that contains a backdoor: their signature scheme returns the signing key if the to-be-signed message is equivalent to the program code of the hash function that was used as replacement to the random oracle. Indeed, most counter-examples exhibit a more or less severe violation of good cryptographic practice which is the main argument put forward by Koblitz and Menezes [KM07, KM15] who strongly encourage the use of the methodology. They write:

> *"Like [5], [4], and [18], the work by Goldwasser and Tauman seems to be another case where leading experts dedicate considerable energy in an attempt to refute the validity of the random oracle model, but can only come up with a contrived construction that has no plausible connection with actual cryptographic practice. Our confidence in the random oracle assumption is unshaken."*[1] [KM07]

While random oracle schemes work well in practice the absence of standard model proofs should make us suspicious. In particular, we will see that not all counter arguments to the random oracle model are contrived. An intriguing example of what can go wrong when instantiating random oracles in "natural random oracle constructions" was given by Halevi and Krawczyk [HK07]. They note that the random oracle encryption scheme of Black et al. [BRS03] which is provably secure against key-dependent message attacks in the random oracle model may not have this property for natural instantiations of the random oracle. For example, if the random oracle is instantiated via the Davies–Meyer construction [Win83] the resulting instantiation will not be secure against key-dependent message attacks even if the underlying block-cipher is assumed to be ideal.

If we can only prove a scheme secure in the random oracle model we should ask ourselves whether we are missing the right proof techniques or whether the absence of a standard model proof is simply

[1] Here, [5,4] refers to [BBP04] and its full version [BBP03], [18] refers to the work by Canetti et al. [CGH98] and the mentioned work by Goldwasser and Tauman refers to [GK03].

a result of the obvious: we cannot prove the scheme secure because it is not. In one of his lecture notes [Rey03] Leonid Reyzin gives an accurate description of this dilemma (also see the chapter note): He calls the random oracle model a *funny fake model* attributing both the fact that it works well in practice but that ideally we should be able to prove the security of schemes without the help of random oracles, if they are, indeed, secure.

What is necessary is a better understanding of random oracle proofs and the properties of random oracles that are used by our constructions. Once isolated we can then try to obtain such properties in the standard model. Naturally, we will not succeed with all properties as we know, due to the result of CGH, that properties exist that cannot be replicated in the standard model. But it is exactly this discrepancy that we are interested in since any property that we rely upon in the random oracle model, but which we cannot replicate in the standard model, is potentially dangerous. On the other hand, properties that we can replicate in the standard model may allow us to ultimately move beyond random oracles and provide standard model proofs of security for schemes that today we can only assume to be secure.

Universal Computational Extractors

Several standard-model properties of random oracles have been suggested over the years. Canetti, for instance, introduced *oracle hashing schemes* [Can97], which were later renamed *perfectly (probabilistic) one-way hash functions* [CMR98] and which capture that random oracle values "hide all partial information on their input". Boldyreva et al. [BCFW09] identify a rather different property and note that random oracles are *non-malleable* meaning that given a random oracle value $\mathsf{RO}(m)$ for some message m it should be difficult to find a value $\mathsf{RO}(m^*)$ for a related message m^*. In terms of the random oracle methodology these attempts, however, only had limited success as, seemingly, the identified properties were too specific (or too cumbersome to use) in order to tempt people to design schemes directly in the standard model instead of the random oracle model.

With a recent work by Bellare, Hoang, and Keelveedhi ([BHK13:p]; BHK) this picture might finally change. At Crypto 2013 and, thus, twenty years after the formal introduction of the random oracle methodology, BHK introduced the idea of *universal computational extractors*, UCEs for short. Universal computational extractors are defined as a framework to model strong properties of random oracles in the standard model and with the explicit goal to be widely applicable. The fundamental difference to earlier works was that BHK considered UCEs to be standard-model assumptions rather than primitives that can be constructed from existing notions. This point of view allowed them to formulate notions which were, similarly to random oracles, applicable in a wide range of applications spanning all over cryptography. Indeed, in their original work BHK proposed two concrete UCE notions called UCE1 and UCE2 and they showed that a UCE2-secure hash function could take the place of a random oracle in more than ten highly interesting settings. BHK considered, amongst others, deterministic public-key encryption, message-locked encryption, universal hardcore functions with long outputs, point-function obfuscation, OAEP, symmetric encryption secure for key-dependent messages and secure under related-key attack, proofs of storage as well as adaptively-secure garbled circuits with short tokens. For many of these applications all previously known constructions were in the random oracle model and, thus, based on UCEs they obtained the first standard-model constructions of, for example, fully secure deterministic public-key encryption schemes.

With UCEs, Bellare, Hoang, and Keelveedhi focused on applicability rather than instantiability and consequently the resulting UCE assumptions are very strong. In order to gain confidence in such strong assumptions cryptanalytic validation as well as further study into the possibility of realizing UCE notions under other assumptions is necessary. BHK do a first step in this direction employing the random oracle methodology: they show that UCEs can be constructed in the ROM and explain the interpretation of this result as follows:

> "*This at first may seem like a step backwards; wasn't the purpose of UCE to avoid the ROM? As explained in more depth in Section 2, it is a step forward because the security we require from families of functions in implementations has moved from something heuristic and vague, namely to 'behave like a RO,' to something well defined, namely to be UCE-secure.*" [BHK13:p]

As BHK explain, basing a scheme on a well defined notion such as a UCE has several advantages over using the random oracle model, even if UCEs themselves can only be validated in the ROM. For one thing a well defined notion allows to be cryptanalyzed: we can try to prove that a cryptographic hash function, say a keyed version of SHA-3 or HMAC, does not meet UCE security. On the other hand, it is meaningless to show that SHA-3 or any other hash function does not *behave like a random oracle* since we know this to be the case. Besides this very practical reason, basing schemes on UCEs rather than on random oracles also provides further insight into the scheme itself, namely, it allows us to better understand what properties of the random oracle the scheme relies upon. As mentioned UCE is a framework that allows for the formulation of different notions and BHK originally suggest two specific notions UCE1 and UCE2. Thus, schemes that can be proven secure under the same notion of UCE use similar properties of the random oracle. Finally, since we know that there are schemes in the random oracle model that become insecure when instantiating the random oracle with any standard-model hash function we may regard UCEs as an additional safe guard, the hope being, that this is not the case with UCEs. In other words, if a standard-model hash function such as SHA-3 or HMAC is indeed UCE secure then we can safely use these with schemes proven down to UCEs: The proofs have the implication that we would like security proofs to have, namely, that any successful attack (within the adversarial model) against the instantiated scheme must necessarily violate a security assumption.

As of now, we are still far away from such a result. BHK have validated their UCE notions in the random oracle model and suggest to use HMAC [BCK96, KBC97] in practice. For further research efforts they suggest:

> "*We believe that achieving UCE under other assumptions is an interesting and important direction for future work. We suggest to begin by targeting restricted versions of UCE, for example UCE1 for block sources. This we may hope to achieve under first-degree assumptions. [...]. Full UCE security would, of course, require second-degree assumptions.*"[2] [BHK13:p]

[2]Here, *first-degree* assumptions refer to "standard" cryptographic assumptions that are stated relative to one global adversary. Examples include the existence of one-way functions or the existence of IND-CPA secure public-key encryption. *Second degree* (or multi-stage) assumptions, on the other hand, refer to assumptions where the adversary is split into multiple stages that do not share a common state. An example is IND-secure deterministic public-key encryption [BBO07] or UCE.

Furthermore, BHK suggest to also try to further validate the suggestion of using HMAC in place of UCE-secure functions in practice:

> "*An interesting open question is whether the assumption that HMAC provides (say) mUCE2-security [the strongest form of UCE suggested by BHK] can be validated in an idealized model where one assumes the compression function is ideal. (If not, the suggestion that it be used to instantiate UCE families in practice should be reconsidered.)*" [BHK13:f]

We get back to the topic of UCEs and random oracles shortly but next move to a topic which, at first glance, may seem rather unrelated.

Code Obfuscation

The study of *code obfuscation* asks whether we can make programs "unintelligible" while keeping the functionality intact. A successfully obfuscated program should, thus, only be as useful as a black box that can be queried on inputs to receive outputs but everything else (in particular how an output for an input is obtained) should remain completely hidden from the user.

While the idea of using code obfuscation for cryptographic purposes goes back well into the 70's—the idea is usually attributed to Diffie and Hellman [DH76] who considered obfuscation as a means of obtaining public-key encryption schemes—a formal study of code obfuscation was only started in the late 90's by Hada [Had00]. The study of code obfuscation found an early climax in 2001 with the seminal paper of Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI$^+$01, BGI$^+$12] who provided an intuitive formalization of the above idea of *black-box obfuscation* and at the same time showed that such obfuscators cannot exist in general, meaning that there are functions that are inherently unobfuscatable. Following this negative result, the interest in general-purpose obfuscation for cryptographic purposes drastically declined: the few results that were published mostly concentrated on a small sub-field, namely, obfuscation of simple point functions[3]. This changed drastically with the proposal of a candidate obfuscation scheme by Garg, Gentry, Halevi, Raykova, Sahai and Waters [GGH$^+$13b] in 2013.

Garg et al. present a candidate construction for a so-called *indistinguishability obfuscator*, a notion that is weaker than the black-box notion shown to be generally impossible by Barak et al. Indeed, the idea of an indistinguishability obfuscator was already formulated by Barak et al. in the search for workarounds to their impossibility result but had never sparked much interest as it was rather unclear whether indistinguishability obfuscation could be used in any meaningful way.

> *"Certainly, when we thought of it back then, we thought it was a useless definition."*
>
> Amit Sahai, Simons Institute, 2015 Cryptography Boot Camp

The basic idea of an indistinguishability obfuscator is that an obfuscation of a circuit hides from which specific functionally equivalent circuit the obfuscation was obtained. In more detail, if we consider two circuits C_0 and C_1 that are of the same size and compute the same function meaning that for any x we have $C_0(x) = C_1(x)$ then no efficient algorithm can distinguish between obfuscations of C_0

[3] A point function p_x is zero everywhere except on the single point x on which it evaluates to one. Note that while the result due to Barak et al. [BGI$^+$12] rules out the existence of black-box obfuscators which work for all functions, it could well be the case that black-box obfuscation for smaller classes of functions, such as point functions, exists.

and obfuscations of C_1 with good probability (i.e., no efficient algorithm can do significantly better than guessing). In other words, if iO is an indistinguishability obfuscator then the distributions

$$\mathsf{iO}(C_0) \quad \text{and} \quad \mathsf{iO}(C_1)$$

are computationally indistinguishable where the probability is over the randomness of the obfuscator.

Indeed, the security guarantee given by an indistinguishability obfuscator sounds rather vague and it is not at all clear if such a primitive can be put to good use. Contrary to this intuition Garg et al. [GGH+13b] showed how to build a functional encryption scheme for all circuits from indistinguishability obfuscation. Functional encryption[4] is a very strong form of encryption that allows to associate functions to decryption keys. If sk_f is a key with associated function f and c is a ciphertext for plaintext message m then decryption of c under key sk_f would yield the value $f(m)$. Earlier constructions of functional encryption schemes only allowed for limited functionalities and it was unclear whether a functional encryption scheme could be realized for all functions.

Being able to build such a strong primitive from indistinguishability obfuscation, as well as the candidate itself again sparked the interest of the cryptographic community in general-purpose obfuscation and countless breakthrough results followed. According to Google Scholar and as of August 2015 the candidate construction has been cited more than 250 times and a significant number of these works showed that previously unachievable notions can be realized based on obfuscation. Jumping ahead, indistinguishability obfuscation plays a crucial role also in this thesis and we discuss our contribution next.

Contribution of this Thesis

We use obfuscation, and in particular indistinguishability obfuscation to study random oracle constructions. We show both positive and negative results. The thesis is structured into three parts and we cover preliminaries and notation in Chapter 2.

Part I – Background Material

In order to lay the proper foundations we introduce both the random oracle model as well as obfuscation in great detail in Part I of this thesis. In Chapters 3 and 4 we first introduce the random oracle methodology and discuss the controversy behind random oracles in greater detail to then present various random oracle constructions that we will again encounter in later parts of the thesis. Chapter 5 provides a broad introduction to the field of general-purpose obfuscation focusing on indistinguishability obfuscators. Following, in Chapter 6 we introduce the field of point-function obfuscation. Point functions are amongst the simplest objects that we may want to obfuscate. A point function p_x for a value x is a function that evaluates to zero everywhere except on the single point x on which it evaluates to one. A simple extension of point functions is a multi-bit output point function $p_{x,m}$ for a value x and message m which, similarly, evaluates to zero everywhere except on the single point x on which it evaluates to m. Obfuscations of such simple functions have interesting applications, for example, in the domain of authentication. Furthermore, while we know that the

[4] Functional encryption started with work on attribute-based encryption schemes [SW05, GPSW06] and has evolved into a subfield of its own. For further reference see [BSW11, GGH+13b, Wee14] and the references therein.

strong form of *black-box* obfuscation cannot exist in general we do have constructions that achieve black-box obfuscation for point functions [Can97, Wee05]. Obfuscation of point functions plays an integral role for this thesis.

Part II – Random Oracle Uninstantiability

In Part II we begin our study of random oracle ambiguities, that is, we consider results of the form: *if indistinguishability obfuscation exists, then the random oracle in a given construction cannot be securely instantiated*. This means that for any standard-model hash function the construction will be insecure if used with that particular hash function in place of the random oracle model. In Chapter 7 we present an intriguing version of such a random oracle uninstantiability result. We have remarked earlier that most counter-arguments to the random oracle model are to some extent contrived and violate good cryptographic practice. We believe that our result is very different in that it is neither contrived nor does it violate any good cryptographic practice. The reason is that we do not present a specific construction but instead show that an interesting notion that has simple constructions in the random oracle model cannot be achieved *by any standard-model construction*. Note that this form of result is very different from the result presented by CGH [CGH98]. CGH construct specific signature and encryption schemes that are secure in the ROM but insecure if the random oracle is replaced by any standard-model hash function. Such an uninstantiability result highlights that the random oracle methodology as such is unsound but has no further implications for the existence of encryption or signature schemes. In contrast, we consider an interesting (and strong) form of point-function obfuscation—multi-bit output point obfuscation secure in the presence of computationally hard-to-invert auxiliary information—that has a very elegant and simple construction in the random oracle model [LPS04]. However, instead of showing that the random oracle in this particular construction cannot be instantiated we give a much stronger result: We show that if indistinguishability obfuscation exists then no standard-model construction whatsoever can achieve this notion of point-function obfuscation.[5] In that way our result can be better compared to a random oracle uninstantiability result by Nielsen who shows that the task of non-interactive, non-committing encryption is infeasible in the standard model but achievable in the random oracle model [Nie02].

Besides giving further evidence that we should be careful with random oracle constructions our result can also be interpreted in a rather different way. Our result is a one-out-of-two result that considers whether two interesting (but very different) notions of obfuscation can both exist, the answer being, we cannot have them both.[6] In that way, we can interpret the result as evidence that very strong forms of point obfuscation do not exist, or if we choose to believe that it is more likely that such forms of point obfuscation exist—after all, we do have constructions in the random oracle model and, furthermore, obfuscating point functions seems much easier than obfuscating general functions—that indistinguishability obfuscation does not exist. As in the last two years much research effort went into better understanding indistinguishability obfuscation and several new constructions have been presented (admittedly under very strong assumptions) the first interpretation seems the more likely.

[5] Note that the result by CGH holds unconditionally (assuming that public-key encryption exists) while our result is conditioned on the existence of indistinguishability obfuscation.

[6] Of course, it could also be the case that we can have neither, that is, that both indistinguishability obfuscation as well as this strong form of point obfuscation does not exist.

In the subsequent Chapter 8 we follow on the path of more traditional random oracle uninstantiability results and develop techniques that allow us to extend the result of CGH to various random oracle transformations. Random oracle transformations provide blueprints to obtain "strong" primitives in the random oracle model when starting from "weak" primitives in the standard model. Two prominent examples are the Fujisaki–Okamoto transformation [FO99] to construct CCA secure hybrid encryption schemes from IND-CPA secure schemes and the Encrypt-with-Hash transformation [BBO07] to construct deterministic public-key encryption schemes from randomized public-key encryption schemes. We show that a large classes of random oracle transformations, including the two aforementioned ones, are not sound: there exist secure schemes that are transformed into uninstantiable schemes. As with all our negative results, they are conditioned on the existence of indistinguishability obfuscation.

Part III – Universal Computational Extractors

In the final part of the thesis we turn towards positive results and show how to use indistinguishability obfuscation together with forms of point-function obfuscation to obtain standard-model constructions for a large number of interesting primitives. This brings us back to universal computational extractors which we introduce in great detail in Chapter 9.

Our first results for UCEs are negative. We show that, assuming indistinguishability obfuscation exists, then the concrete UCE notions UCE1 and UCE2 as proposed by Bellare, Hoang, and Keelveedhi ([BHK13:p]; BHK) cannot exist in the standard-model. On the positive side, we answer one of the open questions of BHK concerning HMAC and show that assuming the compression function used in HMAC is ideal (i.e., a fixed-input length random oracle) then HMAC achieves the strongest form of UCE security.[7] This further validates the usage of HMAC in place of random oracles in general and in place of UCEs in particular.

Being a framework rather than a single notion, a natural question to ask is whether weaker forms of UCE security may bypass our negative result while at the same time being able to retain (some of) the nice features and, in particular, the general applicability of the UCE1 and UCE2 notions proposed by BHK. We propose two such restrictions called UCE with respect to *statistically unpredictable sources* (short UCE[$\mathcal{S}^{\text{sup}}$]) and UCE with respect to *strongly computationally unpredictable sources* (short UCE[$\mathcal{S}^{\text{s-cup}}$]). (We introduce both the naming and notation in detail in Chapter 9.) BHK independently suggest the notion of UCE[$\mathcal{S}^{\text{sup}}$] as well as several additional notions which allows them to show that all original applications can be salvaged. We show that some of these newly suggested notions are similarly susceptible to our attacks based on indistinguishability obfuscation and we present the extended attack in Chapter 10.

Finally, in Chapter 11 we turn to showing positive results for UCEs, which yield the first standard-model candidate constructions for a variety of interesting applications. We show how to construct q-query UCE[$\mathcal{S}^{\text{sup}}$] as well as single-query UCE[$\mathcal{S}^{\text{s-cup}}$] from indistinguishability obfuscation and certain forms of point-function obfuscation. The query restriction considers a restricted form of adversary that may see only a-priory bounded number of hash values. (In the UCE definition an adversary is split into two stages where the first stage gets only oracle access to the hash function and the query restriction applies to this first stage.) Here q denotes an arbitrary polynomial that goes into the key generation algorithm. While ideally, we would like to obtain UCE security without

[7] We note that we present a more general result in [Mit14] and here extracted only the necessary steps to present the result for UCEs and HMAC.

such a query restriction we can show that for UCE[$\mathcal{S}^{\text{s-cup}}$] the single-query restriction is essentially optimal—a super-logarithmic number of queries would fall prey to an extension of our negative result. Furthermore, even under such restrictions our UCE constructions yield the first standard-model candidate constructions for a variety of interesting primitives. We obtain, amongst others, the first standard-model constructions for

- a universal hardcore function with long output,[8]
- a q-query correlation-input secure hash function, as well as
- a q-query IND-secure deterministic public-key encryption scheme.

Indeed, our UCE constructions can instantiate the random oracle in most of the schemes which have been proven secure under a UCE assumption either to obtain full security or to obtain q-query security.

Interpretation. Current candidate constructions for indistinguishability obfuscation can only be called efficient in complexity theoretic terms. That is, they are polynomial time schemes but they are far from being practical. So how can we interpret our positive results? Security proofs in the random oracle model provide a *heuristic verification* of security. Security proofs down to UCEs may provide much more, given that the form of UCE can indeed be obtained in the standard model. We view our constructions of UCEs as validation that this is possible, that is, our constructions validate the UCE assumptions. While further study is, of course, needed, UCEs provide a viable alternative to the random oracle methodology. For all practical purposes we suggest, following BHK, the use of HMAC which may well achieve some strong form of UCE security; what it cannot achieve is to behave like a random oracle.

The use of point-function obfuscation and padding. We mentioned, in passing, that our constructions of UCE secure functions rely on indistinguishability obfuscation as well as different forms of point-function obfuscation. In Chapter 12 we ask whether assuming the existence of such point-function obfuscators is necessary for our intended goals. We show that this is, indeed, the case, that is, we show that the UCE notions that we construct imply the existence of the point-function obfuscation schemes that we assumed for the construction. An interesting question that we leave for future work is whether also indistinguishability obfuscation is necessary for the existence of strong UCE notions.

A second intricacy of our constructions that we have not yet talked about is that of *padding*. In short, our constructions of UCE secure functions will consist of the indistinguishability obfuscation of a pseudorandom function. That is, if F is a pseudorandom function with key k then we construct a UCE secure function H as

$$\mathsf{H} := \mathsf{iO}(\mathsf{F}(\mathsf{k}, \cdot)).$$

[8] We note that both, the existence of a universal hard-core function (a hard-core function for any one-way function) as well as the existence of hard-core functions with long outputs for any one-way function was a long-standing open problem. While Bellare, Stepanovs, and Tessaro [BST14] recently showed how to construct hardcore functions with long output for any one-way function (based on strong obfuscation assumptions) we provide the first construction of a universal hardcore function with long output.

In other words, our UCE function will be the obfuscation (with an indistinguishability obfuscator iO) of the pseudorandom function with a hard-coded key k. However, in order for our proofs to go through we need to first artificially increase the size of the pseudorandom function before we obfuscate it. If we think of F as a Boolean circuit then this *padding operation* can be thought of as the introduction of bogus gates (no-operation gates) which do not change the functionality but only increase the size. One example would be to add pairs of NOT-gates to the first input wire. This, somewhat strange operation is necessary for our proof strategy but, on the other hand, feels to be an artifact of the proof rather than a true necessity.

In Chapter 13 we initiate a study of such padding operations in obfuscation-based techniques and formulate a framework of assumptions called *superfluous padding assumptions* (SuPA) that intuitively state that padding in certain situations is not really necessary. While it can be shown that, in general, padding is indeed a necessary evil [Hol15, BCC+14], it might be that, for some restricted cases, padding is indeed superfluous in which case also our earlier constructions would be lifted from achieving q-query security to full security. Whether or not this is the case is an open research question and we hope that with our work here we spark some interest in the cryptographic community to further investigate the role of padding. Finally, and going in a similar direction, in the final chapter of the thesis (Chapter 14) we present a direct construction of a q-query deterministic public-key encryption scheme from indistinguishability obfuscation which has certain advantages over the indirect constructions via UCEs. In particular, it seems that padding plays a much less prominent role and that the construction, thus, provides an interesting case study for restricted versions of SuP assumptions.

How to Read this Thesis

As highlighted in the previous section this thesis contains many results and not everybody may be interested in all of them. I have tried to write the thesis such that the presentation of different results is mostly self-contained. In Chapter 2, I present the general notation used within this work. I tried to make the background material presented in Part I accessible also to non-experts in the field and hope that Chapters 3, 5, and 6 can serve as surveys introducing the random oracle methodology, general-purpose obfuscation and point-function obfuscation. The expert reader, on the other hand, can safely skip most of the background material. I suggest to read the introduction of each chapter which I close with pointers to the most important definitions for later results. Parts II and III are mostly self-contained, the exception being that the negative results for UCEs reuse ideas presented earlier for random oracle uninstantiability results. Furthermore, also within Parts II and III, individual chapters should be intelligible on their own, the exceptions being that within Part III the first first chapter (Chapter 9) introduces UCEs and, thus, forms the basis for all subsequent chapters. Of course, even though individual parts should be intelligible on their own the thesis is ultimately meant to be read cover to cover and I would like to wish you an enjoyable read.

Chapter 2

Preliminaries

> "Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."
>
> Lewis Carroll, Alice in Wonderland

Summary. In this chapter we introduce the notation that we use throughout this thesis as well as fundamental concepts. Readers that are familiar with basic cryptographic and complexity theoretic concepts as well as the standard notation used in cryptographic papers can safely skip this section as we introduce more specific concepts later when needed. (In addition, in Appendix A we present definitions of concepts that we only encounter in passing.) We try to be self-contained and provide a detailed build up of the concepts that we use and develop but, of course, we cannot replace a textbook on cryptography. We refer the interested reader to [KL07, Gol00, AB09] for further information on cryptographic concepts and complexity theory.

Chapter content

2.1 Notation

If $n \in \mathbb{N}$ is a natural number then we denote its unary representation by 1^n and by $\langle n \rangle_\ell$ its binary representation using ℓ bits (for $\ell \in \mathbb{N}$ and $n < 2^\ell$). We write $[n]$ to represent the set $\{1, 2, \ldots, n\}$ and capture the (closed) real interval of all values between i and j by $[i, j]$, that is $[i, j] := \{x \in \mathbb{R} : i \leq x \leq j\}$. We denote the set of all bit-strings of length ℓ by $\{0,1\}^\ell$, the set of all bit-strings of finite length by $\{0,1\}^*$, the length of $x \in \{0,1\}^*$ by $|x|$, the concatenation of two strings $x_1, x_2 \in \{0,1\}^*$ by $x_1 \| x_2$, and the exclusive-or of two strings $x_1, x_2 \in \{0,1\}^*$ that have the same length by $x_1 \oplus x_2$. The i-th bit of a string x is selected by $x[i]$ and by $x[i..j] = x[i] \| \ldots \| x[j]$ we denote the substring consisting of bit i up to and including bit j of x. If $x, y \in \{0,1\}^*$ are two bit strings of the same length, then we denote their inner product over $\mathbb{GF}(2)$ by $\langle x, y \rangle$, that is, $\langle x, y \rangle := \oplus_{i=1}^{|x|} x[i] \cdot y[i]$. We write ε to denote the empty string. A vector of strings $\mathbf{x}$ is written in boldface, and $\mathbf{x}[i]$ denotes its i-th entry. The number of entries of $\mathbf{x}$ is denoted by $|\mathbf{x}|$. For a finite set

X, we denote the cardinality of X by $|X|$ and the action of sampling x uniformly at random from X by $x \leftarrow\!\!\$\, X$.

A real-valued function $\nu : \mathbb{N} \to \mathbb{R}_0$ is negligible if $\nu(\lambda) \in \lambda^{-\omega(1)}$ and we denote the set of all negligible functions by negl. Instead of saying that some function ν is negligible we often misuse notation and write

$$\nu(\lambda) \leq \mathsf{negl}(\lambda)$$

which should be understood in asymptotic terms. That is, there exists $\lambda_0 \in \mathbb{N}$ such that the inequality holds for all $\lambda > \lambda_0$, or in other words ν is negligible.

We call a function $\mathsf{p} : \mathbb{N} \to \mathbb{R}_0^+$ polynomial if $\mathsf{p} \in \lambda^{\mathcal{O}(1)}$ and we denote the set of all polynomials by poly. We will frequently overload the notation and refer to poly or negl as an unspecified polynomial (resp. negligible function) instead of explicitly referring to $\mathsf{p} \in \mathsf{poly}$ (or $\nu \in \mathsf{negl}$). We call a function $\delta : \mathbb{N} \to \mathbb{R}_0^+$ noticeable if there exists a polynomial poly such that for all large enough $\lambda \in \mathbb{N}$ function δ is bigger than the inverse of poly, that is, $\delta(\lambda) \geq \frac{1}{\mathsf{poly}(\lambda)}$. We call a function $\gamma : \mathbb{N} \to \mathbb{R}_0^+$ overwhelming if $|1 - \gamma(\lambda)|$ is negligible (which is used to characterize probabilities close to 1).

If E is an event then we denote by $\Pr[\mathsf{E}]$ its probability and if X is a (discrete) random variable then $\Pr[X = x]$ denotes the probability that X takes on value x. We denote the expectation of a random variable X by $\mathbb{E}[X]$ and write $X|\mathsf{E}$ to say that variable X is conditioned on event E. When not clear from context, we will specify the probability space explicitly by putting it in subscript, for example, we write $\Pr_{x \leftarrow\!\!\$\, [10]}[x = 5]$ to denote that the probability of x being 5 when x is chosen uniformly at random from the set $\{1, 2, \ldots, 10\}$. Alternatively, we may separate the probability space by a colon from the statement and separate multiple steps by semicolons: we write

$$\Pr\Big[y = 8 \;:\; x \leftarrow\!\!\$\, [10];\; y \leftarrow\!\!\$\, [2x]\Big]$$

to denote the probability that, if x is sampled uniformly from $[10]$ and then y from $[2x]$, the value y takes value 8.

We write $\wedge$, $\vee$, and $\neg$ for the Boolean operations AND, OR, and NOT (negation). We use Boolean operators both on binary strings (evaluated bit-wise)—for example, $001 \vee 010 = 011$—as well as in a logical sense, for example in probability statements. That is, $\neg\mathsf{E}$ denotes the negation of event E, and by $\mathsf{E} \wedge \mathsf{F}$ we denote the event that both events E and F occur. Within probability statements we write E, F as shorthand for $\mathsf{E} \wedge \mathsf{F}$, that is, we may write $\Pr[\mathsf{E}, \mathsf{F}]$ instead of $\Pr[\mathsf{E} \wedge \mathsf{F}]$. We make use of shorthand notation of quantifiers, that is, we use the existential quantifier denoted by $\exists$ and the universal quantifier denoted by $\forall$. Additionally, we write $\exists!$ to denote the uniqueness quantifier (there is one and only one).

By $\log(x)$ we denote the logarithm of x to base 2. Finally, we denote by $\mathrm{H}_\infty(X)$ the min-entropy of a random variable X, defined as

$$\mathrm{H}_\infty(X) := \min_{x \in \mathrm{Supp}(X)} \log(1/\Pr[X = x])$$

where the probability is over X and $\mathrm{Supp}(X)$ denotes the support of X, i.e., the set of realizations

of X that have non-zero probability.

2.2 Algorithms and Computational Models

Throughout the thesis we consider two models of computation: Turing machines and Boolean circuits. Recall that a Turing machine can process inputs of arbitrary length whereas the input length of a circuit is fixed. We denote the runtime of a Turing machine M on input x by $\mathsf{time}_{\mathsf{M}}(x)$ and its description size by $|\mathsf{M}|$. Throughout, we assume that all Turing machines terminate on every input. We denote the size of a circuit C by $|C|$ where we measure the size as the number of vertices (number of $(\neg, \wedge, \vee)$-gates plus number of input and output nodes) and we denote by the depth of the circuit the maximal length of a path from an input gate to an output gate. A *universal Turing machine* UM is a machine that takes two inputs $(\langle \mathsf{M} \rangle, x)$, interprets $\langle \mathsf{M} \rangle$ as the description of a Turing machine M and returns $\mathsf{M}(x)$. We note that we often do not explicitly distinguish between a Turing machine M and its description $\langle \mathsf{M} \rangle$. That is we my write $\mathsf{UM}(\mathsf{M}, x) = \mathsf{M}(x)$ to capture that the universal Turing machine is run on a description of Turing machine M and input x and outputs $\mathsf{M}(x)$.

A *universal circuit* UC is defined analogously working on descriptions of circuits C and inputs x. Similarly as with Turing machines we often do not explicitly distinguish between the description $\langle C \rangle$ of a circuit C and the circuit itself but rather overload notation and refer to C as both the circuit and its description. Note that the circuit model of computation is *non-uniform* capturing that circuits can only process fixed length inputs. In contrast the Turing machine model is referred to as *uniform* as a single Turing machine can process inputs of arbitrary length and thus a single Turing machine M can compute a function $f : \{0,1\}^* \to \{0,1\}^*$ by which we mean that for all $x \in \{0,1\}^*$ we have that $\mathsf{M}(x) = f(x)$. For circuits, on the other hand, we say that a function $f : \{0,1\}^* \to : \{0,1\}^*$ can be computed by a *circuit family* if there exists a sequence $(C_\lambda)_{\lambda \in \mathbb{N}}$ of circuits such that for all $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$ we have that $f(x) = C_\lambda(x)$. This restriction also applies to the universal circuit UC_λ, which also only accepts inputs (C, x) of total length λ. The universal Turing machine UM, on the other hand, can process inputs of arbitrary length, and thus in particular Turing machine descriptions of arbitrary length.

In order to simplify the presentation and abstract from the actual computational model we sometimes use the term *program* to refer to either a Turing machine or a circuit. We may, therefore, speak of a universal program UP, which denotes either a universal Turing machine UM or a universal circuit UC, and evaluates a program P on some input x (keeping in mind that when UP is replaced by a universal circuit that length restrictions apply). When defining programs, circuits or Turing machines, we use the notation $\mathsf{P}[z](\cdot)$ to emphasize that value z is hard-coded into P.

While we use the term *program* to denote a construction which could be implemented either by a Turing machine or a circuit we often simply speak of an *algorithm* which we assume is always implemented by a Turing machine.

If a Turing machine or circuit (or program) P gets access to one or more oracles $\mathcal{O}_1, \ldots, \mathcal{O}_m$ then we denote this by writing the oracles in superscript $\mathsf{P}^{\mathcal{O}_1,\ldots,\mathcal{O}_m}$. In case of oracle Turing machines we assume that the Turing machine contains one extra tape per oracle on which the machine may write a query and after going into a special state receives the answer to its query in one time step. Similarly, oracle gates to circuits are modeled to be of unit size. Instead of saying *oracle access*, we often also refer to this as *black-box* access.

We employ an explicit *security parameter* to denote the input size of a problem. When specifying a cryptographic scheme we usually describe the scheme's parameters in terms of the security parameter and we measure the security of schemes relative to the security parameter. For example, the key size of an encryption scheme could be described by a polynomial in the security parameter. In this thesis we denote the security parameter by $\lambda \in \mathbb{N}$ and give it implicitly to all algorithms in the unary representation 1^λ, even if not explicitly stated.

We assume that all algorithms run for a bounded number of steps, that is, at some point they halt and output a value (or the empty string ε). Usually we allow algorithms to run for a specified number of steps that are bounded by a polynomial in the algorithm's input size and we call such algorithms *polynomial time algorithms*. To make the runtime more explicit we measure the runtime in terms of the security parameter and call an algorithm *efficient* or PT (short for polynomial time) if it runs in time polynomial in the security parameter. By this we mean that there exists a Turing machine M that implements the algorithm, polynomials p and q and a value $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$ and inputs $x \in \{0,1\}^{\mathsf{q}(\lambda)}$ it holds that

$$\mathsf{time}_{\mathsf{M}}(1^\lambda, x) \leq \mathsf{p}(\lambda).$$

Note that the input size for efficient algorithms will also always be bounded by a polynomial in the security parameter; in the above example the input size was bounded by polynomial q while the runtime was bound by polynomial p. If the algorithm is probabilistic which is usually the case we speak of PPT (short for probabilistic polynomial time). Note that the security parameter is implicitly given as input to all algorithms (if not explicitly stated). If we say that an algorithm runs in unbounded time, then we assume that there is no time limit for the computation. However, we still assume that it always halts and outputs a value eventually.

If we speak of an algorithm we assume the algorithm is randomized, unless stated otherwise. In case the algorithm is modeled as a Turing machine then we assume that the machine has an extra input tape (the randomness tape) that is freshly initialized with a uniformly random string on each invocation of the algorithm. In case of a circuit, we assume the existence of extra input wires supplying the necessary randomness. We often speak of *the random coins* of an algorithm referring to the random bits that the algorithm uses for its computation. By $y \leftarrow \mathcal{A}(x; r)$ we denote that y was output by algorithm $\mathcal{A}$ on input x and randomness r. If $\mathcal{A}$ is randomized and no randomness is specified, then we assume that $\mathcal{A}$ is run with freshly sampled uniform random coins and we write $y \leftarrow\!\!\$\, \mathcal{A}(x)$. We often refer to algorithms, or tuples of algorithms, as adversaries.

An *adversary* is a tuple of stateful PPT algorithms. When an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consists of two stages $\mathcal{A}_1$ and $\mathcal{A}_2$, these two stages are assumed to be distinct algorithms that do *not* share any state or randomness, unless explicitly permitted to do so by a game. We discuss multi-stage adversaries in detail in Section 2.3.3.

2.2.1 Classes and Sequences of Algorithms and Distributions

If we speak of a *class* (or an *ensemble*) $\mathcal{C} := \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ of circuits (or Turing machines, programs or functions, respectively), denoted by a calligraphic letter such as $\mathcal{C}$, we mean that for each security parameter $\lambda \in \mathbb{N}$ each $\mathcal{C}_\lambda$ contains a set of circuits:

$$\mathcal{C}_\lambda := \{C_s\}_{s \in \mathsf{U}_\lambda}$$

where U_λ is usually of cardinality $2^{\mathsf{poly}(\lambda)}$ for some polynomial poly. This allows us to capture, for example, keyed functions $f_\mathsf{k} : \{0,1\}^* \to \{0,1\}^*$ for $\mathsf{k} \in \mathsf{U}_\lambda$.

We speak of a sequence of circuits $(C_\lambda)_{\lambda\in\mathbb{N}}$ (written in normal font) to denote a non-uniform circuit, that is, one circuit for every security parameter $\lambda \in \mathbb{N}$ with input size λ. We can capture a sequence of circuits by a class $\mathcal{C} := \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ where we require $|U_\lambda| = 1$. When we speak of a non-uniform algorithm this can either be a (non-uniform) circuit or a Turing machine that for each security parameter gets a (polynomial-size) advice (string of polynomial length depending only on the length of the input).

Let $\mathcal{D} = (\mathcal{D}_\lambda)_{\lambda\in\mathbb{N}}$ be a sequence of discrete probability distributions over $\{0,1\}^*$ with density function sequence $(f_\lambda)_{\lambda\in\mathbb{N}}$ such that for all $\lambda \in \mathbb{N}$ we have that $\sum_{x\in\{0,1\}^*} f_\lambda(x) = 1$. We say that $\mathcal{D}$ is samplable if there exists (possibly non-uniform) algorithm Sam such that for all $\lambda \in \mathbb{N}$ and $x \in \{0,1\}^\lambda$

$$\Pr_{r\in\{0,1\}^{\mathsf{Sam.rl}(\lambda)}}\left[\mathsf{Sam}(1^\lambda; r) = x\right] = f_\lambda(x)$$

where $\mathsf{Sam.rl} : \mathbb{N} \to \mathbb{N}$ is a function describing the number of random coins needed by algorithm Sam. If Sam is PPT (and hence $\mathsf{Sam.rl}$ is polynomial) we call distribution $\mathcal{D}$ *efficiently samplable*. We often say we *run* a distribution, or write $x \leftarrow\!\!\$\, \mathcal{D}(1^\lambda)$, to denote that the corresponding sample algorithm is invoked on fresh random coins. Finally, if X is a random variable that is distributed according to distribution $\mathcal{D}$ then we write $x \leftarrow\!\!\$\, X(\lambda)$ instead of $x \leftarrow\!\!\$\, \mathcal{D}(\lambda)$; that is x is chosen according to distribution $\mathcal{D}$. We write $x \leftarrow\!\!\$\, X(1^\lambda)$ (the security parameter is provided in unary) to indicate that the underlying distribution is efficiently samplable and $x \leftarrow\!\!\$\, X(\lambda)$ (the security parameter is provided in binary) to indicate that the underlying distribution is not necessarily efficiently samplable.

2.2.2 Statistical and Computational Distance

The distance between random variables or distributions can be measured in statistical and computational terms. If X and Y are two random variables then we define the *statistical distance* between X and Y as

$$\Delta_{X,Y}(\lambda) := \frac{1}{2}\sum_{z\in\{0,1\}^*} \left|\Pr[X(\lambda) = z] - \Pr[Y(\lambda) = z]\right|.$$

Similarly we can define the statistical distance using algorithms that run in unbounded time as

$$\Delta_{X,Y}(\lambda) = \max_{\mathsf{D}} \left|\Pr\left[\mathsf{D}(1^\lambda, X(\lambda)) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, Y(\lambda) = 1\right]\right|$$

where the probability is over random variables X and Y as well as the random coins of distinguisher D and where the maximum $\max_\mathsf{D}$ is over all (possibly unbounded) algorithms.

Analogously, we define the *computational distance* between two random variables X and Y to be the maximum distinguishing advantage of any PPT algorithm D (we usually refer to such an algorithm D as a *distinguisher*):

$$\delta_{X,Y}(\lambda) := \max_{\mathsf{D}} \left|\Pr\left[\mathsf{D}(1^\lambda, X(1^\lambda)) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, Y(1^\lambda) = 1\right]\right|$$

The probability is over random variables X and Y as well as the random coins of distinguisher D and where the maximum is over all PPT algorithms.

We say that two variables are statistically/computationally indistinguishable if the statistical/-computational distance is negligible. We write $X \approx_s Y$ (resp. $X \approx_c Y$) to denote that X and Y are statistically (resp. computationally) indistinguishable.

2.2.3 Function Families and Fundamental Cryptographic Objects

In line with [BHK13:p, BST14] we consider the following formalization of (efficient) function families: a *function family* F is a five tuple[1] of PPT algorithms ($\mathsf{F.KGen}$, $\mathsf{F.Eval}$, $\mathsf{F.kl}$, $\mathsf{F.il}$, $\mathsf{F.ol}$). The algorithms $\mathsf{F.kl}$, $\mathsf{F.il}$, and $\mathsf{F.ol}$ are deterministic and on input λ define the key length, input length, and output length, respectively. The probabilistic key generation algorithm $\mathsf{F.KGen}$ gets the security parameter 1^λ as input and outputs a key $\mathsf{fk} \in \{0,1\}^{\mathsf{F.kl}(\lambda)}$. The deterministic evaluation algorithm $\mathsf{F.Eval}$ takes as input the security parameter 1^λ, a key fk, as well as a message $x \in \{0,1\}^{\mathsf{F.il}(\lambda)}$ and generates a function value $\mathsf{F.Eval}(1^\lambda, \mathsf{fk}, x) \in \{0,1\}^{\mathsf{F.ol}(\lambda)}$. In case we consider randomized function families we model $\mathsf{F.Eval}$ also as a (probabilistic) PPT algorithm and denote by $\mathsf{F.rl} : \mathbb{N} \to \mathbb{N}$ the function that on input λ outputs the number of random coins needed by $\mathsf{F.Eval}$ when run on security parameter 1^λ.

Note that a function family and a function ensemble are different formalizations of the same object when one restricts function ensembles to consist of finite sets per security parameter and identify each function in the set with an explicit key. To simplify notation we usually drop the security parameter from invocations of $\mathsf{F.Eval}$. That is, we write $\mathsf{F.Eval}(\mathsf{fk}, x)$ instead of $\mathsf{F.Eval}(1^\lambda, \mathsf{fk}, x)$ and assume that security parameter 1^λ is implicit in key fk.

Remark. We go by the convention to provide the security parameter in binary to "length functions" such as il,ol,kl or rl. That is, we write $\mathsf{il}(\lambda)$ instead of $\mathsf{il}(1^\lambda)$. This is not to mean that these functions cannot be computed efficiently but rather that they are " helper functions".

One-wayness. A function family $\mathsf{f} := (\mathsf{f.KGen}, \mathsf{f.Eval}, \mathsf{f.kl}, \mathsf{f.il}, \mathsf{f.ol})$ of efficient algorithms is called *one-way* (or one-way function, short OWF) if for any PPT adversary (i..e, algorithm) $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{f},\mathcal{A}}(\lambda)$ defined as

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{f},\mathcal{A}}(\lambda) := \Pr_{\substack{\mathsf{fk} \leftarrow\!\$\, \mathsf{f.KGen}(\lambda) \\ x \leftarrow\!\$\, \{0,1\}^{\mathsf{f.il}(\lambda)}}} \left[\mathcal{A}(1^\lambda, \mathsf{fk}, \mathsf{f.Eval}(\mathsf{fk}, x)) \in \{x' : \mathsf{f.Eval}(\mathsf{fk}, x') = \mathsf{f.Eval}(\mathsf{fk}, x)\}\right]$$

is negligible. In other words, a function is one-way if it can be efficiently evaluated but that for a uniformly random preimage it should be difficult to invert.

One-wayness is usually defined over the uniform distribution, that is, preimage x is chosen uniformly at random. We can extend the definition and consider one-wayness for a specific distribution $\mathcal{D}$ over the preimage space $\mathsf{f.il}(\lambda)$ and say that f is one-way on $\mathcal{D}$ if for all PPT adversaries $\mathcal{A}$ we have that

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{f},\mathcal{D},\mathcal{A}}(\lambda) := \Pr_{\substack{\mathsf{fk} \leftarrow\!\$\, \mathsf{f.KGen}(\lambda) \\ x \leftarrow\!\$\, \mathcal{D}(1^\lambda)}} \left[\mathcal{A}(1^\lambda, \mathsf{fk}, \mathsf{f.Eval}(\mathsf{fk}, x)) \in \{x' : \mathsf{f.Eval}(\mathsf{fk}, x') = \mathsf{f.Eval}(\mathsf{fk}, x)\}\right]$$

is negligible. We note that if a distribution $\mathcal{D}$ is *predictable*, that is, there exists an efficient adversary $\mathcal{A}$ such that

$$\Pr_{x \leftarrow\!\$\, \mathcal{D}(1^\lambda)} \left[x = \mathcal{A}(1^\lambda)\right]$$

[1] For more specific function families additional algorithms may exist.

is non-negligible, then no function can be one-way for $\mathcal{D}$. Jumping ahead, the notion of *predictability* and *unpredictability* plays a central role in this thesis.

Pseudorandom functions and generators. We say that a function family F is *pseudorandom* (or a pseudorandom function, short PRF) if for any PPT distinguisher D—we usually refer to a binary adversary (i.e., an adversary with binary output) as a distinguisher if it is tasked with distinguishing between two worlds—we have that

$$\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathsf{D}}(\lambda) := \left|\Pr\left[\mathsf{D}^{\mathsf{F.Eval}(\mathsf{fk},\cdot)}(1^\lambda)=1\right] - \Pr\left[\mathsf{D}^{f(\cdot)}(1^\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where we denote by $\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathsf{D}}$ the *advantage* of adversary D against F as a pseudorandom function. The probability is over the random coins of D and additionally, in the first term over the random choice of key $\mathsf{fk} \in \{0,1\}^{\mathsf{F.kl}(\lambda)}$ and in the second over the random choice of a function f with domain $\{0,1\}^{\mathsf{F.il}(\lambda)}$ and range $\{0,1\}^{\mathsf{F.ol}(\lambda)}$.

In contrast to pseudorandom functions, pseudorandom generators are usually not keyed. A *pseudorandom generator* is a function family $\mathsf{G} := (\mathsf{G.Eval}, \mathsf{G.il}, \mathsf{G.ol})$ of PPT algorithms where $\mathsf{G.Eval}$ on input the security parameter 1^λ and a string of length $\mathsf{G.il}(\lambda)$ output a string of length $\mathsf{G.ol}(\lambda)$. We call G a secure pseudorandom generator if for any PPT distinguisher D we have that

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{G},\mathsf{D}}(\lambda) := \left|\Pr_{s \leftarrow\$ \{0,1\}^{\mathsf{G.il}(\lambda)}}\left[\mathsf{D}(1^\lambda, \mathsf{G.Eval}(1^\lambda, s)) = 1\right] - \Pr_{y \leftarrow\$ \{0,1\}^{\mathsf{G.ol}(\lambda)}}\left[\mathsf{D}(1^\lambda, y) = 1\right]\right|$$

is negligible.

Hash functions. Due to their diversity, hash functions are a versatile object in cryptography. The term hash function as such usually only describes a function of the form $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{\mathsf{H.ol}(\lambda)}$ that maps arbitrary bit-strings to bit-strings of a fixed length. Sometimes we also consider fixed input-length (FIL) variants which take the form $\mathsf{H} : \{0,1\}^{\mathsf{H.il}(\lambda)} \to \{0,1\}^{\mathsf{H.ol}(\lambda)}$ where usually $\mathsf{H.il}(\lambda) > \mathsf{H.ol}(\lambda)$, that is, the function is compressing. In this work we will mostly consider keyed hash functions (aka. hash function families), that is they take a (usually public) key as additional input: $\mathsf{H} : \{0,1\}^{\mathsf{H.kl}(\lambda)} \times \{0,1\}^{\mathsf{H.il}(\lambda)} \to \{0,1\}^{\mathsf{H.ol}(\lambda)}$.

We can now specify several security properties for hash functions. We can, for example, require that a hash function is a pseudorandom function or one-way (the latter is sometimes also referred to as preimage resistance in the context of hash functions). One commonly required security property of hash functions is *collision resistance*. A function family H is called collision resistant if no efficient adversary can find two inputs that map to the same image, that is, if for any PPT adversary $\mathcal{A}$ we have that the following advantage

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{H.Eval}(\mathsf{hk}, x) = \mathsf{H.Eval}(\mathsf{hk}, x') \wedge x \neq x' \;:\; \mathsf{hk} \leftarrow\$ \mathsf{H.KGen}(1^\lambda); (x, x') \leftarrow\$ \mathcal{A}(1^\lambda, \mathsf{hk})\right]$$

is negligible.

Notation. We go by the convention that f denotes a one-way function, F denotes a pseudorandom function, G is used for pseudorandom generators and H for hash functions.

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
$(\mathsf{st}, m_0, m_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda, \mathsf{pk})$
$b \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(1^\lambda, \mathsf{st}, c)$
return $(b = b' \wedge |m_0| = |m_1|)$

Figure 2.1: A formalization of the IND-CPA security notion for a public-key encryption scheme via code-based games. Note that the adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ consists of two adversarial procedures that communicate explicitly via the variable st (short for state). We say that a scheme PKE is IND-CPA secure if no efficient adversary $\mathcal{A}$ can win in the above game with probability significantly better than guessing.

2.3 Game Based Security

In this thesis we usually formalize security notions via security games as captured by the game-playing framework of Bellare and Rogaway [BR06] (with augmented game procedures as described in [RSS11]). As an example, consider the formalization of the IND-CPA security notion for public-key encryption schemes given in Figure 2.1 as a code-based security game between an encryption scheme PKE and an adversary $\mathcal{A}$. Following is a simplified description and we refer the reader to [RSS11, BR06] for further detail.

2.3.1 A Game-Playing Framework

Games consist of procedures which, in turn, consist of a sequence of statements together with some input and zero or more outputs. Procedures can call other procedures. If a procedure P gets (black-box) access to procedure F we denote this by adding it in superscript P^F. All variables used by procedures are assumed to be of local scope. (An exception are variables of the distinguished Main procedure which may be globally scoped). After the execution of a procedure the variable values are left as they were after the execution of the last statement. If procedures are called multiple times, this allows them to keep track of their state.

A game Game consists of a distinguished procedure which takes the security parameter as input. A game can make use of one or more functionalities F (a collection of procedures, for example a pseudorandom function or a public-key encryption scheme) and one or more adversarial procedures $\mathcal{A}$ (together called "the adversary"). We restrict access to adversarial procedures to the game's Main procedure, i.e., only it can call adversarial procedures and, in particular, adversarial procedures cannot call one another directly.

By $\mathsf{Game}^{\mathcal{A}}_{\mathsf{F}}$ we denote a game using the functionality F and adversary $\mathcal{A}$.[2] We denote by $\mathsf{Game}^{\mathcal{A}}_{\mathsf{F}}(\lambda) = y$ the event that Game produces output y, that is procedure Main returns value y. If Game uses any probabilistic procedure then $\mathsf{Game}^{\mathcal{A}}_{\mathsf{F}}$ is a random variable and by $\Pr\left[\mathsf{Game}^{\mathcal{A}}_{\mathsf{F}}(\lambda) = y\right]$ we denote the probability (over the combined randomness space of the game) that it takes on value y. We usually encounter binary games (i.e., games that output a binary value); games that either

[2]The fact that games are not necessarily efficient is hinted at by not supplying the security parameter in unary notation, that is we write $\mathsf{Game}(\lambda)$ rather than $\mathsf{Game}(1^\lambda)$. As usual, we also often omit explicitly providing the security parameter.

output 0 or 1. In this case we write $\Pr\left[\mathsf{Game}_{\mathsf{F}}^{\mathcal{A}}(\lambda) = 1\right]$ (or short $\Pr\left[\mathsf{Game}_{\mathsf{F}}^{\mathcal{A}}(\lambda)\right]$) to denote the probability that the game outputs 1. Following are game-based formalizations for the one-wayness (left) and pseudorandom function (right) properties defined in the previous section:

$\mathsf{OWF}_{\mathsf{f}}^{\mathcal{A}}(\lambda)$	$\mathsf{PRF}_{\mathsf{F}}^{\mathsf{D}}(\lambda)$	$\textsc{RoR}[\mathsf{fk}, b](x)$
$\mathsf{fk} \leftarrow\!\$\, \mathsf{f.KGen}(1^\lambda)$	$\mathsf{fk} \leftarrow\!\$\, \mathsf{F.KGen}(1^\lambda)$	**if** $b = 0 \wedge T[x] = \bot$ **then**
$x \leftarrow\!\$\, \{0,1\}^{\mathsf{f.il}(\lambda)}$	$b \leftarrow\!\$\, \{0,1\}$	$\quad T[x] \leftarrow \mathsf{F.Eval}(\mathsf{fk}, x)$
$y \leftarrow \mathsf{f.Eval}(\mathsf{fk}, x)$	$b' \leftarrow\!\$\, \mathsf{D}^{\textsc{RoR}[\mathsf{fk},b]}(1^\lambda)$	**elseif** $T[x] = \bot$
$x' \leftarrow\!\$\, \mathcal{A}(1^\lambda, \mathsf{fk}, y)$	**return** $(b = b')$	$\quad T[x] \leftarrow\!\$\, \{0,1\}^{\mathsf{F.ol}(\lambda)}$
$y' \leftarrow \mathsf{f.Eval}(\mathsf{fk}, x')$		**return** $T[x]$
return $(y = y')$		

Note that the PRF game uses an additional procedure RoR (short for real or random) which keeps state, that is, it remembers queries the adversary already made in table T which is assumed to be empty at the beginning of the game. We can now rewrite advantage $\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{f},\mathcal{A}}(\lambda)$ of an adversary $\mathcal{A}$ against the one-wayness of a function f as

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{f},\mathcal{A}}(\lambda) = \Pr\left[\mathsf{OWF}_{\mathsf{f}}^{\mathcal{A}}(\lambda) = 1\right]$$

and similarly advantage $\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathsf{D}}(\lambda)$ as

$$\mathsf{Adv}^{\mathrm{prf}}_{\mathsf{F},\mathsf{D}}(\lambda) = \left|2 \cdot \Pr\left[\mathsf{PRF}_{\mathsf{F}}^{\mathsf{D}}(\lambda) = 1\right] - 1\right|.$$

For this also note that while in the one-wayness setting an adversary needs to invert in order to win game OWF in the pseudorandom function setting it can win with probability $\frac{1}{2}$ by simply guessing bit b.

2.3.2 Game Hopping

We often deploy a proof strategy referred to as *game hopping* where we start from a distribution formalized as a game which we slowly turn into a different distribution by changing the game step by step. For this we describe the employed games both in text and in pseudocode. Here is an example where we show that given an IND-CPA secure public-key encryption scheme we can replace truly random coins by pseudorandom coins. We describe the games next and give the corresponding pseudocode in Figure 2.2. Both in the pseudocode as well as in the textual description we note the reduction targets for each step via arrows going from game to game.

PRG

$\mathsf{Game}_1(\lambda)$**:** The game is identical to the IND-CPA game as given in Figure 2.1.

↓

$\mathsf{Game}_2(\lambda)$**:** The game is as before except that the random coins for the encryption operation are no longer sampled uniformly at random but instead a short seed s in the domain of pseudorandom generator G is sampled and the random coins are then set to $r \leftarrow \mathsf{G}(s)$.

Finally, we would show that the distributions described by games Game_1 and Game_2 are computationally indistinguishable. In this case we can reduce the distinguishing probability of any distinguisher to the security of the pseudorandom generator as follows. We construct an adversary

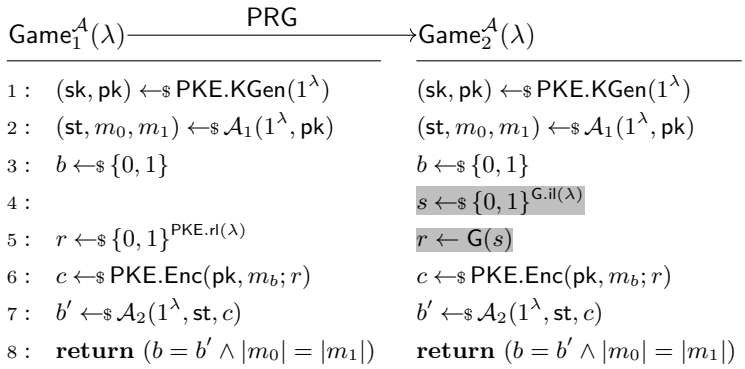


Figure 2.2: Exemplifying a game hop. The changes from game to game are highlighted. In the example in Game_2 the randomness is now obtained via generating a short seed which is then expanded via the application of a pseudorandom generator. Down to the security of the PRG the two distributions are computationally indistinguishable and thus any efficient adversary that has non-negligible advantage in the setting provided by Game_2 also has non-negligible advantage in the setting provided by Game_1.

D against G which gets as input a value r which is either uniformly sampled in $\{0,1\}^{\mathsf{G.ol}(\lambda)}$ or which is sampled as a short uniformly seed $s \in \{0,1\}^{\mathsf{G.il}(\lambda)}$ and then computed by setting $r \leftarrow \mathsf{G}(s)$. Our distinguisher D executes the steps of Game_1 leaving out lines 4 and 5 and using instead its input r as the randomness for the encryption operation. In case r was chosen uniformly at random it perfectly simulates Game_1 and otherwise it perfectly simulates Game_2. It follows that

$$\left|\Pr\left[\mathsf{Game}_2^{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{G},\mathsf{D}}(\lambda)$$

which by assumption on the pseudorandom generator is negligible and, thus, concludes our example of the game hopping technique.

2.3.3 Multi-Stage Games

A characterization of games which at first glance may appear only syntactical but which may have crucial implications is the number of adversarial stages, i.e., the number of invocations of adversarial procedures that do not have access to (or can guess with better than negligible probability) the entire state and randomness of the previous adversarial procedure. The IND-CPA game displayed in Figure 2.1 is a single-stage definition, although we consider two adversarial procedures $\mathcal{A}_1$ and $\mathcal{A}_2$. The reason is that the two adversarial procedures can share their entire state: via variable st adversary $\mathcal{A}_1$ can communicate its internal state to adversary $\mathcal{A}_2$. By restricting variable st to, for example, be short (we could, for instance, require $|\mathsf{st}| \leq \lambda$) we would weaken the definition. For this note that adversaries that were successful before may no longer be valid as they might have relied upon too long state. Such a change would, thus, indeed create a multi-stage definition, to be precise, a two-stage definition.

In later chapters we will see many examples of natural cryptographic primitives which are defined via two-stage (or even multi-stage) games. One example is deterministic public-key encryption which considers public-key encryption schemes that do not use any randomness for the encryption operation. The standard security definition consists of two-stages where a first stage samples two

message vectors (which should be unpredictable) and a second stage gets ciphertexts of either the first or the second vector and which needs to guess which vector was encrypted. If we allowed the second stage to get the state of the first stage an adversary can trivially win: as it gets access to the public-key in the second stage the adversary can simply recompute the encryption and compare it to the ciphertexts that it got. As we consider a *deterministic* encryption scheme these will match if the adversary encrypts the same messages.

Finally, the reason why we should care about the number of adversarial stages is that in multi-stage scenarios some techniques of proving security may not work. One prominent example is a composition result for idealized models called *indifferentiability* which allows to replace an ideal object (for example a random oracle) by a construction from a different ideal object (for example, a fixed-input length random oracle) while ensuring that this does not hurt security. That is, if no adversary can win a game relative to ideal object Π and we have a construction C^{π}, for an ideal object π, which is indifferentiable from Π then the composition theorem of Maurer, Renner, and Holenstein [MRH04] tells us that no adversary can win the game if we replace Π by C^{π}. As shown by Ristenpart, Shacham, and Shrimpton, the indifferentiability composition theorem, however, only covers single-stage games,[3] that is, it may make a difference whether we use C^{π} or Π in a multi-stage game [RSS11]. In a later chapter (Section 9.3), we will see an example of this and defer the formal introduction of indifferentiability to there.

Wichs highlights a second difference between multi-stage and single-stage games [Wic13], namely he shows that multi-stage games may be subject to *simulatable attacks*, which effectively means that we cannot prove the security down to a standard (single-stage) cryptographic assumption, i.e., any assumption which can be characterized by a single stage-game. We note that almost all standard assumptions (e.g., DL, RSA, DDH, LWE, OWF, PRG, IND-CPA and others) are single-stage. To jump ahead, many of the primitives considered in this thesis are multi-stage and, thus, potentially subject to Wichs' simulatable attacks. We provide a more detailed description of his result in Chapter 4, Section 4.2.2.

[3] We note that Maurer et al. never claimed anything else in [MRH04].

PART I

AN INTRODUCTION TO THE RANDOM ORACLE MODEL AND CODE OBFUSCATION

Part Summary

In the following chapters we introduce the random oracle methodology as well as the concept of code obfuscation. This part provides a broad introduction to the topics and contains only few small new results. Readers familiar with the random oracle methodology (and its controversy) can safely skip Chapter 3. In Chapter 4 we introduce various cryptographic notions that we will revisit in later chapters and foreshadow some of our results. In Chapters 5 and 6 we introduce general-purpose code obfuscation and special-purpose (point function) obfuscation. Again, readers familiar with the concepts of obfuscation can safely skip both chapters, although we suggest to go over the definitions of indistinguishability obfuscation and AIPO as these will be critical for later results.

Chapter Summary

In Chapter 3 we introduce the random oracle methodology and discuss, in detail, what it means for a random oracle scheme to be (un)instantiable. We present the seminal result of Canetti, Goldreich, and Halevi on random oracle uninstantiability [CGH98] giving an adapted and to some extent simplified proof.

In Chapter 4 we present various examples of random oracle schemes. In later chapters, we will again encounter all the schemes presented in Chapter 4, either because we can show how to instantiate the random oracle or because we can show an uninstantiability result.

In Chapter 5 we introduce code obfuscation focusing on general-purpose code obfuscators. We introduce the notions of virtual black-box and virtual grey-box obfuscation as well as indistinguishability and differing-inputs obfuscation. We present relations between these various notions of general-purpose obfuscators and discuss ideas behind current constructions for indistinguishability obfuscators.

In Chapter 6 we continue our introduction into code obfuscation, now with the focus on special-purpose point obfuscation schemes. We introduce the notions of AIPO (point obfuscation secure in the presence of hard-to-invert auxiliary information) and MB-AIPO (multi-bit output point obfuscation in the presence of hard-to-invert auxiliary information), that alongside indistinguishability obfuscation play a crucial role in later chapters.

Chapter 3

The Random Oracle Methodology

"[...] it is reasonable to think of a well-constructed real-world hash function as a deterministic function that is essentially indistinguishable from a random function."

Neal Koblitz and Alfred J. Menezes, [KM06]

"They justify this bold statement by their intuition [...] intuition has to be abandoned whenever a rigorous analysis shows that it is wrong."

Oded Goldreich, [Gol06]

Summary. In this chapter we introduce the *random oracle methodology* and discuss the controversy behind random oracles. We define what we mean by *instantiating* random oracles and what is meant by *uninstantiability* of a random oracle scheme. To highlight what it means for a scheme to be uninstantiable or p-uninstantiable we present a variant of the seminal result of Canetti, Goldreich, and Halevi [CGH98] and give a somewhat simplified proof that does not require CS-proofs.

Chapter content

3.1 Introduction

The random oracle model (short ROM), introduced in 1993 by Bellare and Rogaway [BR93], is one of the most influential concepts of modern cryptography. It is based on the simple yet powerful idea that we can construct efficient functions which perfectly mimic the behavior of truly random functions.[1] Besides laying the foundation for countless theoretical works—in the 20-something years since its formal introduction, [BR93] has been cited almost 4000 times according to Google Scholar—the random oracle model also inspired various practical cryptographic constructions that we trust on a daily basis and, indeed, it provides the security basis for several important standardized cryptographic

[1] Bellare and Rogaway provided the first formal treatment of random oracles in cryptography. The usage of random functions in the design of cryptographic schemes, however, has a longer tradition and was most notably used in the work of Fiat and Shamir [FS87]. The usage of idealized objects in cryptography can be traced back to Shannon and his work on idealized ciphers [Sha49].

schemes [RFC 3447, FIPS 186-4]. The random oracle model is also one of the most controversial topics in modern cryptography. This discrepancy is easily explained. The core motivation behind the random oracle model is described by Bellare and Rogaway as:

> "*Goals which are possible but impractical in the standard setting become practical in the random oracle setting.*" [BR93]

In other words, resorting to random oracles allows us to prove schemes secure which we are otherwise incapable of proving secure. The controversy stems from the fact that random oracles are in some sense too strong. That is, random oracles allow us to prove schemes secure in the random oracle model that we know become insecure the moment we instantiate the random oracle (that is, implement the scheme in practice). Hence, if the random oracle model allows us to prove that an *insecure scheme is secure* then it is unclear what a proof of security in the random oracle model means and this uncertainty is at the base of the random oracle controversy.

In this chapter we introduce the random oracle methodology and discuss the controversy behind random oracles. We introduce the concept of what it means to instantiate a random oracle in the standard model (aka. a world without random oracles), discuss where the assumption of random oracles differs from other cryptographic assumptions and, in particular, we introduce two flavors of what it means for a random oracle to be uninstantiable.

A note for the impatient reader. As announced, we present an introduction to the random oracle methodology and the random oracle controversy in this chapter. The expert reader who is familiar with both can safely skip this introduction.

3.2 Standard Model Security

Modern cryptography follows the provable security approach. This means that we give precise mathematical definitions specifying security goals, for example, defining secure encryption via the IND-CPA property (the encryption of two chosen messages cannot be distinguished). Then, when providing a construction we prove that the construction indeed achieves the security property, usually down to a "well-studied" assumption. Thus, proving security involves three parts:

Definition. We define a security model: for example, what does *secure encryption* mean. This, in particular, requires a definition of the capabilities of an adversary.

Assumption. We state a precise (and preferably simple) assumption. These are either complexity theoretic assumptions, such as $\mathsf{NP} \neq \mathsf{P}$, or the assumed existence of objects such as one-way functions or collision-resistant hash functions, or number-theoretic assumptions such as assuming that factoring of large numbers is difficult.

Security proof. Once we have stated the concrete assumptions, we show that breaking the security of the construction violates one or more of these assumptions. Usually this involves a reduction showing that any adversary that breaks the construction can be used to efficiently break the assumption.

The above approach, proving security down to an unproven assumption, is also often referred to as *provable security*. Somewhat counter-intuitive this means that even though a scheme may come with a security proof it could be insecure in case the underlying assumption does not hold. For example, in case $\mathsf{NP} = \mathsf{P}$ and hence one-way functions do not exist most of today's cryptography would be broken (in theory). In contrast to provably secure schemes, unconditionally secure systems are proven secure down to laws of physics (usually down to information-theoretic results) and their security does not hinge on unproven assumptions. We call such systems *unconditionally secure*, *perfectly secure* or *information-theoretically secure*.[2] Perfectly secure systems have, however, limited practical applicability as they usually require very large keys; the *Vernam cipher* (aka. one-time pad) achieves perfect secrecy but requires that the (single-usage) key is at least as long as the message that is encrypted [Sha49]. A direct consequence is that no public-key encryption scheme can exist which achieves perfect security.

For practical purposes, we usually consider provable security rather than perfect security. In many cases, however, we might not quite succeed in proving the security of a scheme down to a well-studied assumption. Or sometimes we can only provide a proof of security for a less efficient version of a construction even though we have an efficient and simple construction that on an intuitive level *looks fine*. In such a situation a popular approach is to model security according to the random oracle methodology which we discuss next.

3.3 The Random Oracle Methodology

The random oracle methodology is based on the idea that we can design hash functions that are indistinguishable from random functions and, thus, in the security analysis we can replace hash functions by random functions. This has wide-ranging consequences: On the one hand, it allows us perform information-theoretic analyses. On the other hand, we need to specify how (computationally bounded) parties can interact with such a random function which has a huge (if not infinitely large) description.[3] To solve this problem we assume that the function is implemented by an oracle that every party gets access to and which evaluates the function for the party. In Section 2.2 we defined oracle machines, i.e., algorithms that operate with black-box access to some functionality (the oracle). By an oracle we usually understand a (deterministic) function that an algorithm gets black-box access to. A *random oracle* is an oracle RO that simply implements a random function from the function space $\{0,1\}^* \to \{0,1\}^{\mathsf{poly}(\lambda)}$. An instructive way to think about a random oracle is to imagine an infinite lookup table T that is initially empty. On the first query x, it is checked if $T[x] \neq \bot$, that is, whether x has already been queried. If so, value $T[x]$ is returned. Otherwise, a fresh uniformly random value $y \leftarrow\!\!\$\, \{0,1\}^{\mathsf{poly}(\lambda)}$ is chosen, stored in the lookup table as $T[x] \leftarrow y$ and returned. This process of building up the oracle on the go is also referred to as *lazy sampling*.

[2] Note that we usually speak of perfectly secure systems if the success probability of an unbounded adversary is not better than pure guessing.

[3] On average, an efficient function cannot implement a random function as a completely random function cannot be compressed significantly below the size needed to store its function table. For example, if we compactly encode all functions from $\{0,1\}^n \to \{0,1\}$ via prefix encodings then a random function from this set would have an expected length of $1 + \sum_{i=1}^{n-1} 2^{2i-n}$ bits. (The empty string encodes one function, with 1 bit we encode 2^0 functions, with 2 bits we encode 2^1 functions, and so forth until all 2^n functions are encoded.) For $n = 512$ (i.e., a function mapping from $\{0,1\}^{512}$ to one bit) this means that the expected length is greater than 2^{510} bits (or roughly $4.19 \cdot 10^{140}$TB).

The *random oracle model* now considers a world in which every party (including the construction and the adversary) have oracle access to a random oracle RO. In particular, no party can evaluate RO on its own without making an explicit query to the random oracle. Statements in the random oracle model are always probabilistic statements that hold over the uniformly random choice of the function implemented by the oracle. The *random oracle methodology* is a design methodology making crucial use of the random oracle model. It consists of two simple steps:

1. We design and prove the security of a scheme in the random oracle model where each party (including the adversary) has black-box access to a random oracle.
2. We replace the random oracle by a *good cryptographic hash function* that we assume *behaves like a random oracle*.

Pseudorandom functions in the random oracle model. As an example of how to use the random oracle model consider the following construction of a pseudorandom function family F consisting of two algorithms, a probabilistic key generation algorithm $\mathsf{F.KGen}$ and a deterministic evaluation algorithm $\mathsf{F.Eval}$. In addition we define the functions $\mathsf{F.il}, \mathsf{F.ol}$, and $\mathsf{F.kl}$ to describe the input, output and key length in relation to the security parameter $\lambda \in \mathbb{N}$. (Also see Section 2.2.3 for an introduction to function families and the definition of pseudorandom functions (PRFs).) We next define our pseudorandom function in the random oracle model by specifying key generation and evaluation:

$\underline{\mathsf{F}^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)}$	$\underline{\mathsf{F}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{fk}, m)}$
$\mathsf{fk} \leftarrow\!\!\$\, \{0,1\}^\lambda$	$y \leftarrow \mathsf{RO}(\mathsf{fk} \| m)$
return fk	**return** y

The pseudorandom function F^{RO} takes as input a message m as well as a key fk that is uniformly distributed in $\{0,1\}^\lambda$ and simply evaluates the random oracle on the concatenation $\mathsf{fk} \| m$. We next establish that F is indeed pseudorandom. For this we need to show that for any efficient adversary $\mathcal{A}$ there exists a negligible function such that for sufficiently large λ

$$\Pr\left[\mathcal{A}^{\mathsf{RO},\mathsf{F}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{fk},\cdot)}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{RO},f(\cdot)}(1^\lambda) = 1\right] \leq \mathsf{negl}(\lambda)\,,$$

where the adversary either gets oracle access to random oracle RO and F^{RO} with a random key or to the random oracle and a function f chosen uniformly at random from the set of functions $\left\{\{0,1\}^{\mathsf{F.il}(\lambda)} \to \{0,1\}^{\mathsf{F.ol}(\lambda)}\right\}$. Note that while, in the standard model, the first probability would be over the choice of fk and the coins of $\mathcal{A}$ whereas the second would be over the coins of $\mathcal{A}$ and the choice of a random function f, in the random oracle setting the probabilities are additionally also over the choice of random oracle RO.

Establishing that F is indeed a pseudorandom function is straight forward in the random oracle model. Let $Q \subset \{0,1\}^*$ be a set of bit-strings and for a function f define

$$\mathsf{f}(Q) := \{\mathsf{f}(q) : q \in Q\}\,,$$

where we assume that Q is in the preimage space of f. This allows us to view $\mathsf{RO}(Q)$ as well as $f(Q)$ as a random variable; the first over the choice of random oracle RO the second over the

choice of function f. For two sets Q_1 and Q_2 it holds that the following variables are statistically indistinguishable

$$(\mathsf{RO}(Q_1), \mathsf{RO}(\mathsf{fk}\|Q_2)) \approx_s (\mathsf{RO}(Q_1), f(Q_2))$$

unless there exists an $m \in Q_2$ such that $\mathsf{fk}\|m \in Q_1$.

Now, consider an adversary $\mathcal{A}$ that makes at most q many queries to its oracles where $\mathsf{q} : \mathbb{N} \to \mathbb{N}$ is a polynomial in the security parameter. Then, by the above argument, in order to distinguish the two worlds (having oracle access to $\mathsf{RO}(\cdot)$ and $\mathsf{F}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{fk}, \cdot)$ or having oracle access to $\mathsf{RO}(\cdot)$ and $f(\cdot)$, the adversary must query RO on a value $\mathsf{fk}\|m$ for some message m. Assuming that the first ℓ queries to RO do not have prefix fk we know that the only information the adversary has about fk is that it is different from the first ℓ prefixes. For any other possible choice (and there are at least $2^\lambda - \ell$ many other choices for a λ-bit prefix) the probability is exactly the same that it matches fk. Hence, by a union bound we get that for a randomly chosen key $\mathsf{fk} \leftarrow\!\!\$\, \{0,1\}^\lambda$ the probability that any of the q queries by $\mathcal{A}$ begins with fk is upper bounded by $\mathsf{q} \cdot 2^{-\lambda}$ which is negligible. Consequently, F is a pseudorandom function in the random oracle model.

Remark. We note that the analysis is information-theoretic and, in particular, independent of the computational power of the adversary. It only requires that the adversary cannot make "too many" queries to the random oracle.

3.3.1 Instantiating the Random Oracle

In the previous section we constructed a pseudorandom function from a random oracle and we will see many more examples of random oracle schemes throughout this thesis. Building a scheme relative to a random oracle and showing its security constitutes the first step of the *random oracle methodology*. In the second step, in order to obtain a scheme that we can actually implement,

> we replace the random oracle by a *good cryptographic hash function* that we assume *behaves like a random oracle*.

In other words, as we design our scheme in an idealized model we also need to specify how we want to deal with the ideal primitive (i.e. the random oracle) when we want to implement the scheme, for example, in hardware or in software. This "dealing with the random oracle in practice" is what is meant by *instantiating the random oracle* and what is usually done is to replace the random oracle by a cryptographic hash function that we *believe* is good. For example, we use SHA-2 or SHA-3 [FIPS 180-4, SHA-3] in place of random oracle RO. This means for our above example that by the random oracle methodology an instantiation of our pseudorandom function with SHA-3 (we denote the function by F[SHA-3]) is given by:

A pseudorandom function from hash function SHA-3 obtained by applying the random oracle methodology.

$\underline{\mathsf{F}[\text{SHA-3}].\mathsf{KGen}(1^\lambda)}$	$\underline{\mathsf{F}[\text{SHA-3}].\mathsf{Eval}(\mathsf{fk}, m)}$
$\mathsf{fk} \leftarrow\!\!\$\, \{0,1\}^\lambda$	$y \leftarrow \text{SHA-3}(\mathsf{fk}\|m)$
return fk	**return** y

A natural question to ask is whether F[SHA-3] indeed a secure pseudorandom function? In a nutshell, this is the very question this thesis revolves around: what happens when we instantiate a random oracle scheme in practice? We will get back to this question shortly.

3.3.2 Observing and Programming Random Oracles

A very powerful feature of random oracles is their so-called *programmability*. Consider a cryptographic construction C and a black-box security reduction R. A black-box reduction, in loose terms, transforms any successful adversary $\mathcal{A}$ against construction C^{RO} into an algorithm $R^{\mathcal{A}}$ that breaks a cryptographic assumption. If R is efficient and $\mathcal{A}$ is efficient, then also the combination $R^{\mathcal{A}}$ (where R gets black-box access to $\mathcal{A}$) is efficient and thus if we believe that no efficient algorithm can break the cryptographic assumption we have a security proof for construction C. Now, if C is defined relative to a random oracle C^{RO} then an adversary $\mathcal{A}$ that wants to evaluate C^{RO} must itself call the random oracle. A reduction can leverage the random oracle now in (at least) two ways. Note that, before, the reduction only had black-box access to adversary $\mathcal{A}$ and could thus only observe its input and output behavior; that is, on supplying an input it sees the output of the adversary. Now, however, as the adversary needs to evaluate the random oracle for evaluating the construction the reduction can simulate the random oracle for the adversary, for example, via lazy sampling. In this way, all the adversary sees is a genuine random oracle, while the reduction suddenly gains extra insight into the adversary's inner workings: it sees the adversary's random oracle queries.

While *observability* is already a powerful proof technique we note that a reduction has even more power: when simulating the random oracle for the adversary it can choose the random oracle values that it provides to the adversary as long as towards the adversary they seem uniformly distributed. This second property is also called *programming* the random oracle and is a powerful technique that many random oracle proofs rely on. Two prominent examples are the random oracle proofs for full-domain hash signatures [BR93] as well as the Fiat–Shamir paradigm [FS87, Oka93, PS00, AABN02].[4]

3.3.3 Domain Separation for Random Oracles

It is instructive to ask whether two random oracles are more powerful than a single random oracle, that is, if relative to two independent random oracles we can construct more powerful or more efficient schemes than if we restrict ourselves to a world with a single oracle. It is easy to see that this is not the case as we can always simulate multiple random oracles with a single one. To see this consider an algorithm $\mathcal{A}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_\ell}$ which has access to ℓ random oracles. We can construct an algorithm $\mathcal{A}_*^{\mathsf{RO}}$ which only has access to a single random oracle but which implements an identical distribution on any input as algorithm $\mathcal{A}$. For this, $\mathcal{A}_*$ executes $\mathcal{A}$ and answers query q to the i-th oracle as

$$\mathsf{RO}(\mathsf{prefix}_i \| q)$$

where prefix_i is a unique prefix corresponding to index i, for example, $\mathsf{prefix}_i := \langle i \rangle_{\lceil \log(\ell) \rceil}$, that is, a binary encoding of i using $\lceil \log(\ell) \rceil$ many bits. As RO is a random function, so is the projection $\mathsf{RO}(\mathsf{prefix}_i \| \cdot)$ and thus the simulation of $\mathcal{A}_*$ is perfect.

[4] It was shown that for Fiat–Shamir the *weakly programmable* random oracle suffices [FLR+10, FF13] where the program capabilities are somewhat restricted.

This technique of obtaining multiple random oracles from a single random oracle is usually referred to as *domain separation*.

3.3.4 Random-Oracle Schemes vs. Random-Oracle Transformations

The above example of a pseudorandom function from a random oracle is what we call a *cryptographic construction* (or cryptographic scheme). We speak of a cryptographic construction, if it is of the type: "if assumption A holds, then construction C fulfills security definition D". In case a construction is based on the random oracle methodology then we call it a *random oracle construction* or *random oracle scheme*.

Cryptographic transformations are special types of constructions that are only indirectly based on an assumption. They usually take a cryptographic scheme and *transform* it into a new scheme that fulfills additional or other security definitions. For a transformation T from a scheme C, we write $\mathsf{T}[C]$. Two prominent examples of cryptographic transformations are the construction of pseudorandom functions from pseudorandom generators due to Goldreich, Goldwasser, and Micali [GGM84] usually simply referred to as the GGM-construction (although more appropriate with our terminology would be to call it the GGM-transformation) and the HILL-construction (i.e. transformation) of pseudorandom generators from one-way functions, due to Håstad, Impagliazzo, Levin, and Luby [HILL99].

By transforming a cryptographic scheme into a different one, transformations indirectly transform security properties into one another. This allows us to study the relationship between assumptions, for example, arguing that two assumptions are equivalent, or that one assumption is strictly weaker than another. Taking the GGM and HILL transformations we can argue that assuming the existence of pseudorandom functions is no stronger than assuming one-way functions since we can transform any one-way function into a pseudorandom generator (HILL) and the result into a pseudorandom function (GGM).

Random-oracle transformations. The picture becomes more complicated once we introduce ideal models. Without random oracles, if we have a construction C_1 and a transformation from C_1 to some scheme C_2 then we can argue that assuming the existence of C_2 (for example, pseudorandom functions) is not stronger than assuming the existence of C_1 (for example, one-way functions). Now, given a random oracle transformation T^{RO} (i.e., transformation T uses a random oracle RO) from a scheme C_1 to a scheme $C_2 := \mathsf{T}^{\mathsf{RO}}[C_1]$ this conclusion does not hold anymore. Assume that C_1 does not use random oracles. In this case, if we wanted to assume the existence of C_2 via the above argument we would need to assume a) that C_1 exists and b) that there exists a hash function which (to some extent) behaves like a random oracle. In particular, it is unclear which properties of the random oracle are necessary for C_2 to be secure. Baecher et al. recently presented techniques to relate schemes proven in idealized models [Bae14, BF11, BFFS13]. Here, we focus on what happens once we instantiate the random oracle (for example in a transformation) and whether we can provide instantiations of the random oracle which result in a provably secure scheme.

3.3.5 Keyed Random Oracles

Most random-oracle transformations and schemes in the literature are analyzed in the "unkeyed" random oracle model, and this reflects the fact that a fixed unkeyed hash function will be used in their instantiations. Keyed hash functions, however, are more powerful and in particular it is

difficult to capture many standard security properties for unkeyed hash-functions: an example is collision-resistance. A fixed and unkeyed hash function cannot be collision-resistant simply because collisions must exist and hence there is a trivial efficient adversary, namely the adversary that has a collision hard-coded. (Note that this adversary would be non-uniform as it has a collision for each security parameter hard-coded.)

When it comes to instantiating random oracles with a *keyed* hash function we need to decide how the hash key is generated and who gets access to it as the construction in the random oracle model does not capture this. For example, if we consider a symmetric encryption scheme in the random oracle model, then the hash key could be part of the key-generation process in which case it remains hidden from the adversary, or it could be a parameter generated during set-up, in which case it would be available to the adversary. We, therefore, consider a generalization of the standard random oracle model whereby all parties get access to a *keyed* random function. More precisely, in the $(\mathsf{kl}, \mathsf{il}, \mathsf{ol})$-ROM, where functions $(\mathsf{kl}, \mathsf{il}, \mathsf{ol})$ specify key-length, input-length and output-length for each security parameter (see also Section 2.2.3), on security parameter λ all parties get access to a random function of the form

$$\mathsf{RO}(\cdot,\cdot) : \{0,1\}^{\mathsf{kl}(\lambda)} \times \{0,1\}^{\mathsf{il}(\lambda)} \longrightarrow \{0,1\}^{\mathsf{ol}(\lambda)} \ .$$

Note that we recover the standard unkeyed random oracle model when $\mathsf{kl}(\lambda) = 0$ (there is only one key ε). In defining the security of a cryptosystem, the underlying probability space is extended to include a random choice of a keyed function (and choices of hash keys as specified by the scheme). Whether or not a party gets to see the hash key depends on the specification of the scheme and its security model. For instance, if a keyed ROM scheme includes hash keys under its public keys, an honest or malicious party gets to see the hash key whenever it gets to see the public key.

3.4 Random Oracles vs. Standard Model Security

Contrary to instantiated random oracle schemes, we call schemes that follow the provable security approach, that is, they were designed and proven secure without random oracles (or other idealized objects) *standard model schemes* and speak of the *standard model* to capture a world without any idealized objects (cf. Section 3.2). It is easily seen that a proof of security for a scheme with random oracles is not a proof for the instantiated scheme. In particular, even though a scheme is secure in the random oracle model, an instantiation (or even every possible instantiation) may be completely insecure. For this consider the following (pathological) variant of our *random oracle pseudorandom* function:

$\underline{\widetilde{\mathsf{F}}^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)}$	$\underline{\widetilde{\mathsf{F}}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{fk}, m)}$
$\mathsf{fk} \leftarrow\!\!\$\ \{0,1\}^\lambda$	**if** SHA-1$(\mathsf{fk}, m) = \mathsf{RO}(\mathsf{fk} \Vert m)$ **then**
return fk	$\quad y \leftarrow \mathsf{fk}$
	else $y \leftarrow \mathsf{RO}(\mathsf{fk} \Vert m)$
	return y

Now, the scheme first checks if an evaluation on SHA-1 is equivalent to what its oracle does and if so it returns the key. Hence if we instantiate the above with SHA-1 we trivially get an insecure scheme.

Bellare and Rogaway warned against such counter examples and argued that a scheme needs to be independent from the function that is used to instantiate the oracle:

> *"Our thesis is that an appropriate instantiation for a random oracle ought to work for any protocol which did not intentionally frustrate our method by anticipating the exact mechanism which would instantiate its oracles."* [BR93]

If this is ensured the hope was that there could be an efficient function which behaves sufficiently like a random oracle and makes all hash-function independent schemes secure.

Regrettably, this is not the case and we will shortly see examples of schemes which are secure in the random oracle model, which are independent of a concrete hash function but which cannot be made secure in the standard model.

3.4.1 Uninstantiability

In the following we lift the study of *instantiating a random oracle scheme* onto formal grounds. In particular we are interested in whether there exists a standard model hash function which makes a random-oracle scheme secure when the random oracle is replaced by this hash function. If such a function exists, we call the random-oracle scheme *instantiable* and say that this particular hash function securely instantiates the scheme. On the other hand, if no such standard-model function exists which when replacing the random oracle yields a secure scheme we call the the scheme *uninstantiable*. Note that the above tweaked scheme $\widetilde{\mathsf{F}}^{\mathsf{RO}}$ which first compares the result of its oracle on the input to that of the SHA-1 function does not necessarily qualify for an uninstantiable scheme. The scheme is certainly not secure when instantiated with SHA-1, but it may well be secure when instantiated with, say, SHA-3.[5]

Given a scheme in the keyed ROM, we consider its standard-model instantiations via keyed hash functions. Formally, this entails that: (1) using a keyed hash function that has key, input and output lengths that are identical to those of the (keyed) random oracle, (2) running the key-generation algorithm whenever a hash key is generated in the ideal scheme, and (3) calling the evaluation routine of the hash function whenever an oracle query is placed. Given a keyed ROM scheme, a security property (e.g., pseudorandomness) and a security proof showing that the random oracle scheme achieves the property, we say that the scheme is *instantiable* if there exists a standard-model hash function which when replacing the random oracle as described above yields a standard-model scheme that is secure, i.e., can be proven to also achieve the security property. Conversely, we say that a scheme is *uninstantiable* if no such standard-model hash function exists. Finally, for a polynomial bound p, we call a scheme p-*uninstantiable*, if no hash function of size at most $\mathsf{p}(\lambda)$ can securely instantiate the scheme.

3.4.2 The Random Oracle Methodology, Revisited

The study of (un)instantiability of random oracle schemes was initiated by Canetti, Goldreich, and Halevi [CGH98] with their seminal paper "The Random Oracle Methodology, Revisited" which appeared five years after the introduction of the random oracle methodology. In short, CGH showed

[5] Note that SHA-3 in its basic form is not a keyed hash function. We can constructed a keyed variant, for example, via HMAC [BCK96, KBC97].

the existence of (contrived) encryption and signature schemes which are secure in the random oracle model, but become insecure when being instantiated with any standard-model hash function. In their schemes, CGH exploit one of the fundamental differences between random oracles and standard-model hash functions, namely that the latter have an efficient description (i.e., a polynomial size description) while a truly random function's description is of exponential size (or even has no finite description in case its domain is infinite). This means, in particular, that an adversary against the instantiated scheme may ask for an encryption (or signature) for a message equivalent to the code of the hash function which was used to instantiate the random oracle.[6] If the scheme is such that it first checks if its input is a valid program description and if so whether the program behaves identical to the scheme's oracle then we have a similar situation as with our earlier pathological scheme $\widetilde{\mathsf{F}}^{\mathsf{RO}}$ except that now the hash function is no longer built into the scheme. Let us illustrate the idea by giving an adapted version of the CGH result for symmetric encryption schemes. We note that CGH show their result for public-key encryption schemes and signature schemes. A simplified proof for the case of signature schemes appears in [MRH04].

We construct a symmetric encryption scheme SE from our random-oracle pseudorandom function F^{RO} (see Section 3.3) by using it to "generate keys for a one-time-pad encryption[7]":

$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)}$	$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{Enc}(\mathsf{k}, m)}$	$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{Dec}(\mathsf{k}, (c, r))}$
$\mathsf{k} \leftarrow\!\!\$\, \{0,1\}^\lambda$	$r \leftarrow\!\!\$\, \{0,1\}^\lambda$	$m \leftarrow \mathsf{RO}(\mathsf{k}, r) \oplus c$
return k	$c \leftarrow \mathsf{RO}(\mathsf{k}, r) \oplus m$	**return** m
	return (c, r)	

We will later require that the adversary is able to get encryptions for messages of arbitrary (polynomial) length. We note that with a random oracle that has shorter output length than the message size we can always simulate a matching random oracle by running the random oracle in counter mode, i.e.,

$$\mathsf{RO}_{|m|}(\mathsf{k}, r) := \Big(\mathsf{RO}(\mathsf{k}, r \| \langle 1 \rangle) \| \mathsf{RO}(\mathsf{k}, r \| \langle 2 \rangle) \| \ldots \mathsf{RO}(\mathsf{k}, r \| \langle \ell \rangle)[1 \ldots |m|]\Big)$$

where $\mathsf{RO}_{|m|}$ denotes the extended random oracle (also see Section 3.3.3 on domain separation). Also note that we have stated the above scheme in the keyed random oracle model, that is, we write $\mathsf{RO}(\mathsf{k}, r)$ (instead of, for example, prepending the key to the message).

In the random oracle model, the above scheme provides even information-theoretic security (given that an adversary is restricted to at most polynomially many queries to the random oracle). An IND-CPA adversary against the above scheme will get access to a left-or-right oracle LoR which is configured with a hidden bit b and a random key k (indicated by writing them in square brackets) and which returns either encryptions of the first or the second message.

[6] For an example of where such an encryption might be obtained in a practical attack is for example disk encryption.

[7] A one-time-pad is a single use, unconditionally secure, encryption scheme where a message is encrypted with a uniformly random key that has the same length as the message and which may be used only once. If $m \in \{0,1\}^*$ is a message, and $\mathsf{k} \leftarrow\!\!\$\, \{0,1\}^{|m|}$ is a uniformly random key, then an encryption is computed as $c \leftarrow m \oplus \mathsf{k}$ which can be again decrypted as $m \leftarrow c \oplus \mathsf{k}$.

$\underline{\mathsf{LoR}[b,\mathsf{k}](m_0,m_1)}$

if $|m_0| \neq |m_1|$ **then**
  return $\perp$
$(c,r) \leftarrow\!\!\$\ \mathsf{SE}^{\mathsf{RO}}.\mathsf{Enc}(\mathsf{k}, m_b)$
return (c,r)

The adversary's task is to learn bit b from its interaction with the oracle. In the random oracle setting an adversary will fail with overwhelming probability since unless the adversary correctly guesses key k values $\mathsf{RO}(\mathsf{k},r)$ are uniformly random and, hence, the scheme reduces to an instance of a one-time-pad.

In the following, we will change the above symmetric encryption schemes using the ideas and techniques of CGH such that the resulting scheme remains secure in the random oracle model but such that we can prove that no instantiation of the scheme will be secure. For this, we adapt the encryption algorithm to perform one additional check and, if the check succeeds, to output the secret key k. In a first step, we adapt encryption and decryption to add some bogus text into the ciphertext which is ignored on decryption:

$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{Enc}(\mathsf{k},m)}$	$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{Dec}(\mathsf{k},(c,r,s))}$
$s \leftarrow\!\!\$\ \{0,1\}^\lambda$	$m \leftarrow \mathsf{RO}_{\lvert c\rvert}(\mathsf{k},r) \oplus c$
$r \leftarrow\!\!\$\ \{0,1\}^\lambda$	**return** y
$c \leftarrow \mathsf{RO}_{\lvert m\rvert}(\mathsf{k},r) \oplus m$	
return (c,r,s)	

We have highlighted the changes from the original scheme. Also note that the decryption scheme simply ignores the additional ciphertext field s. As value s is simply a uniformly random value the security of the scheme remains intact.

In a next step we further change our scheme and introduce a backdoor which when triggered leaks the scheme's secret key as part of the newly introduced field s. For this we adapt our scheme as follows:

$\underline{\mathsf{SE}^{\mathsf{RO}}.\mathsf{Enc}(\mathsf{k},m)}$	$\underline{\mathsf{check}^{\mathsf{RO}}(\mathsf{k},\mathsf{P})}$
if $\mathsf{check}^{\mathsf{RO}}(\mathsf{k},m)=1$ **then** $s \leftarrow \mathsf{k}$	**if** P is valid encoding of program **then**
else $s \leftarrow\!\!\$\ \{0,1\}^\lambda$	$x \leftarrow\!\!\$\ \{0,1\}^\lambda$
$r \leftarrow\!\!\$\ \{0,1\}^\lambda$	**if** $\mathsf{RO}(\mathsf{k},x) = \mathsf{UP}(\mathsf{P},(\mathsf{k},x))$
$c \leftarrow \mathsf{RO}_{\lvert m\rvert}(\mathsf{k},r) \oplus m$	**return** 1
return (c,r,s)	**return** 0

We added an extra Boolean procedure $\mathsf{check}^{\mathsf{RO}}$ which is called as the first step by the encryption procedure on input key k and message m. In case procedure check succeeds, that is, if it returns 1 then the previously introduced field s is set to hold the secret key k. Procedure check first tests whether its second input P (i.e. message m) encodes a valid program.[8] For the moment we abstract from the specific computational model (Turing machines or circuits) and speak of programs (also see Section 2.2). Note that for both Turing machines and circuits we can fix an encoding such that it can be efficiently checked whether a string is a valid encoding or not. If this test succeeds procedure

[8]Note that program P is not given oracle access to the random oracle.

check chooses a random bit-string $x \in \{0,1\}^\lambda$ and computes $\mathsf{RO}(\mathsf{k}, x)$. Additionally it evaluates program P on inputs k and x, to obtain value $\mathsf{P}(\mathsf{k}, x)$. Note that the latter is computed via the universal program

$$\mathsf{UP}(\mathsf{P}, (\mathsf{k}, x)) = \mathsf{P}(\mathsf{k}, x)$$

If the two computations have the same result, i.e., if $\mathsf{RO}(\mathsf{k}, x) = \mathsf{P}(\mathsf{k}, x)$, then procedure check returns 1. Otherwise, it returns 0.

Let us analyze our scheme in the random oracle model. In order for procedure check to return 1 an adversary must provide an encoding of a program P which "guesses" the result of the random oracle on a random point x. Since x is chosen fresh and uniformly at random for each check an adversary can predict value $\mathsf{RO}(\mathsf{k}, x)$ only with probability $\max\{2^{-\mathsf{RO.ol}(\lambda)}, 2^{-\lambda}\}$ which is assuming the adversary knows key k. Hence, with overwhelming probability procedure check returns 0 on all invocations and the security of the scheme remains intact.

Let us next analyze the scheme when the random oracle is instantiated with a standard-model hash function H. We denote the resulting standard-model scheme by $\mathsf{SE}[\mathsf{H}]$. In this case random oracle RO is also replaced by function H within procedure check which becomes:

$\mathsf{check}[\mathsf{H}](\mathsf{k}, \mathsf{P})$

if P is valid encoding of program **then**
  $x \leftarrow\!\!\$\ \{0,1\}^\lambda$
  if $\mathsf{H}(\mathsf{k}, x) = \mathsf{UP}(\mathsf{P}, (\mathsf{k}, x))$
    return 1
return 0

Assume that an adversary chooses its message as $\langle \mathsf{H} \rangle$, that is, as an encoding of function H and note that we did not restrict the input length to Enc and check. Then, procedure check will test whether

$$\mathsf{H}(\mathsf{k}, x) = \mathsf{UP}(\langle \mathsf{H} \rangle, (\mathsf{k}, x))$$

which will always evaluate to true since

$$\mathsf{H}(\mathsf{k}, x) = \mathsf{H}(\mathsf{k}, x).$$

Thus, check returns 1 and as a result scheme $\mathsf{SE}[\mathsf{H}]$ sets $s \leftarrow \mathsf{k}$ which allows the adversary to learn key k which, in turn, allows it to win with probability 1.

We have established that in the random oracle model secure symmetric encryption schemes exist (note that we did not use any assumption besides the existence of random oracles) but that some of these become insecure once instantiated with any standard model hash function. There is one detail in the above argument which we neglected: the choice of computational model for program P. We discuss how the choice of a computational model affects the above argument next.

3.4.3 The Choice of Computational Model and the Type of Uninstantiability

In the previous section we abstracted from the choice of a computational model, that is, we thought of scheme SE to be modeled as a program and in particular we thought of input P to procedure

check as a program which can be evaluated by a universal program UP. We now discuss in turn the consequences when choosing a specific computational model.

We first consider the case that the above symmetric encryption scheme SE is modeled using Boolean circuits. In this case we have a circuit $\mathsf{SE.Enc}_\lambda$ for each security parameter $\lambda \in \mathbb{N}$ and hence also a circuit check_λ for each $\lambda \in \mathbb{N}$. As a circuit has a fixed input length both circuits $\mathsf{SE.Enc}_\lambda$ and check_λ take inputs of a fixed length $\mathsf{SE.il}(\lambda)$. As our adversary chooses message m to be an encoding $\langle \mathsf{H} \rangle$ of hash function H, it is now restricted to encodings of length $\mathsf{SE.il}(\lambda)$. If the hash function used in the instantiation of the random oracle does not admit for a short description the scheme can no longer be attacked in the way described above. Hence, utilizing circuits as our computational model we get a weaker p-uninstantiability result of the form

Theorem 3.1. *For each polynomial* p *there exists a symmetric encryption scheme which is secure in the random oracle model but which cannot be instantiated by any hash function which has a description of size less than* p.

If we consider Turing machines then one might be tempted to assume that the uninstantiability result gets amplified to full uninstantiability since Turing machines can handle arbitrary length inputs and hence an adversary can input the encoding of any (efficient) hash function. The crux of using Turing machines is that the encryption operation is no longer strictly a polynomial time algorithm. Consider an adversary that sends an encoding of the following program to its encryption oracle:

A brute-force attempt to disprove Goldbach's conjecture stating that any integer greater than 2 can be expressed as the sum of two primes.

GOLDBACH()

for $i = 4, 6, 8, 10, \ldots$ **do**
  if $\nexists$ primes p, q such that $p + q = i$ **then**
    return 0
return 1

Goldbach's conjecture goes back to a correspondence between Goldbach and Euler in 1742 [Gol42] and states that every integer greater than 2 can be written as the sum of two primes.[9] It is one of the oldest unproven number theoretical conjectures and part of Hilbert's 23 problems [Hil01]. For the GOLDBACH program above we negated the statement and formulated a brute-force search for an even integer that cannot be expressed by the sum of two primes. It is easily seen that the program only halts in case Goldbach's conjecture is wrong and if, as is generally believed, Goldbach's conjecture is correct then the program will run forever. Consequently, since our adapted symmetric encryption scheme now simply evaluates any Turing machine that it gets as input and the description of a Turing machine is independent of its runtime it follows that our scheme is no longer an efficient (i.e., polynomial time) scheme.

We can easily rectify the runtime by integrating an upper bound and only run the submitted program for a specific number of steps. This, however, again weakens the uninstantiability result that we obtain as then an instantiation with hash functions which need to run longer than the fixed bound might lead to secure instantiations. In order to overcome the runtime issue while still allowing

[9] In his letter to Euler [Gol42] Goldbach conjectured that every integer greater than 2 can be written as the sum of three primes, where he thought of 1 as a prime number [Dic05]. This conjecture, also called the *ternary Goldbach conjecture* is implied by the above form which is usually referred to as the *strong Goldbach conjecture*.

any efficient hash function to be ruled out, CGH make use of Micali's computationally sound proofs (CS-proofs) [Mic00]. In a CS-proof a prover tries to convince a verifier of a statement of the form *on input x program M outputs y within at most t steps*. Furthermore, it is required that the time it takes to verify a proof is much shorter than the time to generate a proof. In fact, while the time to generate a proof is polynomially related to the running time of machine M on input x, the running time of the verifier is only poly-logarithmic in parameter t. Micali shows that in the random oracle model non-interactive CS-proofs exist without further assumptions [Mic00] which is essentially the setting which is needed by CGH. The underlying idea is that an adversary instead of sending an encoding of hash function H can compute a CS-proof to the effect that H on some input x outputs value $\mathsf{RO}(x)$. As this puts the bulk of the computational strain on the adversary (the prover) the scheme remains efficient as it now only has to verify the proof. We here omit the details and instead present an alternative to the use of CS-proofs.

An alternative to CS-proofs. To allow arbitrarily long inputs while ensuring that the scheme remains efficient we again switch computational models, i.e., we again consider circuits but only within procedure check. For this, we consider a check procedure (implemented as a Turing machine) which interprets P as a circuit and for evaluation of P implements the universal circuit.

In contrast to Turing machines, the size of a circuit $|C|$ is directly related to its running time and hence the universal circuit for a circuit of size $s(\lambda)$ is of size $\mathcal{O}(s(\lambda)^c)$ for some constant $c \in \mathbb{N}$. There is one technicality that we need to take into account which is the encoding of circuits. One can easily think of encodings that allow polynomial sized encodings to describe super-polynomial size circuits: one such example would be to consider a circuit encoding that allows to describe the circuit via a Turing machine which constructs the circuit. If we, however, fix an encoding scheme such that the encoding $\langle C \rangle$ of a circuit C is polynomially related to the size $|C|$ of C (that is, $\exists c : |\langle C \rangle| \in \mathcal{O}(|C|^c)$) then a Turing machine M which simulates the universal circuit UC and applies it to its input is efficient (i.e., runs in polynomial time). As circuits can be represented as graphs we could, for example, choose to encode a circuit by an adjacency matrix in combination with a list denoting for each vertex the type of gate or whether it is an input or output node.

If we now consider that procedure check is implemented as a Turing machine which simulates the universal circuit to compute $\mathsf{UC}(\langle C \rangle, \mathsf{k}, x)$ where $(\mathsf{k}, \langle C \rangle)$ is the input to check then we have that check runs in polynomial time $\mathcal{O}(\mathsf{poly}(|\langle C \rangle| + \lambda))$ and, thus, also $\mathsf{SE.Enc}$ runs in polynomial time. As, furthermore, any efficient Turing machine can be efficiently converted into a polynomial size circuit we have that an adversary can efficiently construct a polynomial sized circuit description for any (efficient) hash function thus showing that the scheme cannot be instantiated with any (efficient) hash function.

Jumping ahead, in this thesis, we develop an extension to the techniques of CGH allowing us to present uninstantiability results for a large number of random oracle constructions, and in particular, for random oracle transformations—including the widely used Fujisaki–Okamoto transform [FO99] dating back to 1999—which fall outside the scope of the CGH attacks.

Summing up, CGH show the following (note that CGH showed their result for public-key encryption schemes and signature schemes).

Theorem 3.2 (informal [CGH98])**.** *There exist public-key encryption and signature schemes that are secure in the random oracle model but uninstantiable, that is, the standard model instantiation for any efficient hash function is insecure.*

Remark on message lengths. We note that our above scheme, as well as the schemes presented by CGH require (arbitrary) long inputs. CGH, in a later work, showed how to extend their techniques yielding an uninstantiability result for signature schemes with bounded messages [CGH03]. Finally, in a very recent work Green et al. [GKMZ14] show how to use obfuscation techniques to obtain (weak) uninstantiability results for any public-key primitive (including bit-encryption, i.e., encryption schemes that encrypt a single bit message). In contrast to CGH, their results are conditioned on the existence of *indistinguishability obfuscation* (a primitive that we will encounter throughout this thesis) and "weak" refers to the fact that for any polynomial p they construct schemes which cannot be instantiated by a hash function of size less than p but which might be instantiable by larger functions.

3.4.4 Random-Oracle Uninstantiability Results

Following CGH, a number of works further studied uninstantiability problems associated with random oracles. As already mentioned, Canetti, Goldreich, and Halevi [CGH03] extend their result in a follow-up work to signature schemes which only support short messages. Bellare, Boldyreva, and Palacio [BBP04] show that no instantiation of the hashed ElGamal key-encapsulation mechanism composes well with symmetric schemes, even though it enjoys this property in the ROM. Goldwasser and Kalai [GK03] study the Fiat–Shamir heuristic and establish uninstantiability results for it. Nielsen [Nie02] gives an uninstantiable cryptographic task, namely that of non-interactive, non-committing encryption, which although achievable in the ROM, is infeasible in the standard model. Finally, CGH-type uninstantiability has been adapted to also other idealized models such as the ideal-cipher model [Bla06] and the generic-group model [Den02].

3.5 The Random Oracle Methodology – A Controversy

The random oracle model has had a huge impact on practical cryptography. Signature schemes in practice—including the widely used RSA-PSS signature scheme [RSA78, BR96], the digital signature algorithm (DSA) [FIPS 186-4] and ElGamal signatures [ElG85]—usually follow the hash-then-sign paradigm where a message is first hashed before it is signed. While ensuring that arbitrary length messages can be signed the hash function plays a crucial role in the schemes' proofs and all before mentioned schemes only have proofs in the random oracle model [BR93, BR96, PS96].[10] Similarly, when we consider encryption we find that the standardized and widely used RSA-OAEP public-key encryption algorithm [BR95, FOPS04] only has a security proof in the random oracle model. A similar picture can be found all over cryptography, making "Random Oracles are Practical" one of the most influential papers in the history of cryptography.

The random oracle methodology has several very good arguments supporting it, most notably, it simply seems to work. Schemes designed using random oracles are often very efficient and, so far, no "natural" random oracle scheme (i.e., schemes that are not designed with the sole purpose of bringing to light random oracle inconsistencies) has been attacked due to the use of the random oracle methodology. Koblitz and Menezes, advocates for the use of the random oracle methodology

[10] We note that DSA signatures have no known security proof, neither in the standard model nor in the random oracle model.

in practice, recently published their "Twenty-Year Retrospective" [KM15] arguing that "there is no evidence that the need for the random oracle assumption in a proof indicates the presence of a real-world security weakness". They show that several schemes that were specifically designed to avoid the usage of random oracles while allowing for a proof of security in the standard model—Google Scholar alone lists 343 papers that have "without random oracles" in their title and more than 5000 where the exact phrase appears somewhere in the text, as of May 3, 2015—have, in fact, worse security properties than their random oracle counterparts. The latter of course assumes that the security proven for the random oracle schemes similarly applies also to the instantiated scheme. Yet, not just from a theoretical standpoint one may not feel at ease when trusting a heuristic for obtaining security—the random oracle methodology is essentially exactly that, a heuristic that we do not understand well. If schemes designed following the random oracle methodology are secure shouldn't we then be able to find and understand the core of the underlying hardness (assumptions)?

The random oracle debate that was started with the seminal paper of CGH is now more than 15 years old and researchers have expressed varying degrees of confidence in random oracle schemes ranging from Koblitz and Menezes who state that

> *"it is reasonable to think of a well-constructed real-world hash function as a deterministic function that is essentially indistinguishable from a random function"* [KM06]

and who suggest that a proof in the random oracle model can be good enough evidence to use a scheme in practice [KM04, KM06, KM15]. On the other end of the spectrum, Oded Goldreich states in the concluding remarks of [CGH98]:

> *"The bottom-line: It should be clear that the Random Oracle Methodology is not sound; that is, the mere fact that a scheme is secure in the Random Oracle Model cannot be taken as evidence (or indication) to the security of (possible) implementations of this scheme. Does this mean that the Random Oracle Model is useless? Not necessarily: it may be useful as a test-bed (or as a sanity check). Indeed, if the scheme does not perform well on the test-bed (resp., fails the sanity check) then it should be dumped. But one should not draw wrong conclusions from the mere fact that a scheme performs well on the test-bed (resp., passes the sanity check). In summary, the Random Oracle Methodology is actually a method for ruling out some insecure designs, but this method is not "complete" (i.e., it may fail to rule out insecure designs)."* [CGH98]

Goldreich also gives a compelling argument why random oracle schemes used in practice have not been subject to attacks based on the usage of the random oracle:

> *"good suggestions should be expected to pass a sanity check."* [CGH98]

The view of the random oracle model I currently hold while writing this thesis is more on a middle ground between the extremes. As random oracle schemes seem to work well we should certainly not try to abolish the random oracle from schemes and proofs, we should rather put effort into understanding what makes them work. Leonid Reyzin states in his lecture notes [Rey03]:

> *"I personally view [the Random Oracle Model] as a way to acknowledge our failures: there are a lot of constructions that seem secure on some intuitive level, but we can't prove them secure in the standard model. So (hopefully until we have a realy [sic] proof of security) we prove the [sic] secure in this funny fake model."* [Rey03]

This thesis is one attempt to better understand this *funny fake model* by, both bringing to light inconsistencies as well as providing standard model candidate constructions for security properties which, before this thesis, were only achievable in the random oracle model. Some such random oracle schemes we discuss in the next chapter.

Chapter 4

Random Oracles are Practical

> "Goals which are possible but impractical in the standard setting become practical in the random oracle setting."
>
> Mihir Bellare and Phillip Rogaway, [BR93]

Summary. In this chapter we present a selection of several primitives for which either all known constructions rely on random oracles, or which have very simple and elegant constructions in the random oracle model. For each of the primitives we discuss random oracle constructions, as well as, standard model counterparts. At the end of each section we briefly outline the results that we will show for this primitive in later parts of this thesis. We note that the results foreshadowed here are not complete and should only be understood as corollaries of results presented in later chapters.

Chapter content

4.1 Introduction

While we introduced the random oracle methodology in the previous chapter and saw some toy examples of random oracle schemes (a pseudorandom function and a symmetric-key encryption scheme) we present several additional cryptographic primitives in this chapter. The notions that we discuss span all over cryptography ranging from simple building blocks such as hardcore functions to more complex ones such as deterministic public-key encryption schemes or code obfuscation for point functions. What all these notions have in common is that the simplest and most efficient—and sometimes even all—constructions, have only been proved secure relative to random oracles. Furthermore, for some of the primitives intriguingly simple random oracle transformations exist which lift weakly secure primitives (in the standard model) into strongly secure ones in the random oracle model. For example, a prominent transformation to construct deterministic public-key encryption, called *Encrypt-with-Hash*, transforms "standard" randomized public-key encryption schemes (for

example, schemes that are IND-CPA secure in the standard model) into deterministic public-key encryption schemes which are provably secure in the random oracle model.

In each of the following sections, we introduce one primitive and discuss existing random oracle constructions as well as standard model constructions, if such exist. At the end of each section, we briefly discuss the new results we present later in the thesis. For example, for various of the here discussed notions we present the first standard model candidate construction at a later point (e.g., for universal hardcore functions or for deterministic public-key encryption). To jump ahead, all our candidate constructions are based on a notion of code obfuscation called *indistinguishability obfuscation*. We give a very brief and self-contained introduction into code obfuscation on page 47 and introduce and discuss it in detail in Chapters 5 and 6.

Besides giving positive results, i.e., candidate constructions for various cryptographic primitives, we also present negative results for many of the random oracle transformation introduced in this chapter. For example, we show that the above mentioned Encrypt-with-Hash transformation may fail assuming indistinguishability obfuscation exists. In more detail, if indistinguishability obfuscation exists then there are secure schemes which when transformed with the Encrypt-with-Hash transformation yield insecure schemes once the random oracle is instantiated with any standard model hash function.

4.2 Correlated-Input Secure Hash Functions

A one-way function f gives the security guarantee that given $\mathsf{f}(m)$ for a uniformly random value m an efficient adversary cannot fully recover a preimage $m' \in \mathsf{f}^{-1}(\mathsf{f}(m))$. While intuitively one-way functions hide some part of their input, the security guarantee given is rather weak. For example, a one-way function may leak large portions of m as long as some part remains hard to recover. Furthermore, if m is not drawn from the uniform distribution, i.e., some information about m is known, then also $\mathsf{f}(m)$ may not provide any security. Finally, one-wayness is defined as a single instance notion, that is, in the security game the adversary sees exactly one image $\mathsf{f}(m)$. Although the security extends to adversaries seeing multiple (polynomially many) images $\mathsf{f}(m_1), \ldots, \mathsf{f}(m_q)$ and having to invert at least one of them[1], as long as the values m_1 to m_q are chosen independently.[2] If, on the other hand, an adversary is to see images $\mathsf{f}(m), \mathsf{f}(m+1), \mathsf{f}(m+2), \ldots$, then again, one-wayness is not sufficient to give any security guarantees to be able to argue that an adversary cannot recover m. In fact, "fraudulent" one-way functions which break down completely if an adversary sees $\mathsf{f}(m)$ and $\mathsf{f}(m+1)$ can easily be constructed as shown in Example 4.1.

Correlated-input secure hash functions (CIH) can be regarded as a strengthening of the one-wayness notion. The notion of CIHs was introduced by Goyal, O'Neill, and Rao (GOR; [GOR11]) in three different flavors: *One-wayness* under correlated inputs, *unpredictability* under correlated inputs, and *pseudorandomness* under correlated inputs. These notions describe a hierarchy of security notions with *pseudorandomness* being the strongest notion, when we consider the natural setting of CIHs with a super-logarithmic output length. Intuitively, CIH under one-wayness captures the scenario above saying that an adversary that sees images under f of correlated inputs $m_1, \ldots, m_q$

[1] Consider a reduction, that given a single value $\mathsf{f}(m)$, simply chooses $q-1$ values at random, computes their images under f, and forwards them, including its own challenge to the multi-instance adversary. Then the reduction breaks the one-wayness of f with advantage $1/q$ times the advantage of the multi-instance adversary.

[2] Intuitively the reduction to a multi-instance adversary does not work any longer as an adversary without knowing m cannot produce the correct distribution of inputs.

Code Obfuscation. Most of our results presented in this thesis are related to code obfuscation which we discuss in detail in Chapters 5 and 6. Code obfuscation is a technique to transform code (e.g., the description of a circuit) into something that can still be executed and which implements the same functionality but *hides the workings* of the code. The idea of using obfuscation for cryptographic purposes goes back to Diffie and Hellman who, in their seminal paper on public-key encryption, write

> "*Essentially what is required is a one-way compiler: one which takes an easily understood program written in a high level language and translates it into an incomprehensible program in some machine language.*" [DH76]

A formal study of obfuscation in cryptography, started by Hada [Had00] in 2000 in the context of zero-knowledge, found its climax with the seminal work of Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI+01, BGI+12] who gave a reasonable definition of what it means to obfuscate a program, termed *virtual black-box obfuscation* (VBB), and showed that it cannot be achieved, in general. Intuitively, virtual black-box obfuscation asks that any (efficient) algorithm which takes as input the code of a program can be simulated by an (efficient) algorithm which only has black-box access to the functionality of the code. It can be shown that VBB obfuscation has many cryptographic applications, for example, it can be used to turn symmetric encryption schemes into public-key encryption schemes as envisioned by Diffie and Hellman. The fact that it does not exist in general, however, tells us that there are programs which cannot be obfuscated (according to VBB) and, in particular, we cannot hope to find a universal obfuscator which "works for any program".
Indistinguishability Obfuscation (iO) (the focus in this work) was proposed by Barak et al. [BGI+01, BGI+12] as a possible weakening of VBB. Until today we do not know of any results challenging the possibility that general-purpose indistinguishability obfuscation can exist. Indeed, in the last years several candidate constructions of indistinguishability obfuscator for all circuits in P/poly have been presented building confidence that general code obfuscation (code obfuscation for all programs) may, indeed, exist [GGH+13b, BR14, AGIS14, GLSW14, Zim15]. However, iO gives a much weaker security guarantee than VBB obfuscation: intuitively, indistinguishability obfuscation asks that if two programs P_1 and P_2 compute the same function and are of the same length, that is, for all x we have $P_1(x) = P_2(x)$ (and $|P_1| = |P_2|$), then their obfuscations should be computationally indistinguishable, i.e.,

$$\mathsf{iO}(P_1) \approx_c \mathsf{iO}(P_2).$$

Indistinguishability obfuscation may appear too weak to be useful at first glance—this feeling was shared by one of its inventors, Oded Goldreich, who stated

> "*I must admit that I was very skeptic of the possible applicability of the notion of indistinguishable obfuscation.*" [Gol13]

As it turns out, indistinguishability obfuscation has countless applications and we note that all constructions presented in this thesis crucially depend on iO.

cannot recover any of the preimages. The strongest notion defined by GOR instead requires that an adversary cannot tell apart the distributions

$$(\mathsf{f}(m_1), \ldots, \mathsf{f}(m_q)) \approx_c (r_1, \ldots, r_q)$$

where m_1 to m_q may be correlated and r_i are uniformly random values.

Example 4.1: A one-way function that is not correlation-input secure

Let $\mathsf{f} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a one-way function taking λ-bit inputs. We construct a one-way function $g : \{0,1\}^{2\lambda} \to \{0,1\}^{2\lambda}$ as

$$\mathsf{g}(x_\ell \| x_r) := \begin{cases} x_\ell \| f(x_r) & \text{, if } x_r \text{ is even, i.e., } \mathsf{lsb}(x) = 0 \\ \mathsf{f}(x_\ell) \| x_r & \text{, otherwise} \end{cases}$$

where x_ℓ denotes the first λ input bits and x_r denotes the second λ input bits. It is easily seen that g is one-way if f is. However, for an even $x \in \{0,1\}^{2n}$ we have that

$$(x+1) = \mathsf{g}(x)[1..\lambda] \parallel \mathsf{g}(x+1)[\lambda+1..2\lambda]$$

and, thus, an adversary seeing $\mathsf{g}(x)$ and $\mathsf{g}(x+1)$ can easily reconstruct x for any choice of $x \in \{0,1\}^{2\lambda}$.

So far, we have not yet described how correlated inputs are chosen. In order to get a security definition which is as general and as strong as possible we let the adversary choose the inputs. That is, we consider an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ which consists of two stages where the first stage $\mathcal{A}_1$ chooses inputs $m_1, \ldots, m_q$ and the second stage has to distinguish between either seeing the corresponding images $(\mathsf{f}(m_1), \ldots, \mathsf{f}(m_q))$ or uniformly random values $(r_1, \ldots, r_q)$.[3] In order to exclude trivial attacks we need to restrict the distribution as implemented by $\mathcal{A}_1$. If any of the outputs of $\mathcal{A}_1$ are *predictable* then an adversary can trivially win by simply guessing an output m_i of $\mathcal{A}_1$, recomputing $\mathsf{f}(m_i)$ and comparing it to the given values. To exclude such pathological adversaries we only consider admissible adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We call an adversary admissible, if adversary $\mathcal{A}_1$ on input the security parameter 1^λ outputs a vector $\mathbf{m}$ of distinct values and length $|\mathbf{m}| = \nu(\lambda)$, where ν is a polynomial depending on $\mathcal{A}_1$. Furthermore, we require that the guessing probability of each entry is negligible, that is, the min-entropy of $\mathbf{m}[i]$ for all $i = 1, \ldots, \nu(\lambda)$ must be at least super-logarithmic in the security parameter. This yields the following definition and we present the full security game CIH in Figure 4.1.

Definition 4.1 (Correlated-input secure hashing (CIH))**.** *We say that a function* H *is* CIH-secure *if the advantage of any* CIH*-admissible* PPT *adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *defined as*

$$\mathsf{Adv}^{\mathrm{cih}}_{\mathsf{H},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{CIH}^{\mathcal{A}}_{\mathsf{H}}(\lambda) = 1\right] - 1$$

is negligible where game CIH *is defined in Figure 4.1. An adversary* $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ *is called* CIH-admissible *if* $\mathcal{A}_1$ *implements a statistically unpredictable distribution, that is, on input the security parameter adversary* $\mathcal{A}_1$ *outputs a vector* $\mathbf{m}$ *of distinct values and of length* $|\mathbf{m}| = \nu(\lambda)$ *where* ν *is a polynomial depending on* $\mathcal{A}_1$ *and where for all* $i = 1, \ldots, |\mathbf{m}|$ *and all* $x \in \mathsf{H.il}(\lambda)$ *it holds that*

$$\Pr\left[x = \mathbf{m}[i] : \mathbf{m} \leftarrow\!\!\$\, \mathcal{A}_1(1^\lambda)\right] \leq \mathsf{negl}(\lambda)$$

[3] Recall that as defined earlier (cf. page 22) adversaries that are split over several stages and that do not freely share state constitute a two-stage game and hence, CIH, is a two-stage security notion.

$\underline{\mathrm{CIH}^{\mathcal{A}}_{\mathsf{H}}(\lambda)}$

$b \leftarrow\!\!\$\, \{0,1\}$
$\mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$
$\mathbf{m} \leftarrow\!\!\$\, \mathcal{A}_1(1^\lambda)$
for $i = 1, \ldots, |\mathbf{m}|$ **do**
  $\mathbf{h}_0[i] \leftarrow\!\!\$\, \{0,1\}^{\mathsf{H.ol}(\lambda)}$
  $\mathbf{h}_1[i] \leftarrow \mathsf{H.Eval}(\mathsf{hk}, \mathbf{m}[i])$
$b' \leftarrow\!\!\$\, \mathcal{A}_2(1^\lambda, \mathsf{hk}, \mathbf{h}_b)$
return $(b = b')$

Figure 4.1: The security game for correlated-input secure hash functions.

An equivalent way of formalizing the unpredictability is via min-entropy. That is, we require that for all $i \in \mathbb{N}$

$$\mathrm{H}_\infty(\mathbf{m}[i]) \in \omega(\log(\lambda)),$$

where $\mathbf{m}[i]$ denotes the random variable that runs $\mathcal{A}_1(\lambda)$ and outputs the i-th message component. This notion of unpredictability is also sometimes referred to as *statistical* unpredictability. Another equivalent, and more algorithmic way of defining unpredictability is to ask that no unbounded algorithm P, called predictor, can predict any of the outputs of $\mathcal{A}_1$ with noticeable probability. That is, for any (potentially unbounded) predictor P there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr\left[\left(\begin{array}{l}\mathbf{m} \leftarrow\!\!\$\, \mathcal{A}_1(1^\lambda)\\ \tau \leftarrow \mathsf{P}(1^\lambda)\\ \textbf{if } \exists i : \mathbf{m}[i] = \tau \textbf{ then}\\ \quad \textbf{return } 1\\ \textbf{return } 0\end{array}\right) = 1\right] \leq \mathsf{negl}(\lambda)$$

where the pseudocode represents a binary random variable and the randomness is over the random coins of $\mathcal{A}_1$ (note that we can assume the predictor to be deterministic as it is unbounded and can thus compute the best random coins).

4.2.1 CIH in the Random Oracle Model

In the random oracle model the adversary (i.e., $\mathcal{A}_1$ and $\mathcal{A}_2$) as well as the construction can access the random oracle. In order to get a meaningful security notion we need to strengthen the restrictions on the first adversarial stage to hold independent of the random oracle. When viewing $\mathbf{m}[i]$ as the random variable which evaluates $\mathcal{A}_1^{\mathsf{RO}}(1^\lambda)$ then the min-entropy requirement now becomes

$$\mathrm{H}_\infty(\mathbf{m}[i] \mid \langle \mathsf{RO} \rangle\ ) \in \omega(\log(\lambda)).$$

In other words we require that for all $i \in \mathbb{N}$ and each choice of RO the probability

$$\Pr\left[x = \mathbf{m}[i] : \mathbf{m} \leftarrow\!\!\$\, \mathcal{A}_1^{\mathsf{RO}}(1^\lambda)\right]$$

is negligible; that is, the probability is *not* over the choice of random oracle but only over the random coins of $\mathcal{A}_1$.

Correlation-input secure hash functions trivially exist in the random oracle model: the random oracle itself yields a correlation-input secure function. To see that this is the case, note that the min-entropy of a random variable X does not change given $\mathsf{RO}(X)$ since $\mathsf{RO}(X)$ is a uniformly random value (over the choice of the random oracle) revealing no information about X, i.e.,

$$\mathrm{H}_\infty(X) = \mathrm{H}_\infty(X \mid \mathsf{RO}(X)).$$

Following [ADW09] we can rewrite the above and consider an unbounded adversary P that is able to make up to p many queries to the random oracle (for some bound p):

$$\mathrm{H}_\infty\left(X \;\middle|\; \mathsf{P}^{\mathsf{RO}}(\mathsf{RO}(X))\right) := -\log \Pr\left[\mathsf{P}^{\mathsf{RO}}(\mathsf{RO}(X)) = X\right]$$

Intuitively, with each oracle query P can discard one value m by checking that $\mathsf{RO}(m) \neq \mathsf{RO}(X)$ and, thus, the min-entropy slightly decreases. This means that if X has super-logarithmic min-entropy and predictor P can make at most polynomially many queries to RO then X has still super-logarithmic min-entropy even conditioned on the output of P.

It is worth mentioning that we can also use the random oracle to construct a keyed correlation-secure hash function by reusing our construction of a pseudorandom function from the previous chapter

$\underline{\mathsf{H}^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)}$	$\underline{\mathsf{H}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{fk}, m)}$
$\mathsf{fk} \leftarrow\!\!\$\ \{0,1\}^\lambda$	$y \leftarrow \mathsf{RO}(\mathsf{fk} \| m)$
return fk	**return** y

or simply by switching to the keyed random oracle setting (see Section 3.3.5).

4.2.2 Barriers in Cryptography

In Section 2.3.3, we commented on the importance of distinguishing single-stage security notions (i.e., notions defined by a single-stage game) from multi-stage security notions. Correlated input secure hashing is a two-stage notion since the adversarial procedures $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot freely share state. Wichs shows that for CIH there exists an inefficient attack that can be efficiently simulated [Wic13] which means, in essence, that we cannot hope to find a security reduction for a CIH construction down to any standard (single-stage) cryptographic assumption such as factoring, discrete log, DDH, LWE, etc. Indeed, no standard model CIH-secure functions have been proposed that achieve full CIH-security. We will discuss existing standard model constructions (and how they relate to Wichs' result) in the following section and next present the intuition behind Wichs' result.

Consider a single-stage security notion such as one-wayness and consider a (black-box) security reduction $\mathcal{R}$ for some primitive achieving this notion, that is, a reduction which breaks the underlying difficult problem, say it solves the decisional Diffie-Hellman (DDH) problem with non-negligible probability, when it is given black-box access to any adversary that wins in the security game with non-negligible probability. Now, consider an inefficient adversary $\mathcal{A}$ that breaks the security game, then, not surprisingly $\mathcal{R}^{\mathcal{A}}$ solves the difficult problem too. This is not too surprising because, given enough time, an inefficient algorithm can easily solve DDH. Now, the crux is that if we can simulate

$\mathcal{A}$ efficiently, i.e., if there exists a simulator Sim such that for any efficient distinguisher D which only has black-box access to either $\mathcal{A}$ or Sim, we have that

$$\left|\Pr\left[\mathsf{D}^{\mathcal{A}}(1^\lambda)=1\right]-\Pr\left[\mathsf{D}^{\mathsf{Sim}(1^\lambda)}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\,,$$

then also $\mathcal{R}^{\mathsf{Sim}}$ (which is now efficient, since both $\mathcal{R}$ and Sim are efficient) must be able to break the hard problem. Basically, this means that our simulator is as good as our adversary and, hence, our scheme was not secure to begin with (that is such a reduction $\mathcal{R}$ cannot exist). In the single-stage setting, finding an efficiently simulatable (inefficient) attack is as difficult as finding an attack in the first place. When considering multi-stage security notions this may, however, not be true.

Let us consider the multi-stage correlated-input secure hash game. Here, we consider two adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$ which do not freely share state: we require that $\mathcal{A}_1$ implements an unpredictable distribution and that $\mathcal{A}_2$ should not learn the random coins of $\mathcal{A}_1$. A simulator Sim for a successful (inefficient) adversary $(\mathcal{A}_1,\mathcal{A}_2)$ does not need to honor this structural requirement, as it simulates both stages "together". All that we asked of the simulator is that for any efficient distinguisher it holds that:

$$\left|\Pr\left[\mathsf{D}^{\mathcal{A}_1,\mathcal{A}_2}(1^\lambda)=1\right]-\Pr\left[\mathsf{D}^{\mathsf{Sim}(1^\lambda)}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\,.$$

Thus, the simulation itself does not yield a valid attack. However, if there was a black-box security reduction $\mathcal{R}^{\mathcal{A}_1,\mathcal{A}_2}$ to a single-stage assumption (such as factoring or DDH) then also $\mathcal{R}^{\mathsf{Sim}(1^\lambda)}(1^\lambda)$ would need to break that assumption, and hence, if we believe the assumption such a reduction $\mathcal{R}$ cannot exist.[4]

In a nutshell, if for a multi-stage game we can show that a simulatable attack exists (where the simulator does not need to respect the requirements of the game) then we cannot hope to reduce the security to a single-stage assumption.

4.2.3 CIH in the Standard Model

So far, constructions in the standard model only achieve restricted versions of correlation security, in particular, they have to find a way around Wichs' impossibility result (see above). The usual workaround is to restrict the form of allowed correlation or to put (very) strong entropy requirements on the first stage. As Wichs' simulated attack [Wic13] requires both, a high correlation and low-entropy these allow to bypass his result at the cost of not achieving full CIH-security.

Goyal, O'Neill, and Rao [GOR11], who also introduce the CIH notions, construct restricted CIHs that are secure under polynomially related inputs. Freeman et al. [FGK+13] as well as Rosen and Segev [RS10] use a fresh key for every input rather than a single key for all inputs.

A notion related to CIH as defined above is *statistically* secure q-query CIH-security. Here, the key size may grow with the number of queries and the same hash key is used for each query. In contrast to our security notion it is required that the output is statistically close to random given the hash key, instead of only computationally close. For this note that in Definition 4.1 we consider only efficient adversaries and in particular restrict $\mathcal{A}_2$ to be efficient. When considering statistical security, this notion is only achievable for distributions that come with a notable amount of entropy, that is, the q preimages need to have entropy at least q times the output length. In turn, for the

[4] This technique is also often called a *meta-reduction* (or a reduction against reductions) [BV98].

notion of entropy that we consider, the entropy of the preimages does not need to grow with q and can also be less than the length of the output. In fact, it is sufficient if each preimage individually only has super-logarithmic min-entropy in the security parameter.

We note that statistically secure CIH only considers a substantially smaller class of distributions and as shown by Fuller, O'Neill, and Reyzin a q-wise independent hash function is already sufficient to obtain statistical CIH security for adversaries that see at most q-many queries [FOR12].

> **New Results.** Similarly to Fuller et al. [FOR12] we consider q-query CIH, however, in the computational setting. That is, we are in the setting of Definition 4.1 but allow construction H to depend on a polynomial q and require security only for adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ outputs message vectors $\mathbf{m}$ of length at most $|\mathbf{m}| \leq \mathsf{q}(\lambda)$. We note that for this notion of q-query CIH Wichs' impossibility result (see Section 4.2.2) applies and yet, we manage to work around it by reducing security to a two-stage assumption; namely point-function obfuscation. In addition, our construction depends on indistinguishability obfuscation.
>
> In summary, we give the first standard model candidate construction for q-query CIH via an intermediary primitive called UCE (see Chapters 9 and 11).

4.3 A Universal Hardcore Function with Polynomial Output

As discussed in the previous section, one-way functions may leak a significant amount of information about their input. The only requirement is that for a randomly chosen preimage the image hides some information about the preimage. Given that one-wayness only provides a very weak security guarantee an intriguing problem is how to bootstrap one-way functions in order to obtain stronger guarantees. To this end we may use so-called *hardcore functions* which are defined over the same preimage space as the corresponding one-way function [BM84]. The requirement for a hardcore function hc for a one-way function f is that the distributions

$$(\langle \mathsf{f} \rangle, \langle \mathsf{hc} \rangle, \mathsf{f}(x), \mathsf{hc}(x)) \qquad \text{and} \qquad (\langle \mathsf{f} \rangle, \langle \mathsf{hc} \rangle, \mathsf{f}(x), r)$$

are computationally indistinguishable where x is chosen uniformly at random from the preimage space of f and r is a uniformly random string with the same length as $\mathsf{hc}(x)$. More formally, we consider a possibly keyed one-way function f and a keyed hardcore function hc and require that no efficient adversary can win the HC game specified in Figure 4.2.

Definition 4.2 (Hardcore functions)**.** *Let f be a (possibly keyed) one-way function. We say that a deterministic function hc is* hardcore *for f if the advantage of any PPT adversary $\mathcal{A}$ in game $\mathrm{HC}^{\mathcal{A}}_{\mathsf{f},\mathsf{hc}}$ (given in Figure 4.2) is negligible, where we define the advantage of adversary $\mathcal{A}$ as*

$$\mathsf{Adv}^{\mathrm{hc}}_{\mathsf{f},\mathsf{hc},\mathcal{A}}(\lambda) := 2 \cdot \Pr\left[\mathrm{HC}^{\mathcal{A}}_{\mathsf{f},\mathsf{hc}}(\lambda) = 1\right] - 1 \ .$$

We say that a function hc is a universal hardcore function *if it is hardcore for every one-way function.*

Note that for a universal hardcore function uhc it is, indeed, necessary to be keyed in order to avoid that a one-way function can depend on it. If the latter is possible then for any function uhc it

$\mathrm{HC}^{\mathcal{A}}_{\mathsf{f},\mathsf{hc}}(\lambda)$

$b \leftarrow\!\$\ \{0,1\}$
$\mathsf{k} \leftarrow\!\$\ \mathsf{f.KGen}(1^\lambda);\ \mathsf{hk} \leftarrow\!\$\ \mathsf{hc.KGen}(1^\lambda)$
$x \leftarrow\!\$\ \{0,1\}^{\mathsf{f.il}(\lambda)}$
$y \leftarrow \mathsf{f.Eval}(\mathsf{k}, x)$
$r_0 \leftarrow\!\$\ \{0,1\}^{\mathsf{hc.ol}(\lambda)}$
$r_1 \leftarrow \mathsf{hc.Eval}(\mathsf{hk}, x)$
$b' \leftarrow\!\$\ \mathcal{A}(1^\lambda, \mathsf{k}, \mathsf{hk}, y, r_b)$
return $(b = b')$

Figure 4.2: We here give the security game for hardcore functions.

is easy to construct a pathological one-way function g such that uhc is not hardcore for g even on the uniform distribution.[5]

Remark. Hardcore functions are generally formalized with respect to the uniform distribution. We can strengthen the definition and consider any unpredictable adversarial distribution assuming that the one-way function in question is also one-way on this distribution.

4.3.1 Hardcore Functions in the Random Oracle Model

As a hardcore function for a one-way function f is a (deterministic) algorithm whose output on an unpredictable point x is indistinguishable from random even given $\mathsf{f}(x)$, it is easy to see that (keyed) random oracles are natural universal hardcore functions, i.e., hardcore functions for any one-way function. (If we do not allow the one-way function to depend on the random oracle it also suffices to consider unkeyed random oracles.) For an adversary, the distributions $(\mathsf{RO}(x), \mathsf{f}(x))$ and $(r, \mathsf{f}(x))$ where r is a uniformly random value are indistinguishable unless it queries RO on x. In Figure 4.3, we give an (informal) black-box reduction which turns any PPT adversary against the hardcoreness of the random oracle for some function f into an inverter. Assuming the function is one-way such an inverter cannot exist, thus, proving the claim.

4.3.2 Hardcore Functions in the Standard Model

Hardcore functions, introduced in 1984 by Blum and Micali [BM84], have been studied for over 30 years. The first breakthrough was due to Goldreich and Levin ([GL89]; GL) who showed in 1989 how to construct a hardcore predicate (a hardcore function which outputs a single bit) from any one-way function, more precisely, they gave a single predicate for every (slightly adapted) one-way function. Formally, GL showed that if f is a one-way function then we can construct a hardcore predicate for one-way function g defined as

$$\mathsf{g}(x, r) = \mathsf{f}(x) \| r \quad \text{with } |x| = |r|$$

[5] Consider the function $g(x) := \mathsf{f}(x) \| \mathsf{uhc}(x)$.

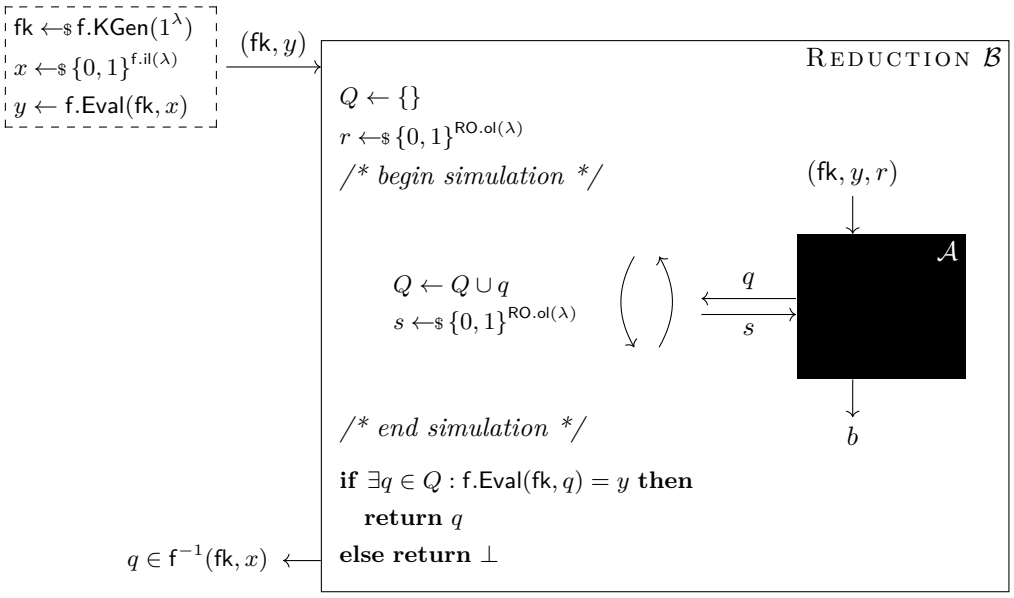


Figure 4.3: A security reduction showing that a random oracle is a universal hardcore function. Without loss of generality we assume that adversary $\mathcal{A}$ never repeats a query to its oracle. The reduction takes as input a key fk for function f and an image y for a uniformly random preimage x. It chooses a random value r and runs adversary $\mathcal{A}$ on (fk, y, r). It simulates the random oracle for $\mathcal{A}$ and answers any oracle query with a random value while keeping a list of all the queries. Finally, it checks if any of the queries are a valid preimage for $\mathsf{f.Eval}(\mathsf{fk}, x)$.

as the inner product of x and r

$$\mathsf{hc}(x, r) = \langle x, r \rangle .$$

Note that the GL predicate is thus not a universal hardcore predicate but that from any one-way function we can construct an adapted one-way function for which the GL predicate is hardcore.

Hardcore functions in general and the GL predicate in particular have numerous applications in cryptography; the GL paper to this day has been cited more than 950 times according to Google scholar. A prominent application of hardcore functions is the construction of pseudorandom generators from one-way permutations. If f is a one-way permutation, then $\mathsf{g}(x, r) := \mathsf{f}(x) \| \langle x, r \rangle$ yields a pseudorandom generator with a stretch of one bit. A second application, which we discuss in more detail in the following section, is the construction of public-key encryption schemes from arbitrary trapdoor functions. Here hardcore functions play a fundamental role.

It is worth mentioning that the GL construction can be extended to extract logarithmically many hardcore bits. The question whether we can extract arbitrarily many (i.e., polynomially many) was open until very recently. In 2013 Bellare, Stepanovs, and Tessaro (BST; [BST14]) showed how to construct a hardcore function with long output from any one-way function assuming so called differing-inputs obfuscation. In a second construction they show how to construct a hardcore function for any injective one-way function assuming the weaker notion of indistinguishability obfuscation. We note that differing-inputs obfuscation has recently been shown to be conditionally impossible by Garg et al. [GGHW14] under a special-purpose assumption on obfuscators for signature schemes. (We introduce code obfuscation in detail in Chapter 5 where we also discuss the conditional impossibility

$\underline{\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$

1 : $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
2 : $(\mathsf{st}, m_0, m_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda, \mathsf{pk})$
3 : $b \leftarrow\!\!\$\ \{0,1\}$
4 : $c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$
5 : $b' \leftarrow\!\!\$\ \mathcal{A}_2(1^\lambda, \mathsf{st}, c)$
6 : **return** $(b = b' \wedge |m_0| = |m_1|)$

Figure 4.4: The IND-CPA security notion for public-key encryption.

result of Garg et al.)

Besides the question of whether hardcore functions with long output (i.e. super-logarithmic output) exist for every one-way function, the existence of universal hardcore functions—a single hardcore function for every one-way function—and in particular the existence of universal hardcore functions with long output is still open. Concerning this, note that neither of the presented options accomplishes this goal as the GL predicate requires to first adapt the one-way function and the construction of BST depends on the one-way function.

> **New Results.** In this thesis we give the first standard model candidate construction of a universal hardcore function with a long output length, or more precisely, the output length of our construction can be an arbitrary polynomial. Our construction assumes the existence of indistinguishability obfuscation and strong forms of point obfuscation (see Chapters 9 and 11). We note that our construction is not only hardcore for the uniform distribution but for any unpredictable distribution.

4.3.3 Public-key Encryption from any Trapdoor Function

Hardcore functions are at the core of a large number of cryptographic constructions and overall had a huge impact on the theory of cryptography. One particularly interesting use-case is the construction of public-key encryption schemes from trapdoor functions.

Trapdoor functions. Trapdoor functions (TDF) and trapdoor permutations (TDP) can be regarded as public-key counterparts to one-way functions. A function family f is called trapdoor function, if its key generation algorithm $\mathsf{f.KGen}$ on input the security parameter outputs a key-pair $(\mathsf{sk}, \mathsf{fk})$ where fk denotes the public evaluation key such that given only key fk function $\mathsf{f.Eval}(\mathsf{fk}, \cdot)$ should be one-way. On the other hand, given the secret trapdoor key sk it should be easy to invert. As we usually consider only injective trapdoor functions we can require for correctness that for security parameters $\lambda \in \mathbb{N}$ and all $x \in \{0,1\}^{\mathsf{f.il}(\lambda)}$ we have that

$$\Pr\left[\mathsf{f.Inv}\left(\mathsf{sk}, \mathsf{f.Eval}(\mathsf{fk}, x)\right) = x\right] = 1$$

where the probability is over the key generation and Inv denotes the efficient (deterministic) inversion algorithm.

Public-key encryption vs. trapdoor functions. Although trapdoor functions have the feel of encryption to them, they give a much weaker security guarantee to what we would require from encryption. The reason is that at the heart of a trapdoor function is a one-way function which

requires only to be hard *on a small part* of its input and, furthermore, it only has to hide something if the input is chosen from the *uniform distribution*. For secure encryption, on the other hand we would like to have the guarantee that an encryption of a *chosen message* leaks nothing but the length of the message. A standard definition of secure public-key encryption is the indistinguishability under chosen plaintext (IND-CPA) notion which we formalize next.

Formally, we define a public-key encryption scheme $\mathsf{PKE} := (\mathsf{PKE.KGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ to consist of three PPT algorithms as follows. On input the security parameter, the randomized key-generation algorithm $\mathsf{PKE.KGen}(1^\lambda)$ generates a key pair $(\mathsf{sk}, \mathsf{pk})$. The randomized encryption algorithm $\mathsf{PKE.Enc}(\mathsf{pk}, m)$ takes as input a public key pk, a message m (and some random coins) and outputs a ciphertext c. The deterministic decryption algorithm $\mathsf{PKE.Dec}(\mathsf{sk}, c)$ takes as input a secret key sk and a ciphertext c and outputs a plaintext message m or a special symbol $\perp$. We denote the supported message length by $\mathsf{PKE.il}(\lambda)$ and the maximum length of random strings used to encrypt a $\mathsf{PKE.il}(\lambda)$-bit message by $\mathsf{PKE.rl}(\lambda)$. We say that scheme PKE is correct if for all $\lambda \in \mathbb{N}$, all $m \in \mathsf{PKE.il}(\lambda)$, all $(\mathsf{sk}, \mathsf{pk}) \in \mathrm{Supp}\big(\mathsf{PKE.KGen}(1^\lambda)\big)$ and all $c \in \mathrm{Supp}\big(\mathsf{Enc}(\mathsf{pk}, m)\big)$ we have that $\mathsf{PKE.Dec}(\mathsf{sk}, c) = m$. We say that PKE is IND-CPA secure if the advantage of any PPT adversary $\mathcal{A}$ in the IND-CPA game (given in Figure 4.4) is negligible:

$$\mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := 2 \cdot \Pr\Big[\text{IND-CPA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)\Big] - 1.$$

Public-key encryption from hardcore functions. Yao showed how to use a hardcore function for a trapdoor function in order to construct an IND-CPA secure public-key encryption scheme [Yao82b]. Let f be a trapdoor function and let hc be a hardcore function for f. Then, Yao's intriguingly simple construction is:

$\mathsf{PKE.KGen}(1^\lambda)$	$\mathsf{PKE.Enc}((\mathsf{fk}, \mathsf{hk}), m; r)$	$\mathsf{PKE.Dec}((\mathsf{sk}, \mathsf{hk}), (s, c))$
$(\mathsf{sk}, \mathsf{fk}) \leftarrow\!\!\$\ \mathsf{f.KGen}(1^\lambda)$	$s \leftarrow \mathsf{f.Eval}(\mathsf{fk}, r)$	$r \leftarrow \mathsf{f.Inv}(\mathsf{sk}, s)$
$(\mathsf{hk}) \leftarrow\!\!\$\ \mathsf{hc.KGen}(1^\lambda)$	$c \leftarrow m \oplus \mathsf{hc}(\mathsf{hk}, r)$	$m \leftarrow c \oplus \mathsf{hc}(\mathsf{hk}, r)$
return $((\mathsf{sk}, \mathsf{hk}), (\mathsf{fk}, \mathsf{hk}))$	**return** (s, c)	**return** m

Key generation simply runs the key generation algorithms of the trapdoor function and the hardcore function. The secret key is then composed of the inversion key sk for the trapdoor function and the evaluation key hk for the hardcore function. The public key consists of the evaluation keys of both the trapdoor function (fk) and the hardcore function (hk). To encrypt, randomness r is first *blinded* using the trapdoor function. Then, a pseudorandom string $\mathsf{hc}(\mathsf{hk}, r)$ is generated from the randomness using the hardcore function which is then used to encrypt message m by xoring it onto the message, i.e., $c \leftarrow m \oplus \mathsf{hc}(\mathsf{hk}, r)$. The ciphertext consists of the blinded randomness $s \leftarrow \mathsf{f.Eval}(\mathsf{fk}, r)$ and the encrypted message c. For decryption first the randomness is recovered using the inversion key for the trapdoor function which then allows decryption by recomputing $\mathsf{hc}(\mathsf{hk}, r)$ and xoring the result to c. We observe that the scheme is correct.

The security of the above construction is easily explained. Since hc is hardcore for f the distributions

$$(\mathsf{f.Eval}(\mathsf{fk}, r), \mathsf{hc}(\mathsf{fk}, r)) \qquad \text{and} \qquad (\mathsf{f.Eval}(\mathsf{fk}, r), s)$$

are computationally indistinguishable where r is uniformly distributed and s denotes a uniformly random value of the same length as $\mathsf{hc}(\mathsf{fk}, r)$. Hence, $\mathsf{hc}(\mathsf{fk}, r)$ is pseudorandom and the security of

the scheme reduces to the security of the one-time pad.

While the above scheme is elegant and simple, its efficiency depends on the number of output bits of the hardcore function. For example, instantiated with the GL hardcore predicate, we obtain a public-key encryption scheme that encrypts 1-bit messages.

The BR93 PKE scheme. We have already seen that random oracles are universal hardcore functions. In their seminal paper on random oracles, Bellare and Rogaway show that a random oracle can substitute the hardcore function in Yao's scheme [BR93] which yields a very efficient public-key encryption scheme. We refer to this random oracle PKE scheme as the BR93 public-key encryption scheme throughout this thesis.

> **New Results.** Given our construction of a universal hardcore function we can securely instantiate the BR93 PKE scheme.

4.4 CCA-Secure Public-Key Encryption

A standard security notion for public-key encryption schemes is *indistinguishability against chosen plaintext attacks* (IND-CPA) which considers adversaries that have full control over the choice of plaintexts. IND-CPA can be further strengthened by giving the adversary access to a decryption oracle and allowing it to decrypt arbitrary messages (except for the challenge message). This yields the notion of *indistinguishability against chosen ciphertext attacks* (IND-CCA). Chosen-ciphertext security is an important security notion as decryption oracles (or something close) can often be found in practice—a discussion on the necessity of resilience against chosen-ciphertext attacks (CCA) is given by Shoup in [Sho98].

Defining chosen-ciphertext security. A formal study of chosen-ciphertext security (IND-CCA) was started with the works of Naor and Yung [NY90] and Rackoff and Simon [RS92]. Chosen-ciphertext security security today comes in various flavors most notably IND-CCA1 and IND-CCA2. In IND-CCA1 the adversary is allowed decryption queries only before seeing the challenge ciphertext whereas in IND-CCA2 the adversary is allowed to continue querying the decryption oracle also after seeing the challenge with the only restriction that it is not allowed to query the challenge ciphertext.[6] We give a game-based formalization in Figure 4.5 where we require that no efficient adversary can win game IND-CCA with probability better than $\frac{1}{2}$.

Chosen ciphertext attacks. The popular ElGamal encryption scheme [ElG84] which is IND-CPA secure under the decisional Diffie–Hellman assumption (DDH) takes a public key

$$(\mathbb{G}, \mathsf{q}, g, h)$$

where $\mathbb{G}$ is a cyclic group of order q, g is a group generator, and h is set to $h \leftarrow g^x$ for a uniformly random value x. In order to encrypt a message $m \in \mathbb{G}$, a random value $y \leftarrow\!\!\$\, \mathbb{Z}_q$ is chosen to obtain

[6] We refer to [BDPR98] for a study of various flavors of IND-CCA public-key encryption and note that defining the condition that the adversary should not query the challenge ciphertext to its decryption oracle is surprisingly subtle and different formalizations may not be equivalent [BHK15].

$\underline{\text{IND-CCA}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)}$

$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
$(\mathsf{st}, m_0, m_1) \leftarrow\!\!\$\ \mathcal{A}_1^{\textsc{Dec}[\mathsf{sk},\perp](\cdot)}(1^\lambda, \mathsf{pk})$
$b \leftarrow\!\!\$\ \{0,1\}$
$c \leftarrow\!\!\$\ \mathsf{PKE.Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow\!\!\$\ \mathcal{A}_2^{\textsc{Dec}[\mathsf{sk},c](\cdot)}(1^\lambda, \mathsf{st}, c)$
return $(b = b' \wedge |m_0| = |m_1|)$

$\underline{\mathsf{Dec}[\mathsf{sk},\mathrm{c}](\mathrm{x})}$

if $c = x$ **then**
  return $\perp$
$m \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}, x)$
return m

Figure 4.5: A formalization of the IND-CCA security notion for public-key encryption. For IND-CCA1 security adversary $\mathcal{A}_2$ is not given access to the decryption oracle, whereas IND-CCA2 security allows also $\mathcal{A}_2$ to query $\textsc{Dec}$ (with the only restriction that the challenge ciphertext c may not be queried).

ciphertext

$$C := (Y, Z) := (g^y, h^y \cdot m).$$

Message m thus only goes into the "Z-part" of ciphertext C and it is easy to see that for any $m' \in \mathbb{G}$ the mauled ciphertext

$$C' := (Y, Z \cdot m') = (g^y, h^y \cdot m \cdot m')$$

is a valid ciphertext for message $m \cdot m'$. The consequence is that the ElGamal public-key encryption scheme is not secure against adversaries in the chosen ciphertext scenario.

The most prominent practical chosen-ciphertext attack is probably due to Bleichenbacher who showed how to attack the standardized RSA PKCS #1 v1.5.: Bleichenbacher shows how to decrypt a target ciphertext when given an oracle that tells if a ciphertext is valid (i.e., an oracle which decides whether or not a ciphertext can be successfully decrypted) [Ble98].[7] Such information is often easy to obtain, for example, a protocol might specify a specific error message in case a ciphertext cannot be decrypted, e.g., if it is of the wrong format.

4.4.1 CCA-secure Public-key Encryption in the Random Oracle Model

In the random oracle model various constructions of IND-CCA secure public-key encryption schemes are known and, already in their seminal paper on random oracles, Bellare and Rogaway gave a ROM construction of an IND-CCA secure PKE scheme from trapdoor functions [BR93]. A prominent construction, also due to Bellare and Rogaway, is the *Optimal Asymmetric Encryption Padding* (OAEP), a random oracle transformation which constructs IND-CPA secure public-key encryption from any trapdoor function and IND-CCA secure public-key encryption from trapdoor functions with a certain property (which, for example, is present in the RSA trapdoor function) [BR95, Sho01, Sho02, FOPS01, FOPS04]. We note that OAEP has also been standardized to be used with the RSA cryptosystem [RFC 3447].[8]

[7] RSA PKCS #1 v1.5. is standardized as RFCs 2313, 2437, and 3447 [RFC 2313, RFC 2437, RFC 3447].

[8] OAEP yields an efficient random oracle scheme which, in addition, to the evaluation of the trapdoor function needs only two hash function evaluations for an encryption or decryption. In general, RSA-OAEP should be preferred over RSA PKCS # v1.5. A simple rule of thumb which, regrettably, is still occasionally ignored by standardizing bodies [JSS12, DFF+14].

In the following we discuss a second random oracle transformation which was proposed in 1999 by Fujisaki and Okamoto.

CCA via Hybrid Encryption: The FO-Transformation

The Fujisaki–Okamoto (FO) transformation [FO99] is a ROM technique to convert "weak" public-key encryption schemes, e.g., those which are indistinguishable (or even one-way) against chosen-plaintext attacks, into "strong" ones which resist chosen-ciphertext attacks (i.e., are IND-CCA secure). In this transform a public-key encryption scheme PKE, a (deterministic) symmetric encryption scheme SE and two independent random oracles RO_1 and RO_2 are used. As explained in Section 3.3.3 two random oracles can be simulated from a single random oracle via domain separation.

Under the FO transform, a ciphertext for a message m is generated by picking a fresh random value σ which will be hashed and then used as key for the symmetric encryption scheme which in turn is used to encrypt the actual message m. The asymmetric scheme PKE is then used to encrypt value σ in a *checkable* way, meaning that the randomness used to encrypt can be derived from message m and value σ as hash of their concatenation via RO_1. This yields the following transformed encryption operation

$$\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(\mathsf{pk},m;\sigma) := (\mathsf{PKE}.\mathsf{Enc}(\mathsf{pk},\sigma;\mathsf{RO}_1(\sigma\|m)),\mathsf{SE}.\mathsf{Enc}(\mathsf{RO}_2(\sigma),m))\,,$$

with decryption being defined in the natural way: decrypt the public-key part to obtain the symmetric key $\mathsf{RO}_2\sigma$ to then recover message m.

4.4.2 CCA-secure Public-key Encryption in the Standard Model

While IND-CCA secure schemes in the standard model have been known for long time—Naor and Yung [NY90] show how to build IND-CCA secure public-key encryption schemes from trapdoor functions—achieving efficient IND-CCA secure public-key encryption in the standard model is an ongoing research effort. The first practical standard model scheme was presented in 1998 by Cramer and Shoup [CS98]. The Cramer–Shoup cryptosystem can be regarded as an extension of the ElGamal scheme and is also based on the DDH assumption.

> **New Results.** We show that if indistinguishability obfuscation exists, then the Fujisaki–Okamoto (FO) transformation is not sound (Chapter 8). That is, there is an IND-CPA public-key encryption scheme which, when transformed with the FO transformation (for any symmetric encryption scheme and any standard model hash function), yields a completely insecure scheme, more concretely, the resulting scheme is not even IND-CPA secure.

4.5 Deterministic Public-Key Encryption

Encryption schemes are usually randomized as otherwise security notions such as IND-CPA cannot be met. A consequence of using randomness is that randomized public-key encryption schemes need to produce ciphertexts which are longer than the message. A second scenario for which randomized encryption schemes might be disadvantageous are database lookups over encrypted data. For this note that existing database index structures are designed for plaintext values and it is not clear how

to allow for operations such as fast lookups without storing additional information if one only wants to store encrypted data. On the other hand, if the encryption operation was deterministic existing database index structures could simply be used. Third, as shown recently by Bellare, Paterson and Rogaway [BPR14] any randomized encryption scheme is susceptible to so-called *algorithm-substitution attacks* which consider scenarios where encryption algorithms are replaced by subverted algorithms.[9] These subverted algorithms produce ciphertexts which are indistinguishable from the actual encryption algorithm while allowing an adversarial party—the party that generated the subverted algorithm—to compromise the encryption scheme (e.g., information about the encrypted message is leaked, or in the case of symmetric schemes even the secret key). One way around such algorithm-substitution attacks are *deterministic* schemes and in the following we consider deterministic public-key encryption.

A formal study of deterministic public-key encryption was first initiated by Bellare et al. [BBO07]. The syntax and correctness of a deterministic public-key encryption scheme $\mathsf{D\text{-}PKE}$ is defined similarly to a randomized PKE scheme with the only difference that the encryption routine is deterministic. That is, a deterministic public-key encryption scheme $\mathsf{D\text{-}PKE} := (\mathsf{D\text{-}PKE.KGen}, \mathsf{D\text{-}PKE.Enc}, \mathsf{D\text{-}PKE.Dec})$ consists of a PPT algorithm $\mathsf{D\text{-}PKE.KGen}$ and two *deterministic* polynomial-time algorithms $\mathsf{D\text{-}PKE.Enc}$ and $\mathsf{D\text{-}PKE.Dec}$ as follows. On input the security parameter, the randomized key-generation algorithm $\mathsf{D\text{-}PKE.KGen}(1^\lambda)$ generates a key pair $(\mathsf{pk}, \mathsf{sk})$. The deterministic encryption algorithm $\mathsf{D\text{-}PKE.Enc}(m, \mathsf{pk})$ gets a message m and a public key pk and outputs a ciphertext c. The deterministic decryption algorithm $\mathsf{D\text{-}PKE.Dec}(c, \mathsf{sk})$ is given a ciphertext c and secret key sk and outputs a plaintext or a special symbol $\perp$. We denote the supported message-length by $\mathsf{D\text{-}PKE.il}(\lambda)$. Correctness of a deterministic PKE scheme is defined analogously to correctness of randomized schemes.

Strong security notions such as IND-CPA security can be attacked trivially once the public-key encryption scheme is deterministic. For this, consider an adversary that outputs two distinct messages m_0 and m_1 to obtain a ciphertext c. In order to distinguish whether c is an encryption of m_0 or m_1 the adversary can simply recompute an encryption of m_0 as it has access to the public key and compare it to c. As the scheme is correct and deterministic, the values match if, and only if, c is an encryption of m_0.

Bellare et al. [BBO07] study security notions for deterministic public-key encryption and introduce the PRIV security notion which captures semantic security in case there exists some uncertainty about the encrypted messages. In later works, Bellare et al. [BFOR08], and independently Boldyreva et al. [BFO08], introduce an indistinguishability style security notion, called IND, and show that it implies PRIV-security. In this thesis we use the IND notion of security (see Figure 4.6 left). Roughly speaking, an IND adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ consists of two stages that do not share state (i.e., it is a two-stage security notion). On input the security parameter, adversary $\mathcal{A}_1$ outputs a pair of message vectors $(\mathbf{m}_0, \mathbf{m}_1)$ that are of the same length, both having distinct components such that each cross-vector component pair has the same length, that is, $|\mathbf{m}_0[i]| = |\mathbf{m}_1[i]|$ for all i. Furthermore, we require that $\mathcal{A}_1$ is statistically unpredictable for each component, that is each component has super-logarithmic min-entropy. Formally, we capture this by requiring that for any $x \in \mathsf{D\text{-}PKE.il}(\lambda)$, any $b \in \{0,1\}$, and any $i \in \mathbb{N}$ the probability

$$\Pr\left[x = \mathbf{m}_b[i] : (\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\, \mathcal{A}_1(1^\lambda)\right]$$

[9] Extended algorithm-substitution attacks were recently presented by Bellare et al. [BJK15].

$\underline{\text{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{D\text{-}PKE}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{D\text{-}PKE.KGen}(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $\mathbf{c}[i] \leftarrow \mathsf{D\text{-}PKE.Enc}(\mathsf{pk}, \mathbf{m}_b[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{pk}, \mathbf{c})$
return $(b = b')$

$\underline{\$\text{-IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{D\text{-}PKE}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
$\mathbf{m} \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{D\text{-}PKE.KGen}(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $\mathbf{c}_0[i] \leftarrow \mathsf{D\text{-}PKE.Enc}(\mathsf{pk}, \mathbf{m}[i])$
  $\mathbf{c}_1[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{c}_0[i]|}$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{pk}, \mathbf{c}_b)$
return $(b = b')$

Figure 4.6: Left: the IND security game for deterministic PKE scheme D-PKE. Right: the $-IND security game for deterministic PKE schemes.

is negligible. Then, a key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{D\text{-}PKE.KGen}(1^\lambda)$ is chosen and according to a secret bit b either of the two message vectors is encrypted (component-wise). The second adversary $\mathcal{A}_2$ is run on the resulting vector of ciphertexts and the public key to output a bit b'. The adversary wins the game if it correctly guesses the hidden bit b, i.e., if $b = b'$. We define the advantage of an adversary $\mathcal{A}$ in the IND game (Figure 4.6 left) against scheme D-PKE by

$$\mathsf{Adv}^{\text{ind}}_{\mathsf{D\text{-}PKE},\mathcal{A}_1,\mathcal{A}_2}(\lambda) := 2 \cdot \Pr\left[\text{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{D\text{-}PKE}}(\lambda) = 1\right] - 1$$

and call a scheme IND-secure if the advantage of any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that adheres to the entropy requirements defined above is negligible.

Pseudorandom ciphertexts. We introduce a stronger notion of security for D-PKEs that we call $-IND-security and that captures that ciphertexts are indistinguishable from random strings. Here, the first stage of the adversary, namely $\mathcal{A}_1$, outputs only one message vector $\mathbf{m}$, with analogous min-entropy requirements as for the message vectors in standard IND-security. We present the $-IND notion on the right of Figure 4.6.

4.5.1 Deterministic Public-Key Encryption in the Random Oracle Model

As is the case for CIH-secure hash functions, in the random oracle model both adversarial stages $\mathcal{A}_1$ and $\mathcal{A}_2$ (as well as the construction) are given access to the random oracle. Furthermore, we need to strengthen the unpredictability requirement on the message distributions to hold also relative to the random oracle. In other words we require that for all $b \in \{0,1\}$ and $i \in \mathbb{N}$ and each choice of RO the probability

$$\Pr\left[x = \mathbf{m}_b[i] : (\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1^{\mathsf{RO}}(1^\lambda)\right]$$

is negligible; that is, the probability is *not* over the choice of random oracle but only over the random coins of $\mathcal{A}_1$.

In the random oracle model deterministic public-key encryption schemes achieving IND-security can be constructed from any IND-CPA secure encryption scheme via the so-called *Encrypt-with-Hash* random-oracle transformation. As we already saw, we can construct IND-CPA encryption schemes

from a trapdoor function (in the random oracle model), i.e., we get that trapdoor functions imply the existence of IND-secure deterministic public-key encryption schemes in the random oracle model. In the standard model, this implication is not known to hold.

The Encrypt-with-Hash Transformation

The Encrypt-with-Hash (EwH) transform constructs a deterministic public-key encryption scheme $\mathsf{EwH}[\mathsf{PKE}]$ from a (randomized) public-key encryption scheme PKE in the random-oracle model [BBO07]. Here, we present the EwH transformation in the unkeyed random oracle model as it was originally presented by Bellare et al. The key-generation of EwH generates a key pair using the key-generation algorithm of the base PKE scheme. Algorithm $\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{Enc}(m, \mathsf{pk})$ first computes random coins $r \leftarrow \mathsf{RO}(\mathsf{pk}\|m)$ and then invokes the base encryption algorithm on m and pk and with coins r to generate a ciphertext. The decryption routine is identical to that of the underlying scheme.[10] We next present the pseudocode for the EwH transformation in the random oracle model:

$\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)$	$\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{Enc}(\mathsf{pk}, m)$	$\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{Dec}(\mathsf{sk}, c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE}.\mathsf{KGen}(1^\lambda)$	$r \leftarrow \mathsf{RO}(\mathsf{pk}\|m)$	$m \leftarrow \mathsf{PKE}.\mathsf{Dec}(\mathsf{sk}, c)$
return $(\mathsf{pk}, \mathsf{sk})$	$c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}, m; r)$	**return** m
	return c	

Remark. We point out that, as the transform is defined in the unkeyed random oracle setting it is unclear how an instantiation with a keyed hash function should be defined as there are two possibilities: one can generate the hash key as part of the public-key or as a parameter upfront. Note that the difference is that in the IND-security games (cf. Figure 4.6) the first-stage adversary does not get access to the public key.

Bellare et al. [BBO07] show that the EwH transformation yields an IND-secure D-PKE scheme in the random oracle model when starting from any IND-CPA public-key encryption scheme. The proof intuition is that since messages are required to have at least super-logarithmic min-entropy, the randomness extracted from the messages via the random oracle is indistinguishable from actual random coins. This then allows to reduce the security of the scheme to the security of the underlying randomized public-key encryption scheme. We do not replicate the original proof here and refer the interested reader to [BBO07]. We note that we present a proof for an extension of the Encrypt-with-Hash transformation via an intermediate abstraction called a *universal computational extractor* in Chapter 9.

4.5.2 Deterministic Public-key Encryption in the Standard Model

So far, no candidate construction in the standard model yields a fully secure deterministic public-key encryption scheme which is not surprising given that Wichs shows that such a scheme cannot be founded on many standard assumptions [Wic13]. Based on standard assumptions, Bellare et al. [BFOR08] and Boldyreva et al. [BFO08] construct deterministic public-key encryption schemes

[10] In the original work, the decryption operation consists of an additional step which re-computes the ciphertext to ensure non-malleability. For our purpose of showing that the transformation is not sound, this additional step is irrelevant.

in the standard model for block sources where it is assumed that each message contains some fresh min-entropy (it contains min-entropy even conditioned on all other messages). Raghunathan et al. [RSV13] consider security notions for deterministic PKE schemes which are even stronger than IND, namely, they allow messages to depend on the public key (to a certain extent). They present constructions in the random oracle model as well as constructions in the standard model based on lossy trapdoor functions. Their standard-model constructions, similarly, only achieve IND-security against block sources.

Fuller, O'Neill and Reyzin [FOR12] use lossy trapdoor functions to build D-PKE-schemes that achieve q-bounded security for multi-message sources, a class that they introduce and that is incomparable to previous results [BFOR08, BFO08]. In particular, they require each message to have more than $\mathcal{O}(\mathsf{q} \cdot s)$ bits of min-entropy, individually. Here, q is an arbitrary polynomial that goes into the key-generation and gives an upper bound on the number of queries an adversary is allowed to make. Value s is a *lossiness parameter* that is induced by the lossy trapdoor function in their construction. This means that the length of each message needs to be longer than the number of queries tolerated by the system which is not required by previous results. In turn, the class considered by Fuller et al. allows messages to be *arbitrarily* correlated, which the standard-model constructions in [BFOR08, BFO08] cannot handle. For example, the distribution $(m, m+1, m+2, ..., m+q)$, where m is a message drawn at random from $\{0,1\}^{2sq}$ and $m+t$ is defined as counting t steps upwards when considering the string m as a binary number, is admissible for the construction of Fuller et al. [FOR12], but it is not a block-source, as $m+1$ has no min-entropy conditioned on m. Hence, the class of sources considered by Fuller et al. is incomparable to the class of sources considered in [BFOR08, BFO08] and both are strict subclasses of those considered by the strongest notion of IND security.

New Results. On the negative side, we show that the Encrypt-with-Hash random-oracle transformation as well as many other transformations are not generally sound, that is, not every secure IND-CPA randomized PKE scheme can be transformed into an IND-secure deterministic PKE scheme using EwH. To this end, we construct a specific randomized public-key encryption scheme PKE that is IND-CPA secure given that indistinguishability obfuscation exist. We then show that for every hash function H the transformed scheme $\mathsf{EwH}[\mathsf{PKE}, \mathsf{H}]$ is completely insecure and, in particular, not IND-secure (Chapter 8).

On the positive side we give the first standard model candidate construction for \$-IND-secure public-key encryption schemes (where \$-IND security is a strictly stronger notion than the standard IND security notion for D-PKEs). Based on indistinguishability obfuscation and point-function obfuscation we obtain a construction which is secure against adversaries seeing at most q-many encryptions (that is, key generation depends on polynomial q). We can further lift the bound on q under a restricted version of the non-standard *Superfluous Padding Assumption* for indistinguishability obfuscation which we postulate in this thesis. q-query deterministic PKE follows from a form of UCE security that we construct in Chapter 11. In Chapter 14 we present a direct construction which allows for further restrictions when attempting to lift the q-query bound with the help of the superfluous padding assumption that we introduce in Chapter 13.

4.6 Point-Function Obfuscation

Virtual black-box obfuscation and indistinguishability obfuscation (see the beginning of this chapter, page 47, as well as Chapter 5) are general-purpose code obfuscation schemes in that they consider obfuscation of arbitrary functionalities. Similarly, we can consider obfuscation for specific functionalities. Here we are interested in obfuscating very simple functions called *point functions*. A point function p_x for some value $x \in \{0,1\}^*$ is defined as

$$p_x(s) := \begin{cases} 1 & \text{, if } s = x \\ 0 & \text{, otherwise} \end{cases} .$$

While point functions as defined above only return a single bit, a point function with multi-bit output (MBPF) $p_{x,m}$ for values $x, m \in \{0,1\}^*$ is defined as

$$p_{x,m}(s) := \begin{cases} m & \text{, if } s = x \\ 0 & \text{, otherwise} \end{cases} .$$

For an MBPF $p_{x,m}$ we call x the *point address* and m the *point message* or *point value*.

Point-function obfuscation considers obfuscation schemes that take a point x (or a point x and message m) and output a point function p_x (resp. a MBPF $p_{x,m}$) which *hides* the point. For example, when considering virtual black-box (VBB) security (we formally introduce VBB obfuscation in Chapter 5), we would require that for any adversary $\mathcal{A}$ there exists a simulator Sim such that

$$\Big| \Pr\left[\mathcal{A}(1^\lambda, p_x) = 1 \,:\, x \leftarrow\!\!\$\, \{0,1\}^\lambda;\ p_x \leftarrow\!\!\$\, \mathsf{O}(x)\right] - \Pr\left[\mathsf{Sim}^{p_x}(1^\lambda) = 1 \,:\, x \leftarrow\!\!\$\, \{0,1\}^\lambda;\ p_x \leftarrow\!\!\$\, \mathsf{O}(x)\right] \Big| \leq \mathsf{negl}(\lambda) .$$

Here, the probability is over the random choice of x, the coins of the obfuscator and the coins of either $\mathcal{A}$ or simulator Sim. Jumping ahead, we note that there are various flavors for defining point obfuscation, some even stronger than the VBB definition above. We will discuss point obfuscation in detail in Chapter 6.

While point functions are simple objects, point functions and, in particular, point-function obfuscation plays an important role in access control. Consider, for example, the case where a user holds a secret password secret. In order to authenticate towards a service, the service needs to verify that the user knows the password. One way to achieve this is with the following program

CheckPassword(pw)

```
RefPw := 'TheSecretPassphrase'
if RefPw = pw then
  return true
return false
```

which essentially is a point function for the point: `TheSecretPassphrase`. Good practice is not to store passwords in the clear but to store only information that is sufficient to verify that a user knows the password, but which cannot be used by adversaries to retrieve the password. Storing

an obfuscation of the above point function would fulfill these requirements as then we have the guarantee that an adversary that, for example, due to a system's breach, gets hold of the obfuscated point function cannot do better than trying out every possible passphrase.

4.6.1 Point-Function Obfuscation in the Random Oracle Model

A standard practice for password storage is not to store the password itself but a hash of the password (or rather the hash of a salted password; see [BRT12] for a theoretical treatment of salting passwords). In this case we can consider the following check procedure

$\underline{\textsc{CheckPassword}^{\mathsf{H}}(\mathsf{pw})}$

$\mathsf{RefHash} := 0x.....;$ // The hash value of `'TheSecretPassphrase'`
if $\mathsf{RefHash} = \mathsf{H}(\mathsf{pw})$ **then**
  return true
return false

which, in the random oracle model, that is, if we think of H as a random oracle RO yields, in fact, a secure point-function obfuscation scheme.

The study of point-function obfuscation in the random oracle model was started by Lynn, Prabhakaran, and Sahai [LPS04] who also gave a simple random oracle construction for a MBPF obfuscation scheme. Here, in order to obfuscate (x, m) (i.e., point address x with point value m) one computes

$\overline{x} \leftarrow \mathsf{RO}(0\|x)$
$\overline{m} \leftarrow \mathsf{RO}(1\|x) \oplus m$
return $(\overline{x}, \overline{m})$

Knowing x one can recover m by computing $\overline{m} \oplus \mathsf{RO}(1\|x)$ and to check whether x is correct one can test whether $\mathsf{RO}(0\|x)$ is equal to $\overline{x}$. (Note that the above schemes are only correct if we consider an injective random oracle.)

4.6.2 Point-Function Obfuscation in the Standard Model

Efficient point-function obfuscation schemes are also known in the standard model based on different assumptions, for example, based on DDH(see Appendix A.3 for a formal definition). As point obfuscation, especially in the presence of auxiliary information, plays an important role in this thesis, we dedicate Chapter 6 as an introduction to point obfuscation and defer further discussion to there.

> **New Results.** We show that indistinguishability obfuscation and (very) strong forms of obfuscation for multi-bit output point functions are mutually exclusive (Chapter 7). We present standard model constructions of weaker forms of MBPF obfuscation and, in particular, show how to instantiate the random oracle in the point-function obfuscation scheme of Lynn et al. [LPS04] (Chapter 12).

Chapter 5

General-Purpose Code Obfuscation

```
perl -e '$b=shift;
         do {$_=int(rand(2**--$b))|2**$b++}
                    while ((1x$_) =~ /^1?$|^(11+)\1+$/);
                                                print' 15
```

Summary. In this chapter we give a broad introduction to the topic of general-purpose obfuscation focusing on indistinguishability obfuscation.

Chapter content

5.1 Introduction

In this and the following chapter we turn to *code obfuscation* which, on first glance, might seem unrelated to the random oracle methodology but which turns out to be an invaluable tool when it comes to studying and understanding random oracles. The study of code obfuscation asks: *can we write programs in such a way that they hide their secrets (i.e., how they work) even if their code is published?* An example of such a *secretive code piece* is the chapter quote where we tried to obfuscate a Perl program which (rather inefficiently) solves an important cryptographic task.[1] The mere existence of programs that hide their secrets is not sufficient for any practical purpose. We instead need an efficient mechanism to turn programs into obfuscated programs, or in other words, a mechanism which given the code of a program as input outputs code which implements the

[1] The obfuscation is based on an intriguing regular expression due to Abigail [Abi15]; we discuss the code in detail on page 69.

same program but which hides its secrets. We call such a mechanism a *code obfuscator*, a *program obfuscator*, or simply and shorter an *obfuscator*.

Code obfuscators can be very useful. For example, they can be used to protect intellectual property—assume that you want to grant rights to use software but to prevent anyone from decompiling it and reusing parts of the code. Also in the realm of cryptography obfuscators are useful. In their seminal paper on public-key encryption, Diffie and Hellman [DH76] consider the idea of using an obfuscator to turn a symmetric encryption scheme into a public-key encryption scheme by simply obfuscating the encryption algorithm with the secret key hard-coded into it. They write:

> *"Essentially what is required is a one-way compiler: one which takes an easily understood program written in a high level language and translates it into an incomprehensible program in some machine language. The compiler is one-way because it must be feasible to do the compilation, but infeasible to reverse the process."* [DH76]

Given a symmetric encryption scheme SE and secret key sk, Diffie and Hellman envisioned that with a good obfuscation scheme O one could define the corresponding public key pk as

$$\mathsf{pk}(\cdot) := \mathsf{O}(\mathsf{SE.Enc}(\mathsf{sk}, \cdot)).$$

Note that by the above definition public key pk is a function that takes as input a single value (a message m) which it encrypts with the underlying symmetric encryption scheme and with secret key sk. Since, the obfuscation scheme *hides the code's secrets* it should certainly hide secret key sk and hence anybody can use the program (i.e., the public key) to generate encryptions but only the knowledge of key sk should allow for decrypting ciphertexts.[2]

Defining obfuscation. While the intuition behind good obfuscation is easy to state—the obfuscated code should hide the program's secrets—it is challenging how to grasp this formally, that is, how to define what exactly *hiding the secrets* means. One approach, often seen in software engineering, is to consider heuristics (e.g., renaming of variables, introducing bogus methods, etc.) which hopefully make "reverse-engineering" harder [CT02]. Here, we are interested in a precise mathematical definition which allows us to give precise statements about the security of an obfuscator.

Consider Yao's millionaire problem [Yao82a], a classical problem from the field of secure function evaluation where two millionaires, Alice and Bob, want to learn who of them is richer without revealing their wealth.[3] Assume that w_A denotes the wealth of Alice. Then one way for Alice and Bob to compute who is richer would be for Alice to obfuscate the following program which has w_A hard-coded into it:

$\textsc{Millionaire}[w_A](w)$

if $w_A < w$ **then**
  return Bob
elseif $w_A > w$ **then**
  return Alice
return identical wealth

[2] Note that this reasoning requires some form of asymmetry in the encryption and decryption process since if we could use the same encryption box to also perform the decryption operation then essentially anybody can decrypt. For example, stream ciphers usually have the property that encryption and decryption are identical since encryption and decryption is accomplished by xor-ing a stream of pseudorandom bits to the message.

[3] The idea of using Yao's millionaire problem to introduce obfuscation is taken from a talk given by Sahai at the Simons Institute, 2015.

Example 5.1: The Chapter Quote.

```
1 perl -e '$b=shift;
2          do {$_=int(rand(2**--$b))|2**$b++}
3                     while ((1x$_) =~ /^1?$|^(11+)\1+$/);
4                                                print' 15
```

The chapter quote is a simple Perl program which generates a random 15-bit prime number. It is inspired by a cute regular expression attributed to Abigail [Abi15] which tests a (unary) number for primality. Let us analyze the program step by step.

The expression `perl -e` takes a string and interprets it as a Perl program passing on any following expressions as arguments. Thus, in the above the string within single quotes is executed and value 15 is given as its single argument. (15 will be the bit length. You may go higher than that but due to the program's inefficiency do not expect anything much higher than 20 to work.)

The first part stores value `15` in variable `$b` and the very last command (line 4) prints whatever value is currently stored in Perl's special variable `$_`. To get there we need to understand lines 2 and 3. Here, we see a *do-while* loop where in line 2 we store a value in variable `$_` and in line 3 test whether that value is of a specific form. If this test fails we will jump back to line 2 and try again. When looking closely at line 2 we see that the first command executed is `--$b`, that is, value `$b` is decreased by 1 to 14. Then `rand(2**14)` is computed which returns a random (fractional) number between 0 and 2^{14}. The expression `int()` transforms the fractional number into an integer. Finally, `2**$b++` adds 2^{14} (sets the bit at position 15 to one) and then updates `$b` back to 15. The result of the entire computation is stored in `$_`. In summary, what we have is a random 15-bit number stored in `$_`.

Line 3 now says that we should repeat the above process as long as the number that we get is composite or, in other words, we may proceed to line 4 only in case the number was prime. For this test, the number is converted into unary by `1x$_` and then tested against the regular expression `^1?$|^(11+)\1+$` which consists of two parts `^1?$` and `^(11+)\1+$`. The first part checks if the number in `$_` is zero or one and, since we know that it is at least 2^{14} is actually superfluous. The second part now does the actual work. It checks whether the number can be decomposed in a group consisting of at least two ones `(11+)` and a multiple of that group `\1+`. This is best explained with an example. Consider the number 15 which in unary is written as

111 111 111 111 111

that is, it can be written as five times the group of three 1s or as the regular expression would see it, as a group (111) and a multiple of that group (111){4}. Such a decomposition can, however, only be obtained if the number is composite and, hence, if the regular expression matches the number is not prime.

When considering efficiency, the above code is, of course, catastrophic as we effectively implemented a brute force search to check that a number is not a composite number.

Now, if Bob is given an obfuscation of the above program and we assume that it hides all secrets then Bob should not be able to learn w_A. This is, of course, not correct since given just the functionality Bob can easily recover w_A using binary search. Thus, if something is learnable from interacting with the functionality by specifying inputs and observing outputs then it cannot be hidden by an obfuscation.

This leads to a natural definition of obfuscation which asks that an obfuscation of a program should only leak what can be learned from the program when interacting with it as an oracle

(i.e., with black-box access). This idea of obfuscation, called *virtual black-box (VBB) obfuscation*, was originally formalized by Barak et al. [BGI+01, BGI+12] building upon work by Hada [Had00]. Besides formalizing virtual-black box obfuscation, Barak et al. gave a negative result showing that there exist (contrived) functions which cannot be (virtual black-box) obfuscated. Motivated by this negative result, they also gave weaker definitions of obfuscation, most notably, they defined a so-called *indistinguishability obfuscator* which gives a very weak security guarantee: given two programs P_1 and P_2 that compute the same functionality, that is, for all inputs x we have that $\mathsf{P}_1(x) = \mathsf{P}_2(x)$, we have that an adversary that gets as input an obfuscation of either one of them cannot tell whether it was given an obfuscation of P_1 or of P_2. Intuitively, it is unclear what we can use such an obfuscator for:

> *"Certainly, when we thought of it back then, we thought it was a useless definition."*
>
> Amit Sahai, Simons Institute, 2015 Cryptography Boot Camp

As it turns out, indistinguishability obfuscation is tremendously useful as documented by the various breakthrough-results for hard problems that followed the publication of a candidate construction for indistinguishability obfuscation by Garg et al. [GGH+13b] in 2013. In particular, indistinguishability obfuscation together with one-way functions is sufficient to construct most basic cryptographic functionalities (such as signature schemes or encryption schemes) [SW14] as well as various highly complex primitives, some of which where previously not known to exist. For example, Garg et al. [GGH+13b] showed that indistinguishability obfuscation can be used to construct a functional encryption scheme for all circuits. Sahai and Waters go so far as to envision indistinguishability obfuscation becoming a "central hub" of cryptography [SW14]. They write:

> "*In an ideal future 'most' cryptographic primitives will be shown to be realizable from iO and one way functions.*" [SW14]

Jumping ahead, also most results in this thesis including the instantiations of random oracle schemes discussed in the previous chapter are based on indistinguishability obfuscation.

A note for the impatient reader. In this chapter we set out to give a broad overview of general-purpose obfuscation while in Chapter 6 we consider special-purpose obfuscators, namely, obfuscators which solely work on point functions. We note that most of the discussion in this chapter is irrelevant for understanding the results of this thesis but is purely meant as background material on one of our main tools: indistinguishability obfuscation. If you are familiar with general-purpose obfuscation and the notions of VBB, VGB, indistinguishability obfuscation and differing-inputs obfuscation you can safely skip this chapter. The main definition to take along is the definition of an indistinguishability obfuscator which we define in a game-based formulation (Definitions 5.5 and 5.6, on page 78, together with Definition 5.1 which defines the functionality preserving property of obfuscators). Additionally, in this chapter we introduce the notion of puncturable pseudorandom functions (Definition 5.9 on page 88) which alongside indistinguishability obfuscation is one of the main tools for our positive results (standard-model constructions of various primitives).

5.2 Defining Obfuscation

Obfuscation schemes take as input a program description and output a description of a program that has the same functionality. In the following, we use the circuit model of computation and thus consider obfuscators that take as input a description of a circuit C and output a description of an obfuscated circuit $\overline{C}$. This does not mean that the obfuscation algorithm itself is modeled as a circuit, but that it works on circuits. We note that for readability we usually do not distinguish between the description $\langle C \rangle$ of a circuit C and the circuit itself. That is, we often consider C to denote the description of a circuit and write $C(x)$ to denote the circuit's output on input x. Note that, in this thesis, we only consider obfuscation of deterministic functionalities and hence any program or circuit to be obfuscated is deterministic. Obfuscation of probabilistic functionalities has recently been considered by Canetti et al. [CLTV15].

When we speak of a general-purpose obfuscation scheme we consider efficient algorithms that take as input a circuit and which output a functionally equivalent circuit. Additionally, we require that the obfuscated circuit is of polynomial size in the sum of the size of the input circuit and the security parameter. This is captured by the following basic obfuscation definition which defines the *functionality preserving* and *polynomial slowdown* properties:

Definition 5.1 (Obfuscation scheme)**.** *A* PPT *algorithm* O *is called an* obfuscation scheme *for a circuit ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *of poly-size circuits if it satisfies:*

Functionality Preserving. *For any* $\lambda \in \mathbb{N}$ *and* $C \in \mathcal{C}_\lambda$*, the obfuscated circuit* $\mathsf{O}(C)$ *computes the same function as* C*, that is, for all* x *it holds*

$$\Pr\left[C'(x) = C(x) \;\middle|\; C' \leftarrow\!\!\$\, \mathsf{O}(1^\lambda, C)\right] = 1.$$

Polynomial slowdown. *There exists a polynomial* poly *such that for any* $\lambda \in \mathbb{N}$ *and* $C \in \mathcal{C}_\lambda$*,*

$$\Pr\left[|C'| \leq \mathsf{poly}(|C| + \lambda) \;\middle|\; C' \leftarrow\!\!\$\, \mathsf{O}(1^\lambda, C)\right] = 1.$$

What this basic definition does not capture is any form of security. The remainder of this chapter introduces various forms of security definitions for general-purpose obfuscators starting with the strongest form of obfuscation: virtual black-box obfuscation.

5.3 Virtual Black-Box Obfuscation

We next present the formalization of worst-case virtual black-box (VBB) obfuscation due to Barak et al. [BGI+12, GK05] which captures the idea that an obfuscation should not leak any more information than (black-box) interaction with the functionality should.[4] This is formalized by requiring that for any efficient binary adversary $\mathcal{A}$ (an adversary that outputs either 0 or 1) there exists an efficient simulator Sim such that for any circuit C the following distributions are computationally

[4] Barak et al. consider virtual black-box obfuscation without additional auxiliary information [BGI+12]. The stronger auxiliary information setting that we present here is due to Goldwasser and Tauman-Kalai [GK05].

indistinguishable: $\mathcal{A}(1^\lambda, \mathsf{O}(C))$, where the adversary is given as input an obfuscation of C, and $\mathsf{Sim}^C(1^\lambda, |C|)$, where the simulator is given only oracle access to C and the size of C. In addition to getting an obfuscated circuit as input we can consider obfuscation in the presence of *auxiliary information* where both simulator and adversary get some additional information (which may depend on circuit C) as input.

The following definition as well as all later definitions of obfuscators build upon Definition 5.1, that is, we do not repeat the functionality preserving and polynomial slowdown properties.

Definition 5.2 (Worst-case virtual black-box obfuscator with auxiliary input (VBB-AI))**.**
A PPT *obfuscation scheme* O *for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *of poly-size circuits is a* worst-case virtual black-box obfuscator with auxiliary input *for* $\mathcal{C}$ *if it satisfies:*

VIRTUAL BLACK-BOX. *For any* PPT *adversary* $\mathcal{A}$ *there exists a* PPT *simulator* Sim *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, for all* $C \in \mathcal{C}_\lambda$ *and* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ *it holds that*

$$\left|\Pr\left[\mathcal{A}(\mathsf{O}(1^\lambda, C), \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^C(1^\lambda, 1^{|C|}, \mathsf{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda) \tag{5.1}$$

where the probability is taken over the coins of $\mathcal{A}$*,* Sim*, and* O*.*

Note that the order of quantification allows the auxiliary information aux to depend on circuit C. This is also sometimes referred to as *dependent auxiliary information* in contrast to *independent* auxiliary information which is not allowed to depend on the obfuscated circuit. Whenever we consider auxiliary information, we assume the stronger *dependent* setting unless explicitly stated otherwise.

5.3.1 Average-Case Obfuscation

We defined VBB obfuscation in a *worst-case* setting, where worst-case is meant relative to the obfuscator: the obfuscator has to work even for the worst possible combination of circuit and auxiliary information. In contrast, we can also define a weaker requirement where the obfuscator is only required to be secure on average (the definition goes back to the work of Goldwasser and Tauman-Kalai [GK05]). For this, we change the above definition and require that a slightly adapted version of Equation 5.1 holds if circuit C is chosen uniformly at random from $\mathcal{C}_\lambda$.

Definition 5.3 (Average-case virtual black-box obfuscator with auxiliary input (aVBB-AI))**.**
A PPT *obfuscation scheme* O *for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *of poly-size circuits is an* average-case virtual black-box obfuscator with auxiliary input *for* $\mathcal{C}$ *if it satisfies:*

VIRTUAL BLACK-BOX. *For any* PPT *adversary* $\mathcal{A}$ *there exists a* PPT *simulator* Sim *such that for all sufficiently large* $\lambda \in \mathbb{N}$*, for all predicates* π*, and for all auxiliary information functions* $\mathsf{aux} : \{0,1\}^{\mathsf{poly}(\lambda)} \to \{0,1\}^{\mathsf{poly}'(\lambda)}$ *it holds that*

$$\left|\Pr_{C \leftarrow\$ \mathcal{C}_\lambda}\left[\mathcal{A}(\mathsf{O}(1^\lambda, C), \mathsf{aux}(C)) = \pi(C, \mathsf{aux}(C))\right] - \Pr_{C \leftarrow\$ \mathcal{C}_\lambda}\left[\mathsf{Sim}^C(1^\lambda, 1^{|C|}, \mathsf{aux}(C)) = \pi(C, \mathsf{aux}(C))\right]\right| \leq \mathsf{negl}(\lambda)$$

where the probability is taken over the random choice of circuit C *and the coins of* $\mathcal{A}$*,* Sim*, and obfuscator* O*.*

Observe that we formalize the auxiliary information as a function in order to allow the auxiliary information to depend on the sampled circuit. Further note that if we keep Equation 5.1 as is and there exists an efficient way to sample from $\mathcal{C}_\lambda$ then a trivial simulator exists that simply ignores its oracle, samples a fresh circuit, obfuscates it and runs the adversary. Thus, to get a meaningful definition, we instead ask adversary and simulator to output a predicate π of circuit and auxiliary information. For the worst-case definition the two formulations (with and without predicate π coincide [BGI+12]). For the remainder of this thesis we consider the worst-case formulation of VBB unless explicitly stated otherwise.

5.3.2 Impossibility of General VBB Obfuscation

Barak et al. [BGI+12] not only gave a clear and intuitive definition of what we would like to get from good obfuscators—a good obfuscator should hide everything except for the functionality—they also showed that such obfuscators do not exist in general. In order to show that VBB obfuscators do not exist in general it is sufficient to come up with a single function which cannot be obfuscated by any obfuscator. For this, consider, for the moment, a physical lock and key (we will shortly see how to grasp these mathematically). The functionality of a lock is that, given the right key, the lock opens and, given the wrong key, the lock does not. If given a lock and a key one can thus easily check if key and lock fit together, i.e., if the key opens the lock. Assuming there was a form of physical obfuscation that hides how the key and lock work (think, e.g., of an electronic key), then still, being given an obfuscated lock and an obfuscated key one should be able to check if the key opens the lock as we require from obfuscation (and thus from physical obfuscation) that the functionality is not changed. Now, consider that you are only given oracle access to the key and lock. Can you still check whether key and lock fit together? In the physical world, we can imagine that we have a friend at a remote location that holds the lock and a second friend (who knows nothing of the other friend) who holds the key. We can send keys to our friend with the lock and he or she tells us if they fit and we can send locks to our key friend learning if the sent lock matches his or her key, but since both friends hold on to their possession we cannot bring their key and lock together.

Let us translate the above intuition into mathematical objects. We consider the key to be a multi-bit output point function $p_{\alpha,\beta}$ which outputs $\perp$ everywhere[5] except on input α where it outputs β, i.e.,

$$p_{\alpha,\beta}(x) := \begin{cases} \beta & \text{if } x = \alpha \\ \perp & \text{otherwise} \end{cases}$$

Our lock will be a more complicated function $T_{\alpha,\beta}$ which interprets its input as a program description which it evaluates on input α. If this program returns β then $T_{\alpha,\beta}$ outputs 1 and otherwise it outputs $\perp$:

$$T_{\alpha,\beta}(\mathsf{P}) := \begin{cases} 1 & \text{if } \mathsf{P} \text{ has valid encoding and } \mathsf{P}(\alpha) = \beta \\ \perp & \text{otherwise} \end{cases}$$

[5]We often let circuits output $\perp$ instead of 0 to suggest that this output does not contain useful information.

Given the code of a key function $p_{\alpha,\beta}$ and a test function $T_{\gamma,\delta}$, we can check if $(\alpha,\beta) = (\gamma,\delta)$ by computing $T_{\gamma,\delta}(p_{\alpha,\beta})$. Note that this is independent of the way the code looks like, that is, whether or not we get obfuscated code or not, we can run the above check. However, any efficient algorithm that is given only black-box access to the functionalities, that is,

$$\mathcal{A}^{p_{\alpha,\beta},T_{\gamma,\delta}}(1^\lambda)$$

will fail with overwhelming probability in learning whether or not $(\alpha,\beta) = (\gamma,\delta)$ (assuming that the values are of super-logarithmic length in the security parameter). For this note that in order to learn anything from an oracle that implements $p_{\alpha,\beta}$ one would need to guess α and in order to learn anything from an oracle implementing $T_{\gamma,\delta}$ one would need to guess δ. Hence, if α and β are uniformly random and $|\alpha|$ and $|\delta|$ are in $\omega(\log\lambda)$ where λ is the security parameter, then the probability of guessing these values even for an unbounded algorithm which is restricted to make only polynomially many oracle queries is negligible.

The above formalization directly yields an impossibility result for a slightly stronger form of obfuscation where we obfuscate two circuits or Turing machines (p and T) in parallel.

Proposition 5.1 ([BGI+12] Proposition 3.4)**.** 2-*Turing machine or* 2-*circuit virtual black-box obfuscation for the class of all polynomial-time programs (resp. polynomial-sized circuits) does not exist.*

Barak et al. go on to strengthen their result to also rule out general VBB obfuscation of a single Turing machine or circuit. Here, we present the idea for Turing machines and refer to [BGI+12] for how to handle circuits. Interestingly, obfuscating Turing machines seems much harder than obfuscating circuits which is also reflected by the case that ruling out VBB obfuscation for Turing machines is straight forward given the above intuition, while ruling out VBB obfuscation for circuits requires much more finesse.

In order to bootstrap the above impossibility result to the general 1-Turing machine setting, we combine the two functions into a single "key-lock function". As Turing machines take arbitrarily long inputs we can combine two functionalities by encoding them in one Turing machine and use, for example, the first input bit to denote which functionality should be executed. This yields the combined complex program:

$$pT_{\alpha,\beta,\gamma,\delta}(x) := \begin{cases} p_{\alpha,\beta}(x[2..|x|]) & \text{if } x[1] = 0 \\ T_{\gamma,\delta}(x[2..|x|]) & \text{if } x[1] = 1 \end{cases}$$

If we write the input as $b\|x$ (where b denotes a single bit) then we can rewrite the above as

$$pT_{\alpha,\beta,\gamma,\delta}(b\|x) := \begin{cases} \beta & \text{if } b = 0 \wedge x = \alpha \\ 1 & \text{if } b = 1 \text{ and if } x \text{ has valid encoding and if } x(\gamma) = \delta \\ \bot & \text{otherwise} \end{cases}$$

Since Turing machines process arbitrarily long inputs we can, in particular, run a Turing machine on its own description. Thus, given the code of a program $pT_{\alpha,\beta,\gamma,\delta}$ we can define two programs $pT\big|_0$

and $pT\big|_1$ which simply fix the first bit of the input to $pT_{\alpha,\beta,\gamma,\delta}$ to either 0 or 1. That is,

$$pT\big|_b(x) := pT_{\alpha,\beta,\gamma,\delta}(b\|x),$$

which is functionally equivalent to $T_{\gamma,\delta}$ if $b = 1$ and functionally equivalent to $p_{\alpha,\beta}$ if $b = 0$. As in the two-function case above, we have the code of $pT\big|_0$ and $pT\big|_1$ which allows us to, again, test if $(\alpha, \beta) = (\gamma, \delta)$ by computing

$$pT\big|_1(pT\big|_0) = T_{\gamma,\delta}(p_{\alpha,\beta}).$$

Note that this argument only works since we can effectively run a Turing machine on a description of itself. For circuits, on the other hand, the same approach does not work as we require the size of $pT\big|_0$ to be at most the input length of $pT\big|_1$ as, for circuits, the input length is restricted. However, by construction we have that $\left|pT\big|_0\right| = \left|pT\big|_1\right|$ and, thus, in particular the size of $pT\big|_0$ is larger than the input length supported by $pT\big|_1$.

To work around this, Barak et al. basically construct a symmetric homomorphic encryption scheme allowing to test whether $p_{\alpha,\beta}$ and $T_{\gamma,\delta}$ match given the code of $p_{\alpha,\beta}$ and oracle access to a special version of $T_{\gamma,\delta}$. Due to the use of pseudorandom functions in the construction for circuits the impossibility result is thus conditioned on the existence of one-way functions. We refer to [BGI+12] for a detailed description.

Theorem 5.2 ([BGI+12] Theorems 3.5 and 3.7)**.** *Virtual black-box obfuscators for Turing machines do not exist. Furthermore, if one-way functions exist, then virtual black-box obfuscators for circuits do not exist.*

Note that the above impossibility result does not require any auxiliary information aux. If we consider the auxiliary information setting then stronger impossibility results are known [GK05, BCC+14]. Furthermore, also in idealized models (such as the random oracle model) impossibility results for VBB obfuscation are known [CKP15, Pas15, MMN15].

5.4 Virtual Grey-Box Obfuscation

The impossibility result for general VBB obfuscation led to a search for weaker definitions that do not suffer from impossibility results. A possible relaxation in the VBB definition is to consider simulators which are unbounded, but which are still restricted to make at most polynomially many queries. This yields the so-called *virtual grey-box* (VGB) definition for obfuscation which was introduced by Bitansky and Canetti [BC14] in their study of point-function obfuscators.

Definition 5.4. *A* PPT *obfuscation scheme* O *for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *of poly-size circuits is a* worst-case virtual grey-box obfuscator with auxiliary input *for* $\mathcal{C}$ *if it satisfies:*

Semi-bounded simulation. *For any* PPT *adversary* $\mathcal{A}$ *there exists an unbounded simulator* Sim *which makes at most polynomially many oracle queries such that for all sufficiently large* $\lambda \in \mathbb{N}$*, all* $C \in \mathcal{C}_\lambda$ *and* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$*:*

$$\left|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{O}(1^\lambda, C), \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^C(1^\lambda, 1^{|C|}, \mathsf{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where the probability is taken over the coins of $\mathcal{A}$, Sim *and* O.

Similarly to VBB obfuscation we can define an average-case virtual grey-box obfuscator that considers security only over a uniformly random choice of the to-be-obfuscated circuit.

VGB obfuscation is weaker than VBB obfuscation since if a bounded simulator exists, then so does a semi-bounded one. Similarly to VBB obfuscation we do, however, also have impossibility results for VGB obfuscation. As noted by Bitansky and Canetti the Turing machine counter example that we have seen above for VBB obfuscation also yields an unobfuscateable function for VGB obfuscation for Turing machines.

Proposition 5.3 ([BC14])**.** *Virtual grey-box obfuscators for Turing-machines do not exist.*

Furthermore, the 2-circuit example that we have seen for VBB obfuscation also applies to VGB obfuscators. However, we do not know of any counter examples for VGB obfuscation of a single circuit and in fact, we do have a candidate construction for VGB obfuscation for NC^1 circuits under novel and strong assumptions [BCKP14].

5.4.1 Virtual Grey-Box Obfuscation with Auxiliary Information

We defined virtual grey-box obfuscation also in the auxiliary input setting where both adversary and simulator can get an additional string aux which, in particular, may depend on the obfuscated circuit. While for VBB obfuscation auxiliary inputs seem to strengthen the assumption on the obfuscation scheme, the notions of (worst-case) VGB obfuscation with and without auxiliary information are equivalent as shown by Bitansky and Canetti [BC14].[6]

Proposition 5.4 ([BC14] Proposition A.3)**.** *Let* O *be a worst-case VGB obfuscator without auxiliary information for a circuit ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$*. Then* O *is also a worst-case VGB obfuscator with auxiliary input for the ensemble.*

The proof idea is that for any fixed auxiliary information, the VGB property tells us that there exists a good simulator, but for any aux this can be a different one. However, as VGB allows for unbounded simulators given aux we can compute the *best simulator* and, thus, reverse the order of quantifiers. We refer to [BC14] for further details but note that the above argument critically requires that the simulator is good for any circuit and any auxiliary input (in contrast to average-case obfuscators where the simulator only needs to be good over a random choice of circuits).

5.4.2 VGB Obfuscation for Pseudorandom Functions

For the results presented in this thesis we usually obfuscate one out of two objects: pseudorandom functions or point functions (or more complex circuits containing point functions and/or pseudorandom functions as the "main ingredient"). While VGB obfuscation was introduced in the setting of point-function obfuscation, it is not quite clear what a VGB obfuscation of a pseudorandom function guarantees. Intuitively, given sufficient oracle queries to a keyed PRF, an unbounded algorithm should be able to recover the key (up to redundancy in the key) and, thus, there is a trivial simulator: the simulator extracts the key from the oracle, constructs the pseudorandom function,

[6] We note that this equivalence is not known to hold for average-case VGB obfuscation.

obfuscates it and then runs the adversary on the obfuscation outputting whatever the adversary outputs. Consequently, it is not clear what, if anything, is hidden whenever we obfuscate a PRF using VGB obfuscation.

Luckily, PRFs may come with additional properties that break the above intuition and allow us to argue that VGB obfuscation and also weaker notions of obfuscation, in particular, *indistinguishability obfuscation*, hide critical aspects of the function, thereby allowing us to use obfuscation of such special PRFs in a large variety of applications. One such property which makes PRFs ideal for obfuscation is referred to as *constrainability*. A *constrained PRF* allows to generate restricted keys which allow to evaluate the PRF only on a subset of its original domain. In particular, we will be interested in a special case called *puncturable PRFs* where a punctured key k_S is associated to a key k and a polynomial sized set S and allows to evaluate the PRF with key k on every input x which is not in S. We will see that even the most restricted case where S consists of only a single value is highly useful when used together with obfuscation. For this note that if a simulator is given oracle access to the PRF with such a punctured key, that is, we consider $\mathsf{Sim}^{\mathsf{F.Eval}(\mathsf{k}_S,\cdot)}$ for a uniformly chosen key and set S, then even if Sim is unbounded it will fail with overwhelming probability in recovering S as its number of oracle queries is restricted to be polynomial.[7]

Puncturable PRFs were proposed in the context of indistinguishability obfuscation which is a further weakening of a general-purpose obfuscator that we discuss next. We formally introduce puncturable PRFs in Section 5.5.5 when we introduce techniques of working with indistinguishability obfuscation in cryptographic contexts.

5.5 Indistinguishability Obfuscation

When Barak et al. presented their negative results on VBB obfuscation they also introduced two weaker forms of obfuscation, called *indistinguishability obfuscation* and *differing-inputs obfuscation* [BGI+12]. Back when the definitions were introduced, it was not at all clear whether they could be useful:

> *"I must admit that I was very skeptic of the possible applicability of the notion of indistinguishable obfuscation."* Oded Goldreich, [Gol13]

However, none of the two definitions was known to be impossible. We note that recently, Garg, Gentry, Halevi, and Wichs showed a conditional impossibility result for differing-inputs obfuscation stating that if a special-purpose obfuscator for a signature scheme exists, then differing-inputs obfuscation cannot exist in general [GGHW14]. For indistinguishability obfuscation, on the other hand, until today we do not know of any impossibilities and recent candidate constructions allow us to gain confidence in the existence of a general purpose indistinguishability obfuscator [GGH+13b, PST14, AGIS14, BR14, BGK+14, GLSW14, AB15, Zim15, AJ15, BV15].

While indistinguishability obfuscation (iO) intuitively captures that the obfuscation of two functionally equivalent circuits cannot be distinguished, differing-inputs obfuscation (diO) says that if it is difficult to find inputs where two circuits differ, then their obfuscations are indistinguishable (or the contrapositive: if an efficient distinguisher for two obfuscated circuits exists then one can

[7] We note that $\mathsf{F.Eval}(\mathsf{k}_S,\cdot)$ is not defined on values $x \in S$ but the simulator only notices this discrepancy if it is able to recover a point in S.

efficiently find an input on which the two circuits differ). Both definitions stand in stark contrast to the previous VBB and VGB definitions. While VBB considers the obfuscation of a single circuit iO and diO consider the distinguishing difference between two obfuscations. However, indistinguishability obfuscation and restricted forms of differing-inputs obfuscation can be seen to be strictly weaker than VGB providing us with a hierarchy of general purpose obfuscators which at the one extreme has virtual black-box obfuscation and on the other extreme has indistinguishability obfuscation. In the following, we present and discuss the notion of indistinguishability obfuscation in detail before defining and discussing differing-inputs obfuscation in Section 5.6.

5.5.1 Defining Indistinguishability Obfuscation

We present a game-based definition, following the definitional framework of [BST14] which captures indistinguishability-obfuscation-like notions via the security game IO and a class of samplers Sam which on input the security parameter output two circuits C_0 and C_1 and possibly some auxiliary information aux.

Definition 5.5. *We call an algorithm* Sam *a* circuit sampler *if on input the security parameter* 1^λ *sampler* Sam *outputs* (C_0, C_1, aux) *where* C_0 *and* C_1 *are descriptions of polynomially sized circuits and* aux *is a string of polynomial length. If* Sam *is a circuit sampler and* O *is an obfuscation scheme we define the advantage* $\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\cdot)$ *for a distinguisher* D *relative to game* IO*:*

$$\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda) := 2 \cdot \Pr\left[\mathsf{IO}^{\mathsf{O}}_{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] - 1$$

$\underline{\mathsf{IO}^{\mathsf{O}}_{\mathsf{Sam},\mathsf{D}}(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\}$
$(C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$\overline{C} \leftarrow\!\!\$\ \mathsf{O}(1^\lambda, C_b)$
$b' \leftarrow\!\!\$\ \mathsf{D}(1^\lambda, \overline{C}, \mathsf{aux})$
return $(b = b')$

If $\mathcal{S}$ *is a class of circuit samplers, then we say that* O *is* $\mathcal{S}$*-secure if for all* $\mathsf{Sam} \in \mathcal{S}$ *and all efficient distinguishers* D *advantage* $\mathsf{Adv}^{\mathrm{io}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda)$ *is negligible in* λ*.*

We can now capture indistinguishability obfuscation via restricting the class of samplers to so-called *equality samplers*.

Definition 5.6 (Equality circuit sampler)**.** *Let* Sam *be a circuit sampler. We call* Sam *an* equality circuit sampler *if with overwhelming probability over the coins of* Sam *we have that the circuits* C_0 *and* C_1 *have the same size, number of inputs and number of outputs and are functionally equivalent, that is*

$$\Pr_{(C_0,C_1,\mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)}\left[|C_0| = |C_1| \wedge \forall x : C_0(x) = C_1(x)\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Let $\mathcal{S}_{\mathbf{eq}}$ be the class of all equality circuit samplers (including non-efficient samplers). Then we capture the notion of indistinguishability obfuscation by Barak et al. [BGI+12] by an obfuscation scheme O secure for $\mathcal{S}_{\mathbf{eq}}$.[8]

[8] The original definition of iO by Barak et al. [BGI+12] is not game-based but quantifies over all functionally equivalent circuits stating that for any pair no efficient adversary can distinguish between their obfuscations. We note that the two definitions are equivalent. For this note that a sequence of pairs of functionally equivalent circuits can be described by a deterministic non-uniform sampler. The converse direction follows via an averaging argument.

5.5.2 IO from a Complexity Theoretic Perspective

To get a better understanding of indistinguishability obfuscation we next present a brief discussion of iO from the viewpoint of complexity theory. Most cryptographic primitives imply one-way functions which can be regarded as one of the simplest building blocks in cryptography or, in Impagliazzo's characterization, the fundamental building block of *Minicrypt* which captures a world in which one-way functions exist but public-key cryptography is impossible [Imp95].

Let us first establish that virtual black-box obfuscation implies one-way functions. As VBB obfuscation is a powerful primitive this is rather unsurprising and was already observed by Barak et al. [BGI+12]. For intuition, note that an efficient simulator that gets black-box access to the function that evaluates to zero everywhere or to a function that evaluates to zero everywhere but for a randomly chosen point x should not be able to distinguish with only polynomially many oracle queries. On the other hand, due to the functionality preserving requirement of the obfuscator, obfuscations of the two functions have a distinct support which induces distributions that are computationally indistinguishable but which are statistically far and which in turn implies the existence of one-way functions [Gol90, IL89]. The result was further generalized to also apply to imperfect VBB obfuscators which are only required to preserve the functionality with overwhelming probability [KMN+14].

Lemma 5.5 ([BGI+12] Lemma 3.8)**.** *If VBB obfuscators exist, then one-way functions exist.*

Proof sketch. Consider a point function $p_{\alpha,b}$ defined as

$$p_{\alpha,b}(x) := \begin{cases} b & \text{if } x = \alpha \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha \in \{0,1\}^\lambda$ and b is a single bit. Now we define a function family f_λ for $\lambda \in \mathbb{N}$ as

$$\begin{array}{l} \underline{f_\lambda(\alpha, b, r)} \\ \overline{C} \leftarrow \mathsf{O}(1^\lambda, p_{\alpha,b}; r) \\ \textbf{return } \overline{C} \end{array}$$

That is, f_λ returns the obfuscation of $p_{\alpha,b}$ computed with random coins r. We require that $\alpha \in \{0,1\}^\lambda$ and that $|r| = \mathsf{O.rl}(\lambda)$, that is, r is as long as the randomness required by O on security parameter 1^λ.

Since, by assumption, the obfuscator O is efficient so is our function f_λ. Furthermore, since O preserves the functionality of $p_{\alpha,b}$ it follows that b is information-theoretically determined by $f_\lambda(\alpha, b, r)$ and hence an adversary in the one-way experiment needs to recover, at least, b given $f_\lambda(\alpha, b, r)$. By the security of the VBB obfuscator we have that an efficient adversary is at most as good as an efficient simulator which only has black-box access to $p_{\alpha,b}$. In order to learn b given black-box access to $p_{\alpha,b}$ for a uniformly random α an adversary must guess α which it can do only with probability $q \cdot 2^{-\lambda}$ where q is an upper bound on the number of oracle queries. Thus, for any efficient simulator Sim, we have that

$$\Pr\left[\mathsf{Sim}^{p_{\alpha,b}}(1^\lambda) = b \,:\, \alpha \leftarrow\!\!\$\, \{0,1\}^\lambda, b \leftarrow\!\!\$\, \{0,1\}\right] \leq \frac{1}{2} + q(\lambda) \cdot 2^{-\lambda}.$$

This implies that b is a hardcore bit for f_λ which, in turn, implies that f_λ is a *weak one-way function*, implying the existence of one-way functions [Yao82b, Gol00]. □

The above proof works identically for virtual grey-box obfuscation, so also VGB obfuscation implies the existence of one-way functions. Surprisingly, this does not hold for indistinguishability obfuscators. Indeed, as noted by Barak et al. [BGI+12], inefficient iO exists and, in particular, if $\mathsf{P} = \mathsf{NP}$ then iO exists (but one-way functions do not). We present basic definitions of complexity theory and as well as definitions of complexity classes such as P, NP and $\mathsf{co\text{-}NP}$ in Appendix A.2 on page 298.

Proposition 5.6 ([BGI+12] Proposition 7.2.)**.** *Inefficient indistinguishability obfuscators exist. In particular, if* $\mathsf{P} = \mathsf{NP}$ *then (efficient) iO exists.*

Proof. We define obfuscator $\mathsf{O}(C)$ to search for the lexicographically first circuit (with respect to some fixed encoding) which has the same functionality as C. With the above inefficient obfuscator any two functionally equivalent circuits will be mapped to the same obfuscation and are hence indistinguishable. If $\mathsf{P} = \mathsf{NP}$ then we can efficiently search for the lexicographically first functionally equivalent circuit and hence in such a world efficient iO exists. □

Usually when we use indistinguishability obfuscation we also require one-way functions and by the above these are two separate assumptions. Komargodski et al. [KMN+14] study under which complexity theoretic assumptions iO implies one-way functions and show that this is the case if $\mathsf{NP} \not\subseteq \mathsf{io\text{-}BPP}$ (here io stands for *infinitely often* and should not be confused with indistinguishability obfuscator which we abbreviate with iO).

Statistically secure indistinguishability obfuscation. Throughout this thesis we only consider obfuscation which is secure with respect to computationally bounded distinguishers, that is, we restrict our distinguishers to be PPT.[9] One can also ask whether truly strong obfuscation exists which is also secure against computationally unbounded adversaries. Goldwasser and Rothblum show that such a strong form of obfuscation is unlikely as it implies a collapse of the polynomial hierarchy to its second level [GR14]. Their result is actually stronger as they show that even if statistical iO exists for the restricted class of 3-CNF formulas (a subclass of AC^0) this already implies a breakdown of the polynomial hierarchy. Furthermore, their result even holds if there is an imperfection in the obfuscator, that is, the obfuscator may, with negligible probability, output circuits that are different from the input circuit. In case we consider only perfect obfuscators (as defined in Definition 5.1) we can provide a simpler and in some sense stronger result. We can show that if statistical indistinguishability obfuscation exists for the class of all constant zero circuits (i.e., the circuit which outputs zero on every input) then $\mathsf{NP} = \mathsf{co\text{-}NP}$ and consequently the polynomial hierarchy collapses to its *first level*.[10] Note that while the obfuscator must be indistinguishable only for circuits that are zero on every input we still require the basic properties from Definition 5.1 for all circuits, that is, we require in particular that the obfuscator is functionality preserving for any input circuit.

[9] We note that we also restrict our samplers to be PPT but that this is not a common restriction as many definitions of obfuscation quantify over all circuits instead of considering explicit sample algorithms.

[10] The proof of Theorem 5.7 appeared in an unpublished manuscript prepared together with Christina Brzuska and Pooya Farshim [BFM13b].

Theorem 5.7. *If there is a perfect statistical indistinguishability obfuscator for the class of all constant zero circuits, then* $\mathsf{NP} = \mathsf{co\text{-}NP}$.

Proof. Let $\mathcal{L}$ be an NP-complete language and let $\overline{\mathcal{L}} = \{0,1\}^* \setminus \mathcal{L}$ denote the complement of $\mathcal{L}$. We show for a NP-complete language $\mathcal{L}'$ derived from $\mathcal{L}$, that if perfect statistical indistinguishability obfuscation exists then $\mathcal{L}' \in \mathsf{co\text{-}NP}$. This in turn implies that $\mathsf{NP} = \mathsf{co\text{-}NP}$ and hence the polynomial hierarchy collapses to its first level, that is, $\mathsf{PH} = \Sigma_1^p$ [Sto76]. (See Appendix A.2 for a definition of these complexity classes).

Let us first recall that if $\mathcal{L}$ is an NP-complete language then $\mathcal{L} \in \mathsf{co\text{-}NP}$ is equivalent to the statement: $\mathsf{NP} = \mathsf{co\text{-}NP}$. If $\mathsf{NP} = \mathsf{co\text{-}NP}$ then any NP-complete language is trivially in co-NP. For the other direction we show that for any language $\mathcal{A} \in \mathsf{NP}$ it holds that $\mathcal{A} \in \mathsf{co\text{-}NP}$ and conversely that if $\mathcal{A} \in \mathsf{co\text{-}NP}$ then also $\mathcal{A} \in \mathsf{NP}$. For this assume that $\mathcal{L}$ is NP-complete and that $\mathcal{L} \in \mathsf{co\text{-}NP}$. For any language $\mathcal{A}$ in NP there exists a polynomial time Karp reduction to $\mathcal{L}$ (i.e., $\mathcal{A} \leq_p \mathcal{L}$) since, by assumption, $\mathcal{L}$ is NP-complete. In particular, then also $\overline{\mathcal{A}} \leq_p \overline{\mathcal{L}}$ holds. Since, by assumption, $\mathcal{L} \in \mathsf{co\text{-}NP}$ we have that both $\overline{\mathcal{L}}$ and $\overline{\mathcal{A}}$ are in NP and thus $\mathcal{A} \in \mathsf{co\text{-}NP}$. If, on the other hand, $\mathcal{A} \in \mathsf{co\text{-}NP}$ then $\overline{\mathcal{A}} \in \mathsf{NP}$ which implies (via the previous argument) that $\overline{\mathcal{A}} \in \mathsf{co\text{-}NP}$ which in turn implies that $\mathcal{A} \in \mathsf{NP}$.

We can now turn to prove Theorem 5.7. Let $\mathcal{L}$ be an NP-complete language. In order to show that $\mathcal{L} \in \mathsf{co\text{-}NP}$ we can equivalently show that $\overline{\mathcal{L}} \in \mathsf{NP}$ which is equivalent to showing that there is a polynomial size witness of the fact that some string is not in the language $\mathcal{L}$.

Let us first define a related language $\mathcal{L}'$ which will allow for efficient generation of strings that are not in the language $\mathcal{L}'$. We define $\mathcal{L}'$ as

$$\mathcal{L}' := \{1 \| x \mid x \in \mathcal{L}\}$$

and note that since $\mathcal{L}$ is NP-complete so is $\mathcal{L}'$. Let circuit C_λ be the verifier for $\mathcal{L}'$ that checks whether a witness $w \in \{0,1\}^{\mathsf{poly}(\lambda)}$ is valid for an instance $x \in \{0,1\}^\lambda$, that is, for every $x \in \{0,1\}^\lambda$

$$x \in \mathcal{L}' \iff \exists w \in \{0,1\}^{\mathsf{poly}(\lambda)} : C_\lambda(x,w) = 1.$$

Let $x_{\mathsf{no}} \in \{0,1\}^*$ be such that $x_{\mathsf{no}} \notin \mathcal{L}'$. Then circuit $C_{|x_{\mathsf{no}}|}(x_{\mathsf{no}}, \cdot)$ implements the constant zero circuit since, by definition, there is no string which makes $C_{|x_{\mathsf{no}}|}(x_{\mathsf{no}}, \cdot)$ output 1. Furthermore, if a statistical indistinguishability obfuscator iO exists for the class of all constant zero circuits then the following distributions are statistically close

$$\mathsf{iO}(C_{|x_{\mathsf{no}}|}(x_{\mathsf{no}}, \cdot)) \approx_s \mathsf{iO}(C_{|x'|}(x', \cdot))$$

for any $x' \notin \mathcal{L}'$ with $|x'| = |x_{\mathsf{no}}|$. In particular, statistical closeness of the above distributions guarantees that for any $x' \notin \mathcal{L}'$ with $|x'| = |x_{\mathsf{no}}|$ there exists a "randomness pair" (R, R') of polynomial sized strings such that

$$\mathsf{iO}(C_{|x_{\mathsf{no}}|}(x_{\mathsf{no}}, \cdot); R) = \mathsf{iO}(C_{|x'|}(x', \cdot); R') \tag{5.2}$$

Note that if such a pair does not exist, then the support of the two distributions would be disjoint and, hence, the distributions would not be statistically close. Additionally, for any yes-instance $x_{\mathsf{yes}} \in \mathcal{L}'$

with $|x_{\mathsf{yes}}| = |x_{\mathsf{no}}|$ we know that the support of

$$\mathsf{iO}(C_{|x_{\mathsf{no}}|}(x_{\mathsf{no}}, \cdot)) \qquad \text{and} \qquad \mathsf{iO}(C_{|x_{\mathsf{yes}}|}(x_{\mathsf{yes}}, \cdot)) \tag{5.3}$$

are disjoint due to the perfect functionality preserving property of obfuscator iO. Taken together, Equations (5.2) and (5.3) allow us to conclude that a no-instance x_{no} together with a pair (R, R') as in Equation (5.2) witnesses the fact that a string x' is not in $\mathcal{L}'$. Hence, $\overline{\mathcal{L}'} \in \mathsf{NP}$ and as, by assumption, $\mathcal{L}' \in \mathsf{co\text{-}NP}$ we can conclude that $\mathsf{NP} = \mathsf{co\text{-}NP}$. □

5.5.3 Indistinguishability Obfuscation vs. Virtual Grey-Box Obfuscation

We have already hinted at indistinguishability obfuscation being strictly weaker than virtual grey-box obfuscation. In the following we make this statement formal. For this we present an alternative (but equivalent) definition of indistinguishability obfuscation given by Brakerski and Rothblum [BR14]. They show that if we relax the requirement of the simulator in the VGB definition to make at most polynomially many oracle queries and instead consider a truly unbounded simulator then this definition is equivalent to the definition of indistinguishability obfuscation given in Section 5.5.1.

Definition 5.7 (Alternative definition of indistinguishability obfuscation [BR14])**.** *A* PPT *obfuscation scheme* iO *for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *of poly-size circuits is an* indistinguishability obfuscator *for* $\mathcal{C}$ *if it satisfies:*

Unbounded simulation. *For any* PPT *adversary* $\mathcal{A}$ *there exists an unbounded simulator* Sim *(that runs in unbounded time and can make an unbounded number of oracle queries) such that for all sufficiently large* $\lambda \in \mathbb{N}$*, all* $C \in \mathcal{C}_\lambda$*, and all* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ *we have*

$$\left|\Pr\left[\mathcal{A}(1^\lambda, \mathsf{iO}(1^\lambda, C), \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^C(1^\lambda, 1^{|C|}, \mathsf{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,,$$

where the probability is taken over the coins of $\mathcal{A}$*,* Sim*, and* iO*.*

We now establish that this definition is indeed equivalent to the iO definition based on circuit samplers from Section 5.5.1.

Lemma 5.8 ([BR14] Lemma 2.9)**.** *Definition 5.7 is equivalent to the circuit sampler-based iO-Definition of Section 5.5.1 (Definitions 5.5 and 5.6).*

Proof. Let iO be an obfuscator according to the above simulator-based definition (Definition 5.7). Let $\mathsf{Sam} \in \mathcal{S}_{\mathbf{eq}}$ be an equality circuit sampler and let D be a distinguisher in the IO game (of Definition 5.5). Then for any $(C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)$, we know by the security of iO due to Definition 5.7 that there exists an unbounded simulator Sim such that

$$\left|\Pr\left[\mathsf{D}(1^\lambda, \mathsf{iO}(1^\lambda, C_0), \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^{C_0}(1^{|C_0|}, \mathsf{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

and, equally,

$$\left|\Pr\left[\mathsf{D}(1^\lambda, \mathsf{iO}(1^\lambda, C_1), \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^{C_1}(1^{|C_1|}, \mathsf{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

Note that with overwhelming probability over the coins of Sam circuits C_0 and C_1 are functionally equivalent and, thus,

$$\left|\Pr\left[\mathsf{Sim}^{C_0}(1^{|C_0|},\mathsf{aux})=1\right]-\Pr\left[\mathsf{Sim}^{C_1}(1^{|C_1|},\mathsf{aux})=1\right]\right|\leq \mathsf{negl}(\lambda)\,.$$

Applying the triangle inequality and noting that the sum of two negligible functions is negligible yields that

$$\left|\Pr\left[\mathsf{D}(1^\lambda,\mathsf{iO}(1^\lambda,C_0),\mathsf{aux})=1\right]-\Pr\left[\mathsf{D}(1^\lambda,\mathsf{iO}(1^\lambda,C_1),\mathsf{aux})=1\right]\right|\leq \mathsf{negl}(\lambda)\,.$$

The reverse direction follows from the fact that inefficient iO exists. Formally, let $\mathcal{A}$ be an efficient adversary in Definition 5.7 and let C be a circuit. We construct an unbounded simulator Sim as follows: on input $(1^{|C|},\mathsf{aux})$ the simulator queries its oracle on all possible inputs reconstructing the truth table for C. It then searches for the lexicographically first circuit description C' which agrees with the constructed truth table and that is of size $|C|$. It constructs an obfuscation $\overline{C}\leftarrow\!\!\$\,\mathsf{iO}(C')$ and, finally, runs adversary $\mathcal{A}$ on input $(\overline{C},\mathsf{aux})$ returning whatever $\mathcal{A}$ returns. By construction of Sim, we have that

$$\left|\Pr\left[\mathcal{A}(1^\lambda,\mathsf{iO}(1^\lambda,C),\mathsf{aux})=1\right]-\Pr\left[\mathsf{Sim}^{C}(1^{|C|},\mathsf{aux})=1\right]\right|$$

and

$$\left|\Pr\left[\mathcal{A}(1^\lambda,\mathsf{iO}(1^\lambda,C),\mathsf{aux})=1\right]-\Pr\left[\mathcal{A}(1^\lambda,\mathsf{iO}(1^\lambda,C'),\mathsf{aux})=1\right]\right| \tag{5.4}$$

are equivalent. By the security of iO we have that the difference in equation (5.4) must be negligible as otherwise there exists a (non-uniform) equality sampler Sam which outputs (C,C',aux), which together with (efficient) adversary $\mathcal{A}$, has a non-negligible advantage in game IO of Definition 5.5. □

Using the alternative definition of indistinguishability obfuscation it is easy to see that VGB is stronger, or more accurately, at least as strong as iO as for VGB the simulator is more restricted. This yields the following hierarchy of general-purpose obfuscators

Proposition 5.9. *We have the following hierarchies of obfuscators:*

$$\mathit{VBB} \implies \mathit{VGB} \implies \mathit{iO}$$

Here $A \implies B$ denotes that any obfuscator which achieves A also achieves B.

5.5.4 Indistinguishability Obfuscation is Best-Possible Obfuscation

We have seen two equivalent definitions of indistinguishability obfuscation. A third characterization of iO due to Goldwasser and Rothblum (GR) considers obfuscation from the perspective of what can be learned from a circuit [GR14]. GR introduce and study a notion of obfuscation they call *best-possible* obfuscation. The naming of *best-possible* refers to what sort of obfuscation can be achieved for a given circuit. Intuitively, an obfuscator is a best-possible obfuscator if an obfuscation leaks as little information about the original program as any functionally equivalent circuit and thus, in particular, it leaks as little as an obfuscation with the best obfuscator would leak. Formally:

Definition 5.8 (Best-possible obfuscation [GR14])**.** *A* PPT *obfuscation scheme* O *for an ensemble* $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ *of poly-size circuits is a* best-possible obfuscator *for* $\mathcal{C}$ *if it satisfies:*

- Computational best-possible obfuscation. *For any polynomial-size learner* L*, there exists a polynomial size simulator* Sim *such that for every auxiliary information* $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$ *and any two circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *that compute the same function and satisfy* $|C_0| = |C_1|$ *it holds for any* PPT *distinguisher* D *that*

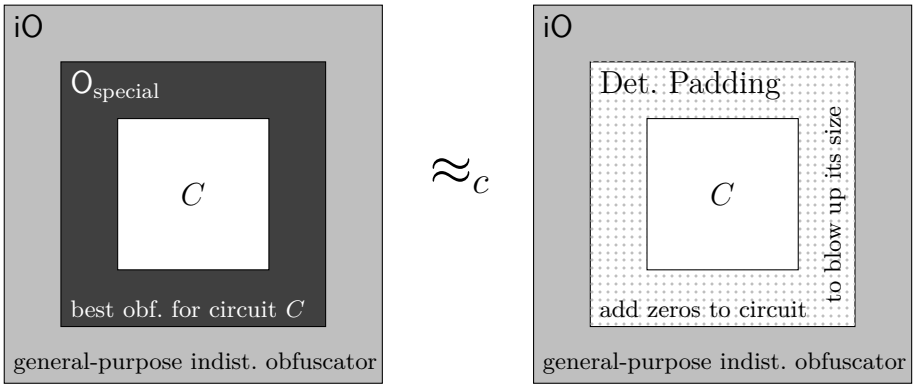
$$\left|\Pr\left[\mathsf{D}(1^\lambda, \mathsf{L}(\mathsf{aux}, \mathsf{O}(C_0))) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, \mathsf{Sim}(\mathsf{aux}, C_1)) = 1\right]\right| \leq \mathsf{negl}(\lambda)\,.$$

  Here the probability is over the coins of the obfuscator, learner and distinguisher or the coins of the simulator and distinguisher.

Goldwasser and Rothblum show that the notions of best-possible obfuscation and indistinguishability obfuscation are equivalent [GR14].[11] We here recall their result and refer to [GR14] for a formal proof.

Proposition 5.10 ([GR14] Propositions 3.4 and 3.5)**.** *Definition 5.8 is equivalent to the circuit sampler-based Definition of Section 5.5.1 (Definitions 5.5 and 5.6).*

The idea is best explained pictorially. Consider a circuit C and some special-purpose obfuscator $\mathsf{O}_{\text{special}}$ for C as well as a general purpose indistinguishability obfuscator iO. On the left (in the following picture) we consider the case where we first apply the special obfuscator $\mathsf{O}_{\text{special}}$ to circuit C and on top of that apply the indistinguishability obfuscator. Assuming that the special-purpose obfuscator is good (for example, VBB) the indistinguishability obfuscator cannot remove any of the protection offered by $\mathsf{O}_{\text{special}}$. In other words, if something cannot be learned from $\mathsf{O}_{\text{special}}(C)$ then this cannot be learned from $\mathsf{iO}(\mathsf{O}_{\text{special}}(C))$ and, thus, intuitively, if $\mathsf{O}_{\text{special}}(C)$ is a good obfuscation then so is $\mathsf{iO}(\mathsf{O}_{\text{special}}(C))$.



On the right, we consider the case where we take circuit C and first increase its size with a deterministic dummy padding (for example we add a sequence of pairs of NOT gates to the first input wire). Let us call the result $\mathsf{PAD}(C)$. Padding does not change the functionality, but using padding we can make the padded circuit as big as the output of the special obfuscator, that is, we

[11] This is only true if we consider efficient obfuscators [GR14]. In this thesis we only consider efficient obfuscation schemes and hence for us best-possible and indistinguishability obfuscation are equivalent.

choose a padding such that $|\mathsf{PAD}(C)| = |\mathsf{O}_{\text{special}}(C)|$. Then, however, by the security guarantee of the indistinguishability obfuscator we know that the distributions

$$\mathsf{iO}(\mathsf{O}_{\text{special}}(C)) \approx_c \mathsf{iO}(\mathsf{PAD}(C))$$

are computationally indistinguishable and, hence, iO must be as good as $\mathsf{O}_{\text{special}}$ (on the padded circuit).

Remark. It may seem that the deterministic padding in the above argument is an artifact of the argument strategy and not really necessary. As it turns out, padding plays a crucial role in the design of cryptosystems based on indistinguishability obfuscation which we discuss in detail in Chapter 13.

5.5.5 Using Indistinguishability Obfuscation

One question that we have not answered so far is how indistinguishability obfuscation can be used in order to construct or attack cryptographic constructions. In this section, we present the two main techniques that we will use in various combinations later in this thesis. The first technique which we call the *zero-circuit technique* allows to argue that an obfuscated circuit hides certain values (e.g., a secret key). This will usually be the technique of choice when attacking a cryptographic scheme but may also be used for constructive purposes. The technique was first introduced by Garg et al. [GGH+13b] to motivate the usefulness of indistinguishability obfuscation in their seminal paper in which they presented the first candidate construction for iO. There, Garg et al. use the zero-circuit technique to present a very simple construction of *witness encryption*, a primitive that allows to encrypt messages relative to an instance of a NP-language [GGSW13].

The second technique that we use and further refine in order to obtain our positive results is the *punctured programs technique* due to Sahai and Waters [SW14]. Here, the basic idea is that certain pseudorandom functions allow for the generation of specialized keys that allow to evaluate the PRF on every point except for one (or a polynomial-sized set) which we call the *punctured point*. The security of such a *punctured PRF* guarantees that if an adversary is given such a specialized key the true value at the punctured point is indistinguishable from a uniformly random value. In combination with obfuscation this allows us to program the obfuscated circuit to, for example, embed challenges or to make the circuit consistent with the view of an adversary.

Finally, we briefly discuss the *two-keys technique* which has been used to bootstrap indistinguishability obfuscation from allowing obfuscation only for circuits in NC^1 to circuits in P/poly [GGH+13b]. (See Appendix A.2 for a definition of these complexity classes).

The Zero-Circuit Technique

Consider a function $f : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ which is length doubling and consider the following circuit $C[r, \mathsf{secret}]$ which has a value $r \in \{0,1\}^{2\lambda}$ and a secret value secret hard-coded.

$C[r, \mathsf{secret}](x)$

if $f(x) = r$ **then**
  return secret
return $\bot$

If we consider an obfuscation $\mathsf{iO}(C[r, \mathsf{secret}])$ of this circuit, does this allow an adversary to learn the secret value secret? The answer is, that it depends on how we choose r and function f. Assume we fix f to be any efficiently computable function and then choose r uniformly at random in $\{0,1\}^{2\lambda}$. Then, with overwhelming probability (of at least $1 - 2^{-\lambda}$) we have that r is not in the image of f and, thus, circuit $C[r, \mathsf{secret}](\cdot)$ outputs $\perp$ on every input. This, however, means that $C[r, \mathsf{secret}](\cdot)$ is functionally equivalent to the constant zero circuit Z which also outputs $\perp$ on every input but which does not have value secret hard-coded. Hence, by the security of the indistinguishability obfuscator we know that, for a uniformly random r,

$$\mathsf{iO}(C[r, \mathsf{secret}]) \approx_c \mathsf{iO}(\mathsf{Z}). \tag{5.5}$$

As Z does not contain value secret (even information-theoretically) we can conclude that $\mathsf{iO}(C[r, \mathsf{secret}])$ hides secret in case r is not in the image of f.

Choosing a function f and a value r not in the image of f seems to be rather besides the point as functionality-wise we would like to be able to also argue that $\mathsf{iO}(C[r, \mathsf{secret}])$ hides secret if r is in the image of f. Thus, in order to be able to use the above technique we combine it with a simple cryptographic primitive: a pseudorandom generator. Let us replace f by a pseudorandom generator $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{2\lambda}$ with stretch factor 2. By the security of PRG we have that the distributions

$$\mathsf{PRG}(s) \approx_c r$$

are computationally indistinguishable where $s \in \{0,1\}^{\lambda}$ and $r \in \{0,1\}^{2\lambda}$ are uniformly random. Thus, if we choose a random seed $s \leftarrow\!\!\$\, \{0,1\}^{\lambda}$ and then obfuscate circuit $C[\mathsf{PRG}(s), \mathsf{secret}]$ (the circuit contains the result of the computation $\mathsf{PRG}(s)$) we can again argue that it hides secret. For this note that by the security of the PRG the circuits

$$C[\mathsf{PRG}(s), \mathsf{secret}] \approx_c C[r, \mathsf{secret}] \tag{5.6}$$

are computationally indistinguishable where, again, $s \in \{0,1\}^{\lambda}$ and $r \in \{0,1\}^{2\lambda}$ are uniformly random values. Combining Equations (5.5) and (5.6) yields that $\mathsf{iO}(C[\mathsf{PRG}(s), \mathsf{secret}])$ hides value secret. Note that what we have constructed is a symmetric encryption scheme as knowing value s allows to recover secret from $\mathsf{iO}(C[\mathsf{PRG}(s), \mathsf{secret}])$ while we can argue that if the "secret key" s is hidden, then $\mathsf{iO}(C[\mathsf{PRG}(s), \mathsf{secret}])$ computationally hides any information about secret.

Extension to public-key encryption. We can easily bootstrap the above idea to obtain a public-key encryption scheme. For this the secret key is chosen as a uniformly random value $s \leftarrow\!\!\$\, \{0,1\}^{\lambda}$ and we define the public key as $\mathsf{pk} \leftarrow \mathsf{PRG}(s)$ (for some pseudorandom generator with at least super-logarithmic stretch). We can now define the encryption of a message $m \in \{0,1\}^*$ (which can be of arbitrary length) as follows: to encrypt message m one simply obfuscates the special circuit $C_{\mathsf{Enc}}[m, \mathsf{pk}]$ that has message m and public key pk hard-coded. Circuit $C_{\mathsf{Enc}}[m, \mathsf{pk}]$ on input a value s' checks if $\mathsf{PRG}(s') = \mathsf{pk}$ and, if so, it outputs m and otherwise it outputs $\perp$. In order to decrypt, one runs the ciphertext (which is itself a circuit) on secret key s. Formally, we describe the scheme as follows:

$\underline{\mathsf{PKE.KGen}(1^\lambda)}$

1 : $\mathsf{sk} \leftarrow\!\!\$\ \{0,1\}^\lambda$
2 : $\mathsf{pk} \leftarrow \mathsf{PRG}(\mathsf{sk})$
3 : **return** $(\mathsf{sk}, \mathsf{pk})$

$\underline{\mathsf{PKE.Enc}(\overline{C}_{\mathsf{pk}}, m)}$

1 : $C_{\mathsf{Enc}}[m, \mathsf{pk}] \leftarrow$ $\underline{C_{\mathsf{Enc}}[m, \mathsf{pk}](s')}$
if $\mathsf{PRG}(s') = \mathsf{pk}$ **then**
return m
return $\bot$
2 : $\overline{C}_{\mathsf{Enc}} \leftarrow\!\!\$\ \mathsf{iO}(C_{\mathsf{Enc}}[m, \mathsf{pk}])$
3 : **return** $\overline{C}_{\mathsf{Enc}}$

$\underline{\mathsf{PKE.Dec}(\mathsf{sk}, \overline{C}_{\mathsf{Enc}})}$

1 : $m \leftarrow \overline{C}_{\mathsf{Enc}}(\mathsf{sk})$
2 : **return** m

The assignments in line 1 of the encryption operation denotes the construction of the (unobfuscated) circuit. To prove the scheme secure (for example IND-CPA) one replaces, in a first step, the public key pk that is given to the adversary by a uniformly random value of the same size. This can be shown to be indistinguishable down to the security of the PRG. Then, in a second step one replaces circuit C_{Enc} by the constant zero circuit, which can be reduced down to iO.

Padding the zero circuit. One detail that we neglected in the above argument is the size of our circuits. Indistinguishability obfuscation only guarantees that the distributions

$$\mathsf{iO}(C_0) \quad \text{and} \quad \mathsf{iO}(C_1)$$

are computationally indistinguishable if the circuits are functionally equivalent **and** have the same size, i.e., we need to ensure that

$$|C_0| = |C_1| .$$

For this we remark that, given a circuit C, we can define an efficient and deterministic padding scheme $\mathsf{PAD}(\mathsf{s}, C)$ which takes as input a circuit C and an integer s and outputs a functionally equivalent circuit of length $\max(|C|, \mathsf{s})$. Note that the size of a circuit is the number of its gates (we consider only Boolean circuits with $\neg$ (NOT), $\wedge$ (AND), and $\vee$ (OR) gates) plus the number of input and output nodes. As an $\wedge$-gate where both inputs are connected to the same wire is functionality preserving—note that $1 \wedge 1 = 1$ and $0 \wedge 0 = 0$—our deterministic padding scheme can simply add such $\wedge$-gates to the first input wire. Further, note that with the above technique we can easily generate zero circuits of any length and input size.

The Punctured Programs Technique

A main ingredient in many of our constructions are so-called *puncturable pseudorandom functions* (puncturable PRF or pPRF) [SW14]. Puncturable PRFs are a generalization of constrained PRFs which were recently independently developed by three groups [BW13, BGI14, KPTZ13]. Intuitively a puncturable PRF is pseudorandom function that allows for the generation of special "punctured" keys k_S which in turn can be used to evaluate the PRF on every input except on inputs in set S. Combined with obfuscation this allows, for example, to embed challenges into obfuscated circuits without an adversary being able to note a difference. We first formally define puncturable PRFs and then discuss how these can be used in the context of obfuscation.

Defining puncturable PRFs. A family of puncturable pseudorandom functions $\mathsf{F} := (\mathsf{F.KGen}, \mathsf{F.Puncture}, \mathsf{F.Eval}, \mathsf{F.kl}, \mathsf{F.il}, \mathsf{F.ol})$ consists of functions to generate keys, evaluate the function, and

functions describing the key- and input- and output-length (see also Section 2.2.3). The additional PPT puncturing algorithm $\mathsf{F.Puncture}$ on input a key k and a polynomial-size set $S \subseteq \{0,1\}^{\mathsf{F.il}(\lambda)}$ outputs a special key k_S. We present a game-based variant of selectively secure puncturable PRFs where the adversary may adaptively request challenge points (via oracle CHALLENGE) while it is only allowed to see a single punctured key.[12] Via a standard hybrid argument it can be shown that this formulation is equivalent to allowing only a single challenge query [BW13].

Definition 5.9. *A family of functions* $\mathsf{F} := (\mathsf{F.KGen}, \mathsf{F.Puncture}, \mathsf{F.Eval}, \mathsf{F.kl}, \mathsf{F.il}, \mathsf{F.ol})$ *is called* puncturable pseudorandom function *(puncturable PRF or pPRF) if it has the following properties:*

FUNCTIONALITY PRESERVED UNDER PUNCTURING. *For every* PPT *adversary* $\mathcal{A}$ *such that* $\mathcal{A}(1^\lambda)$ *outputs a polynomial-size set* $S \subseteq \{0,1\}^{\mathsf{F.il}(\lambda)}$*, it holds for all* $x \in \{0,1\}^{\mathsf{F.il}(\lambda)}$ *where* $x \notin S$ *that:*

$$\Pr\left[\mathsf{F.Eval}(\mathsf{k}, x) = \mathsf{F.Eval}(\mathsf{k}_S, x) : \mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda), \mathsf{k}_S \leftarrow\!\!\$\, \mathsf{F.Puncture}(\mathsf{k}, S)\right] = 1$$

PSEUDORANDOM AT PUNCTURED POINTS. *For every* PPT *adversary* $(\mathcal{A}_1, \mathcal{A}_2)$ *the advantage* $\mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F},\mathcal{A}_1,\mathcal{A}_2}(\cdot)$ *defined as*

$$\mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F},\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 2 \cdot \Pr\left[\mathsf{pPRF}^{\mathsf{F}}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - 1$$

is negligible, where game pPRF *is defined as*

$\mathsf{pPRF}^{\mathsf{F}}_{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$

$S \leftarrow \{\}; b \leftarrow\!\!\$\, \{0,1\}$
$\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$
$\mathsf{state} \leftarrow\!\!\$\, \mathcal{A}_1^{\text{CHALLENGE}}(1^\lambda)$
$\mathsf{k}^* \leftarrow\!\!\$\, \mathsf{F.Puncture}(\mathsf{k}, S)$
$b' \leftarrow\!\!\$\, \mathcal{A}_2(1^\lambda, \mathsf{state}, \mathsf{k}^*)$
return $(b = b')$

CHALLENGE(x)

if $x \in S$ **then return** $\perp$
$S \leftarrow S \cup \{x\}$
if $b = 0$ **then**
  $y \leftarrow \mathsf{F.Eval}(\mathsf{k}, x)$
else $y \leftarrow\!\!\$\, \{0,1\}^{\mathsf{F.ol}(\lambda)}$
return y

As observed by [BW13, BGI14, KPTZ13] puncturable PRFs can, for example, be constructed from pseudorandom generators via the GGM tree-based construction [GGM84]. For this recall that the PRF key in the GGM construction consists of a single random seed k and the the construction uses a pseudorandom generator that is length doubling to build a binary tree: key k makes up the root node and the two child nodes of a node are generated by applying the PRG to the node's value and identifying the first half of the output with the first child node and the second half with the second child node. If we identify the first child of a node with 0 and the second with 1 then we can evaluate the PRF on a value x bit by bit by starting at the root and choosing a path to a leaf by choosing child $x[j]$ on the j-th level (with the root being on level 1). The leaf value yields the PRF value. Now, given a node value somewhere within the tree allows to compute PRF values for a subset of inputs which can be used for the construction of punctured keys. A punctured key, thus, does not consist of the *root key* but of several node values within the tree that allow to evaluate any input except for the inputs in the punctured set.

[12] Adaptive security refers to adversaries that may adaptively ask for punctured keys. In the selective variant that we consider the adversary instead only sees a single key.

The punctured programs technique. Sahai and Waters developed the punctured programs technique which combines puncturable PRFs with indistinguishability obfuscation and use it to build various cryptographic primitives [SW14].

Let F be a puncturable PRF. The punctured programs technique usually consists of three steps. One starts with the obfuscation of the puncturable PRF (or a circuit containing the PRF) with a hard-coded key $\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$. Then, in a second step a specific input value is "branched out" from the code such that it can be treated separately. That is, instead of considering a circuit that computes $\mathsf{F.Eval}(\mathsf{k}, \cdot)$ we consider a circuit

$C[\mathsf{k}_\tau, \tau, y]$

if $x = \tau$ **then**

  return y

return $\mathsf{F.Eval}(\mathsf{k}_\tau, x)$

where τ is branched out, y is computed as $y \leftarrow \mathsf{F.Eval}(\mathsf{k}, \tau)$ (so as to not change the functionality) and k_τ denotes the punctured key, that is, $\mathsf{k}_\tau \leftarrow\!\!\$\, \mathsf{F.Puncture}(\mathsf{k}, \tau)$. As it does not change the program, this step is undetectable under obfuscation. Finally, in the third step, the security of the puncturable PRF allows to change the value returned at the target value into a uniformly random value. We visualize the three steps in Figure 5.1.

We will see various applications of the punctured programs technique throughout Part III of this thesis and will not replicate a complete example at this point. For further reading we refer to [SW14].

Padding. We have already mentioned the importance of padding when working with indistinguishability obfuscation. Indeed, padding is a crucial part when working with the punctured programs technique. Again consider the above circuits. Without additional padding we have that

$$|C_0[\mathsf{k}]| \neq |C_1[\mathsf{k}_\tau, \tau, y \leftarrow \mathsf{F.Eval}(\mathsf{k}, \tau)]|\,.$$

This is for two reasons. Unless the unpunctured key is not encoded compactly, punctured keys must be larger since the punctured points must be encoded. The GGM construction [GGM84] provides a good intuition as an unpunctured key is simply a single PRG seed, while a punctured key contains multiple seeds [BW13, BGI14, KPTZ13]. Furthermore, the encoding of the extra branch and the hard-coded value y need additional gates. Thus, when working with the punctured programs technique one needs to pad the initial circuit sufficiently to allow for later changes. We will see that this is not without consequences. In particular, many of our constructions will be secure only against adversaries that make at most q many queries, where q is an arbitrary polynomial which is given to the key generation algorithm. Ideally, one would like to remove this dependency on q. We discuss padding in detail in Chapter 13.

The Two-Keys Technique

For completeness we also briefly discuss a third technique used in the context of indistinguishability obfuscation.[13] The two-keys technique has its roots in the two-keys paradigm of Naor and

[13] Proofs using iO can usually be categorized into using one or more of the three presented techniques.

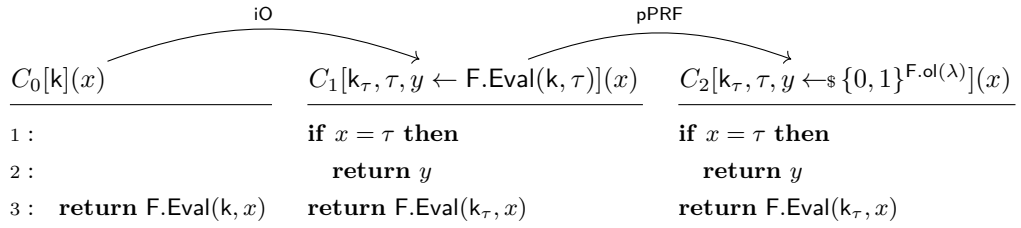


Figure 5.1: Visualization of the punctured programs technique. From circuit C_0 to C_1 we have branched out value τ. On input $x = \tau$ the circuit returns the hard-coded value y which is set to $\mathsf{F.Eval}(\mathsf{k}, \tau)$ in order for the functionality not to change. That is, circuits C_0 and C_1 compute the same function. In a next step the functionality is changed such that C_2 now returns a uniformly random value on input $x = \tau$. By the security of indistinguishability obfuscation we know that under obfuscation the first two circuits are computationally indistinguishable and the security of the puncturable PRF tells us that the distributions of circuits C_1 and C_2 (even without obfuscation) are computationally indistinguishable.

Yung [NY90] who showed how to construct IND-CCA secure public-key encryption where the public key consists of two keys.

Consider a public-key encryption scheme PKE and two key pairs $(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$ and $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$. Now consider circuits $C_0[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0]$ and $C_1[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1]$ which have both public keys hard-coded as well as one of the secret keys:

$C_0[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0](c_0, c_1, \pi)$	$C_1[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1](c_0, c_1, \pi)$
if proof π correct **then**	**if** proof π correct **then**
return $\mathsf{PKE.Dec}(\mathsf{sk}_0, c_0)$	**return** $\mathsf{PKE.Dec}(\mathsf{sk}_1, c_1)$
return $\perp$	**return** $\perp$

The circuits take as input two ciphertexts (c_0, c_1) and some (non-interactive) proof π proving that c_0 and c_1 are encryptions of the same message, c_0 with public key pk_0 and c_1 with public key pk_1. Assuming that the proof is perfectly sound,[14] we have that the two circuits are indeed functionally equivalent since if π is a correct proof (and the underlying PKE scheme is correct) then both circuits $C_0[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0]$ and $C_1[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1]$ properly decrypt an input (c_0, c_1, π) and if π is invalid both circuits always output $\perp$. Now, since C_0 (computationally) hides secret key sk_1 and C_1 (computationally) hides sk_0 and since

$$\mathsf{iO}(C_0[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_0]) \approx_c \mathsf{iO}(C_1[\mathsf{pk}_0, \mathsf{pk}_1, \mathsf{sk}_1])$$

we can argue that an indistinguishability obfuscation of C_0 (and similarly an obfuscation of C_1) hides both secret keys.

This technique has been used by Garg et al. to bootstrap their obfuscation candidate from covering circuits in NC^1 to all circuits in P/poly [GGH+13b]. We will briefly discuss their bootstrapping technique in Section 5.8 where we discuss the intuition behind existing candidate obfuscation schemes. We refer to [GGH+13b] for further details.

[14] A proof system has perfect soundness if no (even unbounded) adversary can convince an honest verifier of a false statement.

5.6 Differing-Inputs Obfuscation

A natural strengthening of indistinguishability obfuscation is the notion of *differing-inputs obfuscation* that recently also gained much attention [ABG+13, BCP14, BP13]. Differing-inputs obfuscation generalizes iO in that it allows also samplers to output circuits that differ on some inputs. The idea is that, if it is difficult to find inputs on which two circuits differ, then their obfuscations should be indistinguishable. As an example, consider the circuits $C_b[\mathsf{vk}]$ for $b \in \{0,1\}$ which have the verification key vk of a signature scheme hard-coded (see Appendix A.1 for a formal definition of signature schemes):

$C_b[\mathsf{vk}](m,\sigma)$

if $\mathsf{S.Vf}(\mathsf{vk}, m, \sigma) = 1$ **then**

  return b

return $\bot$

The two circuits $C_0[\mathsf{vk}]$ and $C_1[\mathsf{vk}]$ will always output $\bot$ unless they are run on a valid message-signature pair. If the secret key of the corresponding signature scheme is not known it is thus difficult to find a *differing input* as such an input is equivalent to a forgery for the signature scheme. The security of a differing-inputs obfuscation scheme diO states that the obfuscations of $\mathsf{diO}(C_0)$ and $\mathsf{diO}(C_1)$ are computationally indistinguishable.

Defining differing-inputs obfuscation. Formally, differing-inputs obfuscation is defined analogously to indistinguishability obfuscation (see Definitions 5.5 and 5.6) by relaxing the class of admissible samplers to so-called *differing-inputs circuit samplers*.

Definition 5.10 (Differing-inputs circuit sampler)**.** *Let* Sam *be a circuit sampler. We call* Sam *a* differing-inputs circuit sampler *if the advantage* $\mathsf{Adv}^{\mathrm{diff}}_{\mathsf{Sam},\mathsf{Ext}}(\cdot)$ *is negligible for all* PPT *algorithms* Ext *where the advantage is defined as (relative to game* Diff *on the right):*

$$\mathsf{Adv}^{\mathrm{diff}}_{\mathsf{Sam},\mathsf{Ext}}(\lambda) := \Pr\left[\mathsf{Diff}^{\mathsf{Ext}}_{\mathsf{Sam}}(\lambda) = 1\right]$$

$\mathsf{Diff}^{\mathsf{Ext}}_{\mathsf{Sam}}(\lambda)$

$(C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$

$x \leftarrow\!\!\$\ \mathsf{Ext}(1^\lambda, C_0, C_1, \mathsf{aux})$

return $(C_0(x) \neq C_1(x))$

We capture differing-inputs obfuscation by considering obfuscators O that are secure for the class of all (not necessarily efficient) differing-inputs samplers $\mathcal{S}_{\mathbf{diff}}$.

5.6.1 On the Plausibility of Differing-Inputs Obfuscation

Differing-inputs obfuscation is a strictly stronger primitive than indistinguishability obfuscation since it allows for a larger class of circuit samplers: $\mathcal{S}_{\mathbf{eq}} \subseteq \mathcal{S}_{\mathbf{diff}}$. As any diO secure obfuscator is necessarily also an indistinguishability obfuscator we can, similarly to Proposition 5.9 where we related the VGB obfuscation and iO, present a second hierarchy of obfuscators:

Proposition 5.11. *We have the following hierarchies of obfuscators:*

$$\mathit{VBB} \implies \mathit{diO} \implies \mathit{iO}$$

As before, $A \implies B$ denotes that any obfuscator which achieves A also achieves B.

Interestingly, we do not know anything about the relationship between VGB and diO. That is, it is still open whether any of them implies the other or whether they are incomparable notions of obfuscation.

In case $\mathsf{NP} = \mathsf{P}$ then given any two circuits C_0 and C_1 for which there exists an x such that $C_0(x) \neq C_1(x)$ we can efficiently find such an x. Consequently, if $\mathsf{NP} = \mathsf{P}$ then $\mathcal{S}_{\mathbf{eq}} = \mathcal{S}_{\mathbf{diff}}$ and differing-inputs obfuscation and indistinguishability obfuscation are equivalent and thus

Corollary 5.12. *If $\mathsf{P} = \mathsf{NP}$ then (efficient) diO exists.*

There is, however, also a second (and more useful; in constructive terms) case in which differing-inputs obfuscation and indistinguishability obfuscation are equivalent. Boyle, Chung, and Pass [BCP14] show that any general purpose indistinguishability obfuscator is also a differing-inputs obfuscator for circuits that differ only on a few (at most polynomially many) inputs. We heavily rely on their result for our positive constructions (see Chapters 11 and 14).

Theorem 5.13 ([BCP14] Theorem 12)**.** *Let iO be an indistinguishability obfuscator for all circuits in P/poly. Let Sam be a differing-inputs circuit sampler for which there exists a polynomial $d : \mathbb{N} \to \mathbb{N}$, such that*

$$\Pr\left[|\{x : C_0(x) \neq C_1(x)\}| \leq d(\lambda) \,:\, (C_0, C_1, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)\right] \geq 1 - \mathsf{negl}(\lambda)\,.$$

Then iO is a differing-inputs obfuscator for Sam, i.e., obfuscator iO is $\{\mathsf{Sam}\}$-secure.

As intuition for the result of Boyle et al. consider the extreme case in which two circuits C_0 and C_1 differ on exactly one point τ. Now, if there exists a distinguisher D that distinguishes obfuscations of the two circuits we can construct an extractor for point τ as follows. Consider all possible inputs x to circuits C_0 and C_1 on a one-dimensional axis. Now we can identify with each point x on the axis a circuit C^x_{mid} which on input $y < x$ evaluates circuit C_0 and on inputs $y \geq x$ evaluates C_1. We thus have that $C^0_{\mathrm{mid}} = C_0$ and $C^{2^\lambda}_{\mathrm{mid}} = C_1$ if we consider inputs to be in $\{0,1\}^\lambda$. Now, if we consider the circuit $C^{2^{\lambda-1}}_{\mathrm{mid}}$ that lies exactly in between C_0 and C_1 then we know $C^{2^{\lambda-1}}_{\mathrm{mid}}$ must be equivalent to either C_0 or to C_1 as target point τ is either smaller than $2^{\lambda-1}$ or greater or equal. Assuming it is greater or equal we have that $C^{2^{\lambda-1}}_{\mathrm{mid}}$ is functionally equivalent to C_0. The idea is that this can be tested using the distinguisher as the distinguisher must have negligible advantage in distinguishing obfuscations of C_0 from $C^{2^{\lambda-1}}_{\mathrm{mid}}$ but non-negligible advantage in distinguishing obfuscations of $C^{2^{\lambda-1}}_{\mathrm{mid}}$ and C_1. Repeating this process recursively allows to recover point τ bit by bit and Boyle et al. [BCP14] show that this can similarly be made to work if the circuits differ not on a single but on polynomially many points.

5.6.2 On the Implausibility of Differing-Inputs Obfuscation

Beside the above connections between diO and iO it is generally conjectured that full diO does not exist based on a conditional impossibility result of Garg et al. [GGHW14]. Garg et al. show that if one can obfuscate a fixed program that contains the secret key of a signature scheme with a special-purpose obfuscator in such a way that the program does not allow to forge a signature then

there cannot be general purpose diO with auxiliary information. We present their "implausibility" result next.

In order to attack diO we must specify a differing-inputs sampler together with a distinguisher that wins with noticeable probability in the IO game (see Definitions 5.5 and 5.10). However, we need to be careful since by the above connection between indistinguishability obfuscation and differing-inputs obfuscation for circuits that only differ on a few (at most polynomially many) points we need to come up with a differing-inputs sampler which samples circuits that differ on more than polynomially many inputs as otherwise any impossibility result would directly also imply an impossibility result for indistinguishability obfuscation which, given all the recent candidate constructions, would be highly surprising.

The two circuits that sampler Sam will output are going to be identical and output 0 everywhere unless they are presented with a valid signature for a fixed public verification key vk of a signature scheme S. On input a message and matching signature, each will output a hard-coded bit b which allows to distinguish the two circuits. More formally, the circuits can be defined as (having bit $b \in \{0,1\}$ and verification key vk hard-coded):

$C_b[\mathsf{vk}](m,\sigma)$

if $\mathsf{S.Vf}(\mathsf{vk}, m, \sigma) = 1$ **then**

  return b

return 0

Next, we need a way to distinguish between C_0 and C_1. For this it would be sufficient to leak the secret signing key or even a single message signature pair as auxiliary information but then our sampler would not be differing-inputs. Instead, our sampler leaks the obfuscation of a special "breaking circuit" C_{break} which has the signing key sk hard-coded. In addition, the breaking circuit makes use of a collision resistant hash function H. We define C_{break} as:

$C_{\text{break}}[\mathsf{H}, \mathsf{sk}](C)$

$m \leftarrow \mathsf{H}(C)$

$\sigma \leftarrow \mathsf{S.Sign}(\mathsf{sk}, m)$

$b \leftarrow C(m,\sigma)[1]$ // return first bit of result

return b

The breaker circuit takes as input a circuit C which has an output length of one bit. It runs it through hash function H to obtain a message m which it signs (using the hard-coded signing key sk) to obtain σ. It then runs C on (m,σ) and outputs the bit returned by this computation.

Now, we can define our sampler Sam to sample a fresh key pair $(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{S.KGen}(1^\lambda)$, to sample a hash function $\mathsf{H} \leftarrow\!\!\$\ \{\mathsf{H}\}_\lambda$, and to construct circuits $C_0[\mathsf{vk}]$ and $C_1[\mathsf{vk}]$ as well as $C_{\text{break}}[\mathsf{H}, \mathsf{sk}]$. It obfuscates $C_{\text{break}}[\mathsf{H}, \mathsf{sk}]$ with a special-purpose obfuscator to get $\overline{C}_{\text{break}}$ and outputs $(C_0[\mathsf{vk}], C_1[\mathsf{vk}], \overline{C}_{\text{break}})$. Once more in pseudocode:

$\mathsf{Sam}(1^\lambda)$

$(\mathsf{vk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{S.KGen}(1^\lambda)$

$\mathsf{H} \leftarrow\!\!\$\ \{\mathsf{H}\}_\lambda$

$\overline{C}_{\text{break}} \leftarrow\!\!\$\ \mathsf{O}_{\text{special}}(C_{\text{break}}[\mathsf{H}, \mathsf{sk}])$

return $(C_0[\mathsf{vk}], C_1[\mathsf{vk}], \overline{C}_{\text{break}})$

A distinguisher that gets as input an obfuscation $\overline{C}$ of either $C_0[\mathsf{vk}]$ or of $C_1[\mathsf{vk}]$ can easily distinguish when given access to $\overline{C}_{\text{break}}$ by evaluating $\overline{C}_{\text{break}}(\overline{C})$. However, it is not clear whether the above sampler is differing-inputs. Proving this requires to show that no efficient adversary that is given as input the verification key vk and an obfuscation $\mathsf{O}_{\text{special}}(C_{\text{break}}[\mathsf{H}, \mathsf{sk}])$ of the breaker circuit can find a valid message-signature pair (m, σ) which precisely makes up the special-purpose assumption of Garg et al.:

Assumption 5.1 ([GGHW14] Conjecture 1)**.** *There exists a signature scheme* S*, a collision resistant hash function* H *and a special-purpose obfuscator* $\mathsf{O}_{special}$ *such that the following holds: the probability for any* PPT *adversary* $\mathcal{A}$ *in finding a valid message-signature pair* (m, σ) *on input the security parameter* 1^λ*, verification key* vk *and obfuscation* $\mathsf{O}_{special}(C_{break}[\mathsf{H}, \mathsf{sk}])$ *is negligible. Here, the probability is over the choice of key pair* $(\mathsf{sk}, \mathsf{vk})$*, the choice of hash function* H*, the randomness of the obfuscator, and the coins of* $\mathcal{A}$*.*

The assumption guarantees that sampler Sam is differing-inputs and, hence, we have a valid counter-example.

Theorem 5.14 ([GGHW14] Theorem 1)**.** *Under the special-purpose obfuscation assumption (Assumption 5.1) general-purpose differing-inputs obfuscators with auxiliary information do not exist.*

5.7 The Choice of Computational Model

The Church–Turing thesis states that any physically realizable computation device can be simulated by a Turing machine. A *strong* form of the thesis strengthens it to say that every physically realizable computation device can be simulated by a Turing machine *with only polynomial overhead*. Although quantum computers (if they exist) might provide evidence against the strong Church–Turing thesis it holds when restricted to Boolean circuits. Any polynomial size uniform circuit family can be simulated by an efficient Turing machine and we can simulate non-uniform circuit families by Turing machines that take advice.[15]

The bottom line of the above discussion is that when considering *efficient computation* it makes no difference whether we consider (non-uniform) circuits or (non-uniform) Turing machines as the underlying computational model. When, on the other hand, we consider functions that take programs as input it seems to make a difference whether these programs are encodings of Turing machines or encodings of circuits (or some other model of computation). A key difference is between uniform models of computation (Turing machines) and non-uniform models (circuits).[16] If f is an effectively computable function then there exists a Turing machine M that computes f. Given an encoding of M one, thus, is able to compute f for any input $\{0,1\}^*$. In particular it is, for example, possible to compute $f(\mathsf{M})$ (i.e., run M on its own encoding). When given an encoding of circuit C_λ belonging to a sequence of circuits $\{C_\lambda\}_{\lambda\in\mathbb{N}}$ that compute f, then one can only evaluate f on inputs in $\{0,1\}^\lambda$. In particular, there might be no way of generating $C_{\lambda+1}$ given only C_λ and thus it might be impossible

[15] Note that a function which can only be computed by a non-uniform circuit (or non-uniform Turing machine) is essentially not *effectively computable*.

[16] We here identify the uniform model of computation with Turing machines and the non-uniform model with circuits which is technically not quite correct since we can consider uniform circuits as well as non-uniform Turing machines.

to compute $f(C_\lambda)$ (given only C_λ). In other words, Turing machine M information-theoretically encodes the entire (infinite) function table of f while circuit C_λ only needs to encode a small (finite) portion of it.[17] A second key difference between Turing machines and circuits is that the size of a circuit is directly related to the amount of computation that is required in the worst case. A Turing machine, on the other hand, may have a short description which is completely independent of the amount of computation it does.

Given the above differences it would not be too surprising if it turns out that there are notions of obfuscation which allow to obfuscate a circuit C_λ that computes a function $f : \{0,1\}^* \to \{0,1\}$ for inputs in $\{0,1\}^\lambda$ but that the corresponding Turing machine M that allows to compute f on all inputs cannot be obfuscated. Indeed, we have a candidate construction for VGB obfuscation for all circuits in NC^1 [BCKP14] but we know that we cannot achieve VGB obfuscation for Turing machines [BC14].

5.7.1 Obfuscation for Turing Machines

As explained in the introduction, we will mostly be using obfuscation for circuits. Indeed, circuit obfuscation is weaker than Turing machine obfuscation. Barak et al. show this to be the case for VBB obfuscation and we note that it generalizes for all general-purpose obfuscators presented in this thesis.

Proposition 5.15 ([BGI+12] Proposition 2.3)**.** *If a Turing machine VBB obfuscator exists, then a circuit VBB obfuscator exists.*

Proof sketch. Given a circuit C_λ with input length $\lambda \in \mathbb{N}$ construct a Turing machine M as follows: on input x Turing machine M checks that $|x| = \lambda$. If this is not the case it stops and outputs $\perp$. Otherwise it emulates C_λ on input x and outputs whatever C_λ outputs. We then apply the Turing machine obfuscator to M. □

The definition of Turing machine obfuscation is syntactically almost identical to the definition for circuits. The main difference is that instead of considering the size of a circuit we consider the running time of a Turing machine.

Definition 5.11 (Turing Machine Obfuscation)**.** *A* PPT *algorithm* O *is called an* obfuscation scheme *for an ensemble* $\mathcal{M} = \{\mathcal{M}\lambda\}_{\lambda\in\mathbb{N}}$ *of deterministic and polynomial-time Turing machines, if it holds that:*

- Functionality Preserving. *For any* $\mathsf{M} \in \mathcal{M}$ *it holds that* $\mathsf{O}(1^\lambda, \mathsf{M})$ *is a valid encoding of a Turing machine such that for all* $x \in \{0,1\}^*$ *it holds that*

$$\Pr\left[\mathsf{O}(1^\lambda, \mathsf{M})(x) = \mathsf{M}(x)\right] = 1.$$

- Polynomial Slowdown. *For any* $\mathsf{M} \in \mathcal{M}$ *the running time of* $\mathsf{O}(1^\lambda, \mathsf{M})$ *is at most polynomially larger than that of* M*. That is, there exists a polynomial* poly *such that* $\mathsf{time}_{\mathsf{O}(\mathsf{M})}(x) \leq \mathsf{poly}(\lambda + \mathsf{time}_{\mathsf{M}}(x))$ *for any input* $x \in \{0,1\}^*$.

[17] Again note that the difference is not necessarily the computational model but the efficiency of the encoding.

Given a definition of a Turing machine obfuscator, we can now recast the different obfuscation variants as Turing machine obfuscation schemes. For virtual black-box and virtual grey-box obfuscation the definitions are identical except that one replaces Definition 5.1 by Definition 5.11. When considering indistinguishability obfuscation or differing-inputs obfuscation we must translate the requirement that $|C_0| = |C_1|$ into the Turing machine language. Instead of the size of circuits, we now consider the running time: equality samplers are required to output two machine descriptions $(\mathsf{M}_0, \mathsf{M}_1)$ and a string aux such that

$$\Pr_{(\mathsf{M}_0,\mathsf{M}_1,\mathsf{aux}) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda)}[\forall x \in \{0,1\}^* : \mathsf{time}_{\mathsf{M}_0}(x) = \mathsf{time}_{\mathsf{M}_1}(x) \wedge \mathsf{M}_0(x) = \mathsf{M}_1(x)] \geq 1 - \mathsf{negl}(\lambda)\,.$$

For differing-input circuit-samplers we, analogously, require that it is difficult to find an x such that $\mathsf{M}_0(x) \neq \mathsf{M}_1(x)$. Furthermore, as Turing machines may have different runtimes on different inputs we also need to require that it is difficult to find an x such that $\mathsf{time}_{\mathsf{M}_0}(x) \neq \mathsf{time}_{\mathsf{M}_1}(x)$.

In the following section we discuss candidate constructions for obfuscation schemes—in particular for indistinguishability obfuscation. In Section 5.8.3 we then discuss assumptions which allow to construct iO also for Turing machines.

5.8 Candidate Indistinguishability Obfuscation Schemes

In the previous sections we have introduced different flavors of general-purpose obfuscation and seen that some of them—in particular VBB obfuscation but also VGB obfuscation for Turing machines—are not generally achievable. A main ingredient in many of the upcoming results will be a PPT indistinguishability obfuscator for all circuits and, thus, in this section we want to give a short introduction into the main ideas behind current candidate constructions of indistinguishability obfuscators. As the specifics of the constructions are irrelevant to the results presented in this thesis—the impatient reader can safely skip this section and continue with the following chapter—we will keep the following descriptions on a high level. For details, we refer to the paper by Garg et al. [GGH$^+$13b] who gave the first candidate construction for indistinguishability obfuscation as well as the many papers that followed [CV13, PST14, AGIS14, BR14, BGK$^+$14, GLSW14, AB15, Zim15, AJ15, BV15]. A recent survey on the constructions of indistinguishability obfuscation is given in [Hor15]

In what follows, we present the intuition and basic ideas behind the candidate obfuscator by Garg et al. [GGH$^+$13b] with extensions presented by Barak et al. [BGK$^+$14]. In particular we cover branching programs and graded encoding schemes.

5.8.1 How to Provably Obfuscate Low-Depth Circuits

The circuit model of computation is highly complex even if we restrict circuits to only consist of $\neg$ (NOT) and $\wedge$ (AND) gates and have binary output (i.e., a single output bit). In particular it is not clear what transformations (e.g., the introduction of extra gates) allow us to argue that the resulting circuit *hides certain information.* As an example consider the following circuit which has a random value $r \in \{0,1\}^\lambda$ hard-coded:

CIRCUIT $C[r](x)$

if $x = r$ **then**

  return 1

return 0

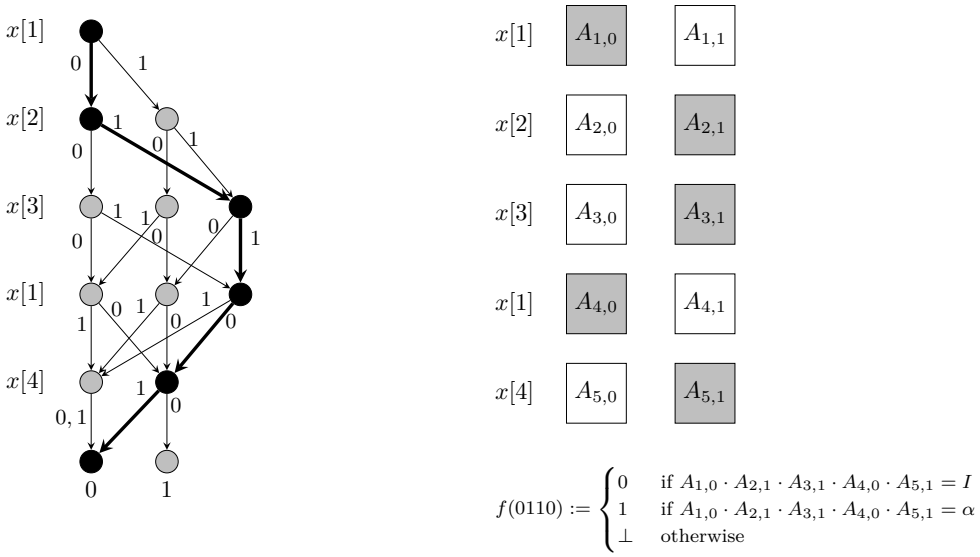
Circuit $C[r]$ computes a point function that is 0 everywhere except on input r. As we will see in the following chapter we have candidate constructions for very strong point-function obfuscation schemes, that is, we have special-purpose obfuscation schemes for the above circuit that we conjecture may even be VBB secure. Thus, as we are aiming at indistinguishability obfuscation, which we know to be best-possible obfuscation (see Section 5.5.4) we know that an indistinguishability obfuscation of $C[r]$ (for a high min-entropy r) must hide r since we assume that good point-function obfuscation exists. So, how can we transform $C[r]$ in a generic way—in particular the transformation should not have to assume that the input circuit computes a point function—into a circuit $\overline{C}[r]$ such that r is hidden?

Branching Programs

The first step in many current obfuscation candidates is to abandon the circuit model of computation and to go for a much simpler and highly structured model: so-called *branching programs*. Branching programs were first introduced by Lee [Lee59]—Lee called them *binary-decision programs*—and can be represented by a rooted layered acyclic directed graph with two leafs where every node except the two leafs has out-degree two. Each layer—a node is in the i-th layer if all paths from the root to the node have length $i-1$—corresponds to a bit position of the input. That is, if we consider a branching program for function $f : \{0,1\}^\lambda \to \{0,1\}$ then we consider a label function $\mathsf{inp} : [\ell] \to [\lambda]$ which assigns to each layer an index in $\{1,\ldots,\lambda\}$. The width w of a branching program is the maximum number of nodes per layer, its length ℓ is the number of layers (without counting the final output layer) and the program's size is the number of nodes. To evaluate a program on an input $x \in \{0,1\}^\lambda$ one starts at the root node and recursively performs the following operation: if at a non-leaf node in layer j one evaluates the label function $\mathsf{inp}(j)$ to obtain a bit position and then follows the outgoing edge labeled with $x[\mathsf{inp}(j)]$. Once a leaf node is reached the evaluation is complete and the function value is the value assigned to the leaf node. (Note that there are two leaf nodes one for 0 and one for 1.) Figure 5.2 contains an example branching program with the evaluation highlighted for input $x = 0111$ (that is, $x[1] = 0$ and $x[2..4] = 111$).

Branching programs, similarly to circuits, are an inherently non-uniform model of computation[18] that can recognize any language $\mathcal{L} \subseteq \{0,1\}^*$ if we do not restrict their width or size. In fact, one can show that it suffices to allow for exponential length and width four, or for exponential width and linear length. For the latter, consider a binary tree of depth λ where the leafs represent strings in $\{0,1\}^\lambda$ and the final level assigns to each string whether or not it is in the language. For the former note that we can think of the width of the branching program as its memory. We now consider a branching program that sequentially compares its input to each string in the language. What we need to remember are four different states: 1) start of a new word, 2) within the comparison of a

[18] We note that the question of non-uniform vs. uniform is a question of encoding and both branching programs and circuits can be encoded in a uniform way, e.g., by encoding a Turing machine that generated the program (or circuit). However, any natural encoding representing the structure of branching programs (or circuits) is non-uniform.



(a) Branching Program

(b) Matrix Branching Program

Figure 5.2: On the left (Figure (a)): a branching program of width $w = 3$ and length $\ell = 5$ computing a Boolean function $f : \{0,1\}^4 \to \{0,1\}$. Each layer is labeled with one input bit position, e.g., the third layer corresponds to input bit at position three and the fourth layer to input bit position one. The program is evaluated by starting at the root and at each step following the edge labeled with the value of the input bit corresponding to the current layer. On the right (Figure (b)): a (bounded width) matrix branching program of length $\ell = 5$. The program consists of 2ℓ matrices. On input $x \in \{0,1\}^4$ it is evaluated by multiplying together the matrices $A_{i,x[\mathsf{inp}(i)]}$ for $i = 1, \ldots, \ell$ and where $\mathsf{inp} : \mathbb{N} \to \mathbb{N}$ is the labeling function mapping the row to the bit position. The program evaluates to 0 if the result is the identity matrix and 1 if it is a fixed matrix α. For both programs we highlighted an evaluation for input $x = 0110$.

word everything matched so far, 3) within the comparison of a word a mismatch was found, 4) a match was found.

While the above tells us that branching programs are indeed a powerful model of computation, if we want to use them in the construction of an obfuscator we can at most make use of polynomial-size branching programs. While we can, for example, compute the AND or PARITY function on a bounded-width polynomial size branching program, it was conjectured that MAJORITY could not be computed by a bounded-width branching program [FSS84, BDFP86] since it seems difficult to count (note that our above intuition was that the width of the program can be thought of as its memory). Indeed, Yao showed a super-polynomial lower bound for MAJORITY for a width-2 program [Yao83]. Surprisingly, it turns out that bounded-width polynomial sized branching programs are much more powerful and exactly recognize the class of languages in NC^1 and thus, in particular, can also compute the MAJORITY function.

Theorem 5.16 ([Bar86])**.** *A language $\mathcal{L} \subseteq \{0,1\}^*$ is in NC^1 if, and only if there exists a polynomial-size width-5 branching program that decides $\mathcal{L}$. In particular, if $f : \{0,1\}^\lambda \to \{0,1\}$ is computable by a circuit C of depth d, then f is computable by a branching program of width 5 and length $\ell = 4^d$.*

Barrington's proof is constructive in that he provides a recipe to construct a polynomial-size width-5 branching program from any NC^1-circuit, i.e., for any polynomial-sized circuit with logarithmic depth. For this, he uses a special form of width-5 branching programs called *permutation branching programs* (PBP) where the two functions at each layer (mapping the i-th node of layer ℓ to the j-th node of layer $\ell+1$) are permutations over $\{1,2,3,4,5\}$. If $\mathsf{P} = (\mathsf{inp}, (A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell_1}))$ is a PBP with permutations $(A_{i,0}, A_{i,1})$ denoting the permutations at the i-th level and where $\mathsf{inp} : [\ell] \to [\lambda]$ is a function mapping levels to input bit positions, then an input $x \in \{0,1\}^\lambda$ induces a permutation consisting of the composition of the permutations[19] at each level corresponding to input x as:

$$\mathsf{P}(x) := A_{1,x[\mathsf{inp}(1)]} \circ A_{2,x[\mathsf{inp}(2)]} \circ \cdots \circ A_{\ell,x[\mathsf{inp}(\ell)]}.$$

Let $\mathsf{P}(x)$ denote this induced permutation. If $\mathcal{L} \subseteq \{0,1\}^*$ is a language and α a permutation then we say that a permutation branching program P *α-accepts* $\mathcal{L}$ if for any $x \in \mathcal{L}$ it holds that $\mathsf{P}(x) = \alpha$ and for any $x \notin \mathcal{L}$ we have that $\mathsf{P}(x) = \mathbf{1}$ where $\mathbf{1}$ denotes the identity permutation. In essence, what Barrington shows is how to construct a polynomial size width-5 PBP that α-accepts a language $\mathcal{L} \in \mathsf{NC}^1$ (for any permutation α that is a 5-cycle; defined shortly) given a circuit C (consisting only of AND and NOT gates) that decides $\mathcal{L}$. His construction recursively replaces parts of the circuit by branching programs which are "merged" together at AND gates (we show the construction of an AND gate in Example 5.8.1). We next present the proof to Barrington's theorem (following the proof given by Boppana and Sipser in their survey on finite functions [BS89]) which yields a recipe of how to construct branching programs given as input a circuit. In addition we give an example of how a simple circuit can be transformed into a branching program on page 102.

Proving Barrington's Theorem. As mentioned Barrington's proof is constructive and recursively builds up the branching program starting with branching programs for the identity circuits (Lemma 5.17) and then showing how to construct and merge branching programs for NOT gates and AND gates. An important concept is that of 5-cycle that we define next.

Definition 5.12 (5-cycle)**.** *A permutation over $\{1,2,3,4,5\}$ is a 5-cycle if it can be written in cycle notation: $(s_1 s_2 s_3 s_4 s_5)$ with distinct $s_i \in \{1,2,3,4,5\}$ and which denotes the permutation that maps $s_1 \to s_2 \to s_3 \to s_4 \to s_5 \to s_1$.*

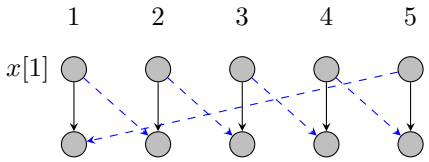
An example of a 5-cycle is the permutation (12345) which maps $1 \to 2, 2 \to 3, 3 \to 4, 4 \to 5$, and $5 \to 1$. We next show that there exists a PBP that α-accepts the identity function for any 5-cycle α.

Lemma 5.17. *Let $f : \{0,1\} \to \{0,1\}$ be the identity function on 1-bit input, that is, $f(1) = 1$ and $f(0) = 0$. Then, for any 5-cycle α there exist a PBP that α-accepts f.*

Proof. The PBP consists of a single layer where the 0-permutation is the identity function and the 1 permutation is α. □

As an example consider the 5-cycle (12345), then we can write the identity function on one bit as

[19] A PBP of width five works over the *symmetric group* $\mathcal{S}_5$.



The blue (dashed) arrows denote permutation α which is chosen if $x = 1$ and the black arrows denote the identity function which is chosen in case $x = 0$.

Lemma 5.18. *Let $f : \{0,1\}^\lambda \to \{0,1\}$ be a function and let α and β be 5-cycles. If there exists a PBP that β-accepts f then there is a PBP of the same size that α-accepts f.*

Proof. As both α and β are 5-cycles, there exists a permutation ρ such that $\alpha = \rho^{-1}\beta\rho$ where $\rho^{-1}\beta\rho$ denotes the composition $\rho^{-1} \circ \beta \circ \rho$. To see this, consider the permutation ρ that maps α to β, that is, consider $\alpha = (\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5)$ and $\beta = (\beta_1\beta_2\beta_3\beta_4\beta_5)$ and define

$$\rho := (\alpha_1 \to \beta_1, \alpha_2 \to \beta_2, \alpha_3 \to \beta_3, \alpha_4 \to \beta_4, \alpha_5 \to \beta_5).$$

Let $\mathsf{P} := (\mathsf{inp}, (A_{1,0}, A_{1,1}), \ldots, (A_{\ell,0}, A_{\ell_1}))$ be the PBP that β-accepts f, then we obtain P' that α-accepts f by setting:

$$\mathsf{P}' := (\mathsf{inp}, (\rho^{-1}A_{1,0}, \rho^{-1}A_{1,1}), (A_{2,0}, A_{2,1}), \ldots, (A_{\ell-1,0}, A_{\ell-1,1}), (A_{\ell,0}\rho, A_{\ell_1}\rho))$$

□

A simple corollary of the above is that if we have a PBP that α-accepts a function f for a 5-cycle α then we can obtain one that α^{-1}-accepts f by noting that if α is a 5-cycle then so is α^{-1}.

Corollary 5.19. *Let $f : \{0,1\}^\lambda \to \{0,1\}$ be a function and α a 5-cycle. If there exists a PBP that α-accepts f then there is a PBP of the same size that α^{-1}-accepts f.*

Next we will see how to combine two branching programs to compute the logical AND of their results.

Lemma 5.20. *Let $f, g : \{0,1\}^\lambda \to \{0,1\}$ be two functions and α and β be 5-cycles. Let f_α be a PBP that α-accepts f and let g_β be a PBP that β-accepts g. Then there exists a PBP P that $(\alpha\beta\alpha^{-1}\beta^{-1})$-accepts $(f \wedge g)$. Furthermore, P is of size $2 \cdot (\mathsf{size}(f_\alpha) + \mathsf{size}(f_\beta))$.*

Proof. We use the previous corollary to obtain programs $f_{\alpha^{-1}}$ and $g_{\beta^{-1}}$ that α^{-1}-accept f and β^{-1}-accept g. We then concatenate the programs in the following order $\alpha, \beta, \alpha^{-1}, \beta^{-1}$ (i.e., $f_\alpha \circ g_\beta \circ f_{\alpha^{-1}} \circ g_{\beta^{-1}}$) to obtain the desired result. For this note that if $f(x) = 0$ (resp. $g(x) = 0$) then we have that $f_\alpha(x) = \mathbf{1}$ (resp. $g_\beta(x) = \mathbf{1}$), i.e., the PBP f_α (resp. g_β) evaluates to the identity permutation. We, thus, get that the resulting program must also be the identity permutation as the program then reduces to $g_\beta \circ g_{\beta^{-1}} = \mathbf{1}$. □

Finally, we need to show that there are two 5-cycles as needed for the previous Lemma.

Lemma 5.21. *There are two 5-cycles α and β such that $\alpha\beta\alpha^{-1}\beta^{-1}$ is again a 5-cycle.*

Proof. Let $\alpha = (12345)$ and $\beta = (13542)$. Then:

$$\alpha\beta\alpha^{-1}\beta^{-1} = (12345) \circ (13542) \circ (54321) \circ (24531) = (13254).$$

□

With that we can now proof Theorem 5.16 by first rewriting an input circuit C such that it only contains AND and NOT gates and then recursively construct a permutation branching program that α-accepts C for some permutation α by applying the above lemmas.

Proof sketch of Theorem 5.16. Let C be a circuit of depth d that only consists of AND and NOT gates. For $d = 1$, Lemma 5.17 yields a branching program as the circuit cannot have any intermediate gates between the (single) input and (single) output gate. The proof follows by induction over the depth of the circuit noting that NOT gates can be transformed using Corollary 5.19 and AND gates using Lemmas 5.18, 5.20, and 5.21.

Finally, note that given a PBP that α-accepts C, we can obtain a regular branching program by identifying the top-left node as the root and use the connected component as the branching program. The bottom left node is associated to the program being 0 and the node corresponding to 1 is given by permutation α as $\alpha(1)$, that is, the image of 1 under permutation α. Also see Figure 5.3 within Example 5.8.1. □

From Branching Programs to Obfuscation. Barrington's theorem provides us with a simple and structured model of computation which is as powerful as low-depth circuits. Indeed, permutation branching programs form the basis of the obfuscator of Garg et al. [GGH+13b]. If we write the permutations in a permutation branching program as matrices (where the composition is simply matrix multiplication), we get matrix branching programs which are defined as follows (we give an example in Figure 5.2 on the right):

Definition 5.13. *A* matrix branching program *of width w and length ℓ for λ-bit inputs consists of a fixed permutation matrix $\alpha \in \{0,1\}^{w\times w}$, that is different from the identity $\alpha \neq I_{w\times w}$, and a sequence:*

$$\mathsf{MP} := (\mathsf{inp}(i), A_{i,0}, A_{i,1})_{i=1,\ldots,\lambda}$$

Each $A_{i,b}$ (for $b \in \{0,1\}$) is a permutation matrix in $\{0,1\}^{w\times w}$ and function $\mathsf{inp} : [\ell] \to [\lambda]$ defines the input bit position for step i. The output of the matrix branching program on input $x \in \{0,1\}^\lambda$ is computed as

$$\mathsf{MP}(x) := \begin{cases} 0 & \text{if } \prod_{i=1}^{\ell} A_{i,\mathsf{inp}(i)} = I_{w\times w} \\ 1 & \text{if } \prod_{i=1}^{\ell} A_{i,\mathsf{inp}(i)} = \alpha \\ \bot & \text{otherwise} \end{cases}$$

Given a matrix branching program (MBP) one can obtain an *oblivious matrix branching program* where function inp is independent of the function being computed by the MBP but only depends on the input length.[20] Transforming the circuit into an oblivious matrix branching program is the

[20] This is similar to the notion of an oblivious Turing machine where the head position at step i only depends on the the input length and not the input itself.

Example 5.2: Example of Permutation Branching Program

As an example consider the language $\mathcal{L} \subseteq \{0,1\}^\lambda$ which contains any $x \in \{0,1\}^\lambda$ such the first two bits are both 1. The corresponding function can be written in Boolean notation as $x[1] \wedge x[2]$. Barrington's construction recursively constructs the branching program by first transforming each input into a branching program and then recursively replacing gates where all inputs have already been replaced. In the example we would thus first create branching programs for the two input variables $x[1]$ and $x[2]$ (see Lemma 5.17). Here, we can choose two arbitrary 5-cycles, say (12345) and (13524). Next we need to replace the single AND gate. To transform an AND gate, the construction first transforms the two ingoing branching programs f and g into four branching programs: one that α-accept and one α^{-1}-accept the same language as f, and g is transformed into programs that β-accept and β^{-1}-accept the same language as g (see Lemma 5.20). Furthermore, it is required that the permutations α and β satisfy that $\alpha\beta\alpha^{-1}\beta^{-1}$ is again a 5-cycle which, for example, is the case for $\alpha = (12345)$ and $\beta = (13524)$. Finally, to obtain a branching program that computes $f \wedge g$ one concatenates the programs as $\alpha\beta\alpha^{-1}\beta^{-1}$. What we end up with is a branching program that $\alpha\beta\alpha^{-1}\beta^{-1}$-accepts language $(x[1] \wedge x[2])$. In Figure 5.3, we present the PBP for language $x[1] \wedge x[2]$. The blue (dashed) arrows correspond to the permutation to choose on the input bit being 1 and the black arrows to the input bit being 0. Note that from a PBP that τ-accepts a language can easily be turned into a basic width-5 branching program by considering the top left node as the start node and considering the bottom left node as the 0 leaf and the leaf number $\tau(1)$ (3 in the example) as the 1 node.

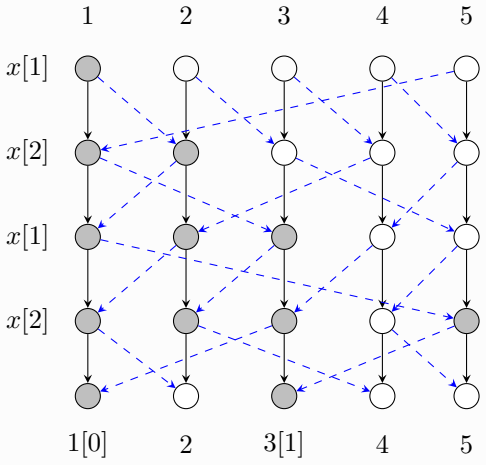


Figure 5.3: Branching program for language $x[1] \wedge x[2]$. The black arrows correspond to the input being 0 and the blue (dashed) arrows correspond to the input being 1. If only considering the gray nodes (and using the squared label in the last layer) one obtains a regular branching program from the PBP.

first step in the candidate construction. Note that this means that the resulting obfuscator can only obfuscate circuits in NC^1. We will see how this can be bootstrapped to circuits in P/poly in Section 5.8.2.

Remark. We note that in the actual construction of Garg et al. [GGH+13b] not the circuit that is to-be obfuscated is transformed into a branching program but rather the universal circuit $\mathsf{UC}(\cdot,\cdot)$ which takes as first input a circuit description C and as second input a string x and which evaluates C on x, that is, $\mathsf{UC}(C,x) = C(x)$. Then, to obfuscate a circuit C one fixes the inputs to the obfuscated branching program computing UC, by throwing away the matrices not corresponding to the to-be obfuscated circuit C. The result is an MBP for $\mathsf{UC}(C,\cdot)$. For our discussion, we neglect this detail and "directly" obfuscate circuit C.

Kilian's Randomization

Although matrix branching programs might seem already rather obfuscatory they do not offer provable guarantees. Thus, in a second step, the MBP is randomized using Kilian's randomization technique [Kil88]. Kilian gives a protocol that allows two parties (Alice and Bob) to securely evaluate an NC^1 circuit C on a joint input (x, y). Here, Alice computes the MBP for circuit C and then chooses $\ell - 1$ random invertible matrices $\{R_i\}_{i\in[\ell-1]} \in \mathbb{Z}_p$. She sets $\tilde{A}_{i,b} \leftarrow R_{i-1}A_{i,b}R_i^{-1}$ for $i = 1, \ldots, \ell$, $b \in \{0, 1\}$ and with R_0 and R_ℓ being the identity.[21] This yields a *randomized MBP*. This does not change the function of the MBP as the random matrices "cancel out" for an honest evaluation. To complete Kilian's protocol Alice sends over the matrices corresponding to her input and engages with Bob in an oblivious transfer protocol such that he can obtain the remaining matrices corresponding to his input. On completion Bob holds all matrices for the joint input and can, thus, evaluate the MBP. Due to the use of oblivious transfer, Alice does not learn Bob's input and Kilian shows that the randomization suffices for Bob not being able to learn Alice's inputs.

The second step for the obfuscation scheme is to apply Kilian's randomization to the MBP to also obtain a randomized MBP. While this is sufficient for the application of jointly evaluating a circuit, the result is not yet a secure obfuscation. For this note that the obfuscation consists of all the matrices whereas, in the two-party computation setting, Bob only learns those matrices corresponding to his input. We have, however, made quite some progress as Garg et al. [GGH+13b] argue that an obfuscator that just produces randomized MBPs allows for only three types of attacks, which they classify as:

Partial evaluation attacks. Partial evaluation attacks consider adversaries that compute the matrix product only up to a level j for different inputs and try to learn something about the functionality from comparing these intermediate values. For this note that the randomization is identical for both matrices at each level.

Mixed-input attacks. Mixed-input attacks consider adversaries that do not respect the input position function inp. As an example consider the MBP in Figure 5.2 on page 98 on the right and an adversary that chooses the second matrix $A_{4,1}$ in the fourth step even though it already fixed input bit $x[1]$ to 1 by choosing matrix $A_{1,0}$ in the first step. Again note that also in this case the randomization does not help as it will still cancel out.

Structure violating attacks. This class of attacks considers adversaries that try to learn something about the program by means other than evaluating multilinear forms of the given elements (i.e., matrices of the branching programs), in other words, the adversary does not only properly multiply matrices from different levels of the branching program but instead uses the matrices (or the encoded values) in any other way.

Most obfuscation candidates perform the aforementioned two steps to transform the circuit into a matrix branching program and then use Kilian's randomization technique. They deviate on how they handle remaining attacks [GGH+13b, CV13, PST14, BR14, BGK+14, GLSW14]. Ananth et al. [AGIS14] provide methods to optimize the generation of branching programs and show how the construction due to Barrington's theorem can be avoided (note that the size of a branching program

[21] Several obfuscation candidate constructions do not set R_0, R_ℓ to the identity but rather set them to correspond to particularly chosen "bookends" which we ignore for the discussion here.

due to Barrington's construction is exponential in the depth of the ingoing circuit). Applebaum and Brakerski [AB15], as well as Zimmerman [Zim15] were the first to provide alternative constructions that do not first require to transform a circuit into a branching program. They do, however, still base their construction on the security of a *Graded Encoding Scheme* (a form of multilinear maps, that we will introduce shortly). Finally, Ananth and Jain [AJ15], and independently Bitansky and Vaikuntanathan [BV15], show how to obtain indistinguishability obfuscation from a general-purpose functional encryption scheme that has succinct ciphertexts and sub-exponential security.

Multilinear Encodings

As explained above, there are still a few attack angles that can be exploited and this is where the various constructions differ. A common tool most constructions use are graded encoding schemes. Exceptions are the constructions from functional encryption [AJ15, BV15], and the construction due to Canetti and Vaikuntanathan [CV13] who instead use black-box pseudo-free groups. Graded encoding schemes can be regarded as an approximation of multilinear maps. Multilinear maps are a generalization of bilinear maps and were first introduced by Boneh and Silverberg [BS03]. Formally, consider $\kappa + 1$ cyclic groups $G_1, \ldots, G_\kappa, G_T$ of the same order p and a κ-multilinear map $e : G_1 \times \cdots \times G_\kappa \to G_T$ such that for any elements $(g_i \in G_i)_{i=1,\ldots,\kappa}$, and any integers $(\alpha_i \in \mathbb{Z}_p)_{i=1,\ldots,\kappa}$ it holds that

$$e(\alpha_1 \cdot g_1, \ldots, \alpha_\kappa \cdot g_\kappa) = \left(\prod_{i=1}^{\kappa} \alpha_i\right) \cdot e(g_1, \ldots, g_\kappa). \tag{5.7}$$

Furthermore e is not degenerate, that is, if elements $(g_i \in G_i)_{i=1,\ldots,\kappa}$ are all generators of their respective groups then $e(g_1, \ldots, g_\kappa)$ is a generator of the target group G_T.

While multilinear maps would have intriguing applications in cryptography—Boneh and Silverberg showed that multilinear maps allow for efficient broadcast encryption and one-round multiparty key-exchange [BS03]—we do not have any candidate construction. In a breakthrough work, Garg, Gentry, and Halevi (GGH; [GGH13a]) consider a relaxation that they call *graded encoding schemes* and show how to obtain such a scheme from lattices. We can think of $a \cdot g_i$ for an integer $a \in \mathbb{Z}_p$ and a generator g_i as an *obfuscated* encoding of the integer a in group G_i. GGH retain this concept of having encodings but consider randomized encodings over a ring R which replaces the space $\mathbb{Z}_p$. In particular, this means that now a value $a \in R$ can have many encodings. As a consequence it becomes non-trivial to check whether two strings encode the same element. Secondly, each encoding is associated with an index set $S \subseteq U$ from some universe U; the ring elements $a \in R$ are associated with the empty index set $\{\}$. While a multilinear map (Equation (5.7)) allows only to multiply exactly κ encodings together (one from each group), a graded encoding scheme allows to multiply any subset of elements given that they have disjoint index sets.

We next give the definition of graded encoding schemes due to [BGK+14].

Definition 5.14. *Let R be a ring and U a universe set. We denote by $[\alpha]_S$ an* encoded element *where $\alpha \in R$ is the* value *and $S \subseteq U$ is the* index *of the element. Furthermore, we define two binary operations over elements as:*

- *For two elements $[\alpha]_S, [\beta]_S$ with identical indices we define $[\alpha]_S + [\beta]_S$ to be the element $[\alpha+\beta]_S$ and similarly $[\alpha]_S - [\beta]_S$ to be the element $[\alpha - \beta]_S$.*

- *For two elements* $[\alpha]_S, [\beta]_T$ *such that* $S \cap T = \{\}$*, we define* $[\alpha]_S \cdot [\beta]_T$ *to be the element* $[\alpha \cdot \beta]_{S \cup T}$*.*

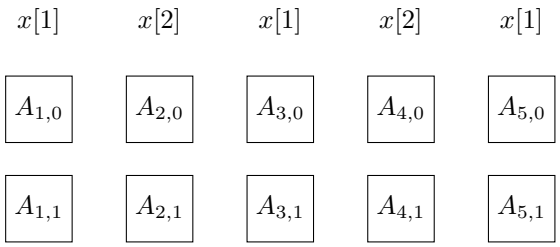
To be useful for cryptographic purposes, a graded encoding scheme must come with efficient algorithms to generate encodings with different indices for any element $\alpha \in R$. The above binary operations addition and multiplication should be efficient and we need an efficient procedure to test if an encoding that has index U (the entire universe) is 0. In particular, if $[\alpha]_S$ for $S \subsetneq U$ is an element then the zero test should return $\perp$ as S is not equivalent to U.

Ideal graded encoding scheme. Given the above definition of a graded encoding scheme we need to specify what security guarantees we require. Garg et al. [GGH13a] model hardness assumptions after the discrete-logarithm and DDH assumptions, which are hardness assumptions for the "clean" multilinear maps [BS03]. Very strong assumptions are obtained when applying the analogue of the *generic group model*[22] where group elements are modeled as random handles and an oracle is provided to perform the group operation. This is called the *generic* or *ideal* graded encoding model and is used as basis for the security of most current candidate constructions.

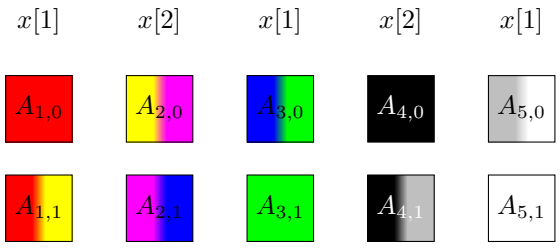
Multilinear maps and obfuscation. Graded encoding schemes form the basis of protections against the remaining attacks on the intermediate obfuscator: the randomized matrix branching program which we obtained by first constructing a matrix branching program and then applying Kilian's randomization technique. For the next step, we encode the matrices in the branching program with the graded encoding scheme. Consider a randomized matrix branching program of size ℓ, that is, it consists of ℓ pairs of matrices. Consider further the universe $U := \{1, \ldots, \ell\}$. Then, if we encode the i-th pair of matrices (element wise) relative to index $\{i\}$ we can still evaluate the branching program: The multiplication operation allows us to multiply together one matrix from each pair to obtain an encoded matrix (each element is encoded) with index U. This, we can then test for being the identity matrix using the zero-test procedure.

What have we gained? By encoding the matrices and assuming that the encoding scheme is ideal we can argue that no adversary employing structure violating attacks can gain any advantage. Furthermore, the encoding also forms the basis of protecting against *mixing input attacks* and *partial evaluation attacks*. For this, Barak et al. [BGK+14] present an intriguingly simple idea leveraging the fact that with graded encodings only elements with disjoint indexes can be multiplied. They present a set system they call *straddling sets* that ensures that if an adversary fixes the input bit for one level of the branching program it cannot later deviate from this choice. The idea is best exemplified by a simple branching program. Consider a branching program that takes inputs of length two and which inspects the first input bit three times and the second input bit two times (the example is rotated, that is each column represents one level):

[22] The *generic group model* was first proposed by Shoup [Sho97] to bound the success probability of adversaries which do not exploit special properties of the encodings of group elements. The generic group model is similar in spirit to the random oracle model in that it is not clear as to what security proofs given the generic group model imply in the real world where we need to pick a particular encoding [Den02].



The idea is that, if an adversary picks an input bit for one level, then it has to pick the same input bit for all the levels that correspond to the same input bit. Remember that for the graded encoding scheme to be able to perform a zero-test we need to obtain an element which is indexed with the entire universe. In the following example we color-coded the different matrices and consider the universe which consists of the all the colors: red, yellow, magenta, blue, green, black, gray, and white.



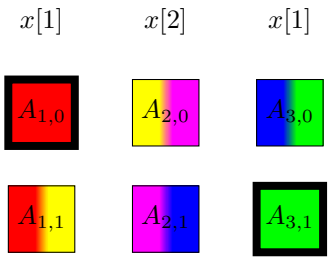
In order for an adversary to get a "meaningful" element it must combine (multiply) the given matrices in such a form, that the result is indexed with all the colors. As we consider two input bits, there should be exactly four valid combinations:

$$\text{(00)}: A_{1,0} \cdot A_{2,0} \cdot A_{3,0} \cdot A_{4,0} \cdot A_{5,0}$$
$$\text{(01)}: A_{1,0} \cdot A_{2,0} \cdot A_{3,0} \cdot A_{4,1} \cdot A_{5,1}$$
$$\text{(10)}: A_{1,1} \cdot A_{2,1} \cdot A_{3,1} \cdot A_{4,0} \cdot A_{5,0}$$

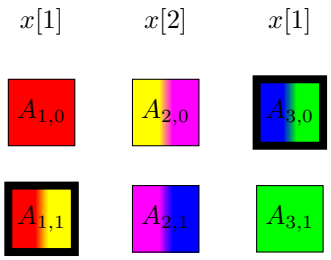
and

$$\text{(11)}: A_{1,1} \cdot A_{2,1} \cdot A_{3,1} \cdot A_{4,1} \cdot A_{5,1}$$

All these combinations have in common that any two matrices have disjoint colors and, thus, can be multiplied together (see Definition 5.14). However, once an adversary tries to mix inputs, for example, if an adversary chooses the matrices $A_{1,0}$ and $A_{3,1}$ then there is no way to choose the other matrices in such a way as to obtain an exact cover of the universe: if the adversary chooses $A_{2,1}$ then yellow is missing, and if it chooses $A_{2,0}$ then blue is missing.



Similarly, if it chooses $A_{1,1}$ and $A_{3,0}$ then it can choose neither $A_{2,0}$ nor $A_{2,1}$ because then the index sets are not disjoint ($A_{1,1}$ clashes with $A_{2,0}$ and $A_{3,0}$ clashes with $A_{2,1}$).



The above idea can easily be generalized and it can be shown that it allows for protecting against mixing inputs and partial evaluation attacks assuming that the underlying graded encoding scheme is ideal. We refer to Barak et al. [BGK+14] for a formalization of straddling set systems and their construction. With this, we conclude our description of obfuscators for low-depth circuits.

5.8.2 Bootstrapping Obfuscation using FHE

In the previous section we introduced the ideas behind core obfuscators for low-depth circuits (i.e., circuits in NC^1). Garg et al. [GGH+13b] show how such an obfuscator can be bootstrapped to obtain obfuscation for any circuit in P/poly if we additionally assume the existence of a perfectly correct, fully homomorphic encryption scheme and a perfectly sound non-interactive witness-indistinguishable proof system.

Fully homomorphic encryption. Fully homomorphic encryption (FHE) was first envisioned by Rivest, Adleman, and Dertouzos [RAD78] in 1978 and a first candidate construction was given over 30 years later in a highly celebrated work by Gentry [Gen09]. In essence, FHE allows to evaluate arbitrary functions over encrypted data. That is, FHE is a public-key encryption scheme that comes with an additional (publicly evaluatable) algorithm $\mathsf{FHE.Eval}_{\mathsf{pk}}(f, c_1, \ldots, c_\ell)$ which takes as input a function f (which itself takes ℓ inputs) and ℓ ciphertexts and outputs a ciphertext c_f such that[23]

$$\Pr[\mathsf{FHE.Dec}_{\mathsf{sk}}(c_f) = f(\mathsf{FHE.Dec}_{\mathsf{sk}}(c_1), \ldots, \mathsf{FHE.Dec}_{\mathsf{sk}}(c_\ell))] \geq 1 - \mathsf{negl}(\lambda)\,. \tag{5.8}$$

[23]Normally, we would need a stronger requirement to exclude the trivial solution in which $\mathsf{FHE.Eval}_{\mathsf{pk}}$ is the identity function and FHE.Dec first decrypts and then evaluates function f. The given requirement should, however, illustrate the idea behind the functionality of an FHE scheme.

Here, $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{FHE.KGen}(1^\lambda)$ denotes the public and secret key. Security can be defined via the usual indistinguishability-based security notions (e.g. IND-CPA).[24] We speak of a perfectly correct FHE scheme if the probability in Equation (5.8) is 1.

Besides allowing to evaluate ciphertexts on a known function we can use a universal circuit to compute $\mathsf{FHE.Eval}_{\mathsf{pk}}(\mathsf{UC}, \cdot)$ and, thus, we can also allow to evaluate ciphertexts on arbitrary (depth-bounded) functions. This is closely related to obfuscation. Consider the case where we want to obfuscate a circuit C. Using FHE we could generate an FHE key pair $(\mathsf{pk}, \mathsf{sk})$ and output the public key pk and an encryption $\overline{C} \leftarrow\!\!\$\ \mathsf{FHE.Enc}_{\mathsf{pk}}(C)$ of circuit C as an obfuscation of C. Given $\overline{C}$ and pk one can now evaluate $\overline{C}$ on any value x by encrypting $\overline{x} \leftarrow\!\!\$\ \mathsf{FHE.Enc}_{\mathsf{pk}}(x)$ and then evaluating $c \leftarrow\!\!\$\ \mathsf{FHE.Eval}_{\mathsf{pk}}(\mathsf{UC}, \overline{C}, \overline{x})$. The result c is an encryption of the output value $C(x)$. Thus, while obfuscation allows us to evaluate an obfuscated circuit on any input "in the plaintext space", FHE allows us to evaluate a circuit on any input "in the ciphertext space".

Non-interactive proof systems. The second ingredient used by Garg et al. [GGH+13b] for their bootstrapping mechanism are non-interactive proof systems [BFM88]. A proof system for an NP-language $\mathcal{L}$ consists of a prover and a verifier where the prover tries to convince the verifier of a true statement. We call a proof system complete if an honest prover can always convince an honest verifier of a true statement $x \in \mathcal{L}$ if the prover is given a witness to the effect. The system is sound if it is infeasible for a malicious prover to convince an honest verifier of a false statement $x \notin \mathcal{L}$. The proof system is called non-interactive if the interaction consists of only a single message sent from the prover to the verifier.

Obfuscation for P/poly

How can we combine FHE, non-interactive proof systems and indistinguishability obfuscation for low-depth circuits to obtain indistinguishability obfuscation for P/poly? The idea follows the two-key paradigm for CCA-encryption of Naor and Yung [NY90] who use an IND-CPA secure public-key encryption scheme and a non-interactive proof system to build CCA-secure encryption. Assume that we have a perfectly correct FHE scheme with a decryption circuit in NC^1.[25] Further assume that we have a perfectly sound non-interactive witness-indistinguishable proof system with low-depth proofs, where the last property means that the verifier V has an implementation in NC^1.[26] Now consider a setup with two key pairs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow\!\!\$\ \mathsf{FHE.KGen}(1^\lambda)$ and $(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow\!\!\$\ \mathsf{FHE.KGen}(1^\lambda)$ and two encryptions of a circuit C denoted by $\hat{C}_1 \leftarrow\!\!\$\ \mathsf{FHE.Enc}_{\mathsf{pk}_1}(C)$ and $\hat{C}_2 \leftarrow\!\!\$\ \mathsf{FHE.Enc}_{\mathsf{pk}_2}(C)$. Furthermore consider the following circuit which has values $\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \hat{C}_1$ and $\hat{C}_2$ hard-coded.

$C_{\mathsf{Dec}}[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \hat{C}_1, \hat{C}_2](c_1, c_2, \pi)$

if proof π verifies relative to $(\mathsf{pk}_1, \mathsf{pk}_2, \hat{C}_1, \hat{C}_2)$ **then**

  return $\mathsf{FHE.Dec}_{\mathsf{sk}_1}(c_1)$

return $\perp$

The low-depth proof π verifies that c_1 and c_2 are generated as

$c_1 \leftarrow\!\!\$\ \mathsf{FHE.Eval}_{\mathsf{pk}_1}(\mathsf{UC}(\cdot, x), \hat{C}_1)$

$c_2 \leftarrow\!\!\$\ \mathsf{FHE.Eval}_{\mathsf{pk}_2}(\mathsf{UC}(\cdot, x), \hat{C}_2)$

for the same input x.

[24] Note that due to the publicly evaluatable Eval algorithm IND-CCA2 security cannot be achieved.

[25] Such a scheme is, for example, given by Brakerski, Gentry and Vaikuntanathan [BGV12]. Furthermore, we note that, in fact, only a leveled FHE scheme is required for the construction to work [GGH+13b].

[26] Garg et al. [GGH+13b] show how to construct such a proof system.

Circuit C_{Dec} takes as input two ciphertexts c_1, c_2 and a proof π which verifies that the two ciphertexts were generated by homomorphically evaluating circuit C on input x once with public key pk_1 and once with public key pk_2. In other words c_b was generated as

$$\hat{x}_b \leftarrow\!\$\, \mathsf{FHE.Enc}_{\mathsf{pk}_b}(x)$$
$$c_b \leftarrow\!\$\, \mathsf{FHE.Eval}_{\mathsf{pk}_b}(\mathsf{UC}, \hat{C}_b, \hat{x}_b)$$

If c_1 and c_2 are properly constructed (which is ensured by proof π) then $C_{\mathsf{Dec}}(c_1, c_2, \pi)$ returns a decryption of c_1 and hence returns value $C(x)$.

In summary, an obfuscation of a circuit C consists of encryptions of C under both public keys, the two public keys as well as the obfuscation of C_{Dec}:

OBFUSCATE(C)

$(\mathsf{sk}_1, \mathsf{pk}_1) \leftarrow\!\$\, \mathsf{FHE.KGen}(1^\lambda)$
$(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow\!\$\, \mathsf{FHE.KGen}(1^\lambda)$
$\hat{C}_1 \leftarrow\!\$\, \mathsf{FHE.Enc}_{\mathsf{pk}_1}(C)$
$\hat{C}_2 \leftarrow\!\$\, \mathsf{FHE.Enc}_{\mathsf{pk}_2}(C)$
$\overline{C}_{\mathsf{Dec}} \leftarrow\!\$\, \mathsf{iO}(C_{\mathsf{Dec}}[\mathsf{sk}_1, \mathsf{pk}_1, \mathsf{pk}_2, \hat{C}_1, \hat{C}_2])$
return $(\mathsf{pk}_1, \mathsf{pk}_2, \hat{C}_1, \hat{C}_2, \overline{C}_{\mathsf{Dec}})$

As π is a low-depth proof and since the decryption operation is in NC^1 we have that the above circuit also is in NC^1 and we can obfuscate it with our core obfuscator presented in Section 5.8.1. Furthermore, we can argue via the two-keys technique (see Section 5.5.5 on page 89) that the above obfuscator is indeed an indistinguishability obfuscator. For this note that an obfuscation of C_{Dec} hides the secret key sk_1 down to the security of the FHE scheme and the proof system.

Summary. In summary, we can obtain an indistinguishability obfuscator for all circuits in P/poly by constructing a core obfuscator for circuits in NC^1 and then bootstrap the obfuscator using FHE. We note that due to the idealized models used in the proof (ideal graded encoding schemes) many candidate constructions are actually shown to be VBB secure. One can hope that the security nicely degrades when we replace ideal graded encoding schemes by real encoding schemes and that, in the end, we end up with VGB obfuscation or at least indistinguishability obfuscation. One can, however also take a more pessimistic view (similarly to the random oracle controversy) and argue that if we can prove a security level that we know cannot exist that this at best can be seen as a sanity check, and that we should search for standard model validations. Indeed, first steps in this direction have been made and we discuss the plausibility of candidate constructions in Section 5.9. Next we briefly discuss obfuscation candidates for Turing machines.

5.8.3 Indistinguishability Obfuscation for Turing Machines

In the previous sections we have seen how to construct indistinguishability obfuscation for all circuits in P/poly under strong assumptions. A strictly stronger result would be the construction of indistinguishability obfuscation for Turing machines as this allows to evaluate the obfuscated program on inputs of arbitrary length.

Ananth et al. [ABG+13] and Boyle et al. [BCP14] give constructions of indistinguishability obfuscators for Turing machines assuming the existence of general purpose differing-inputs obfuscation and succinct non-interactive arguments of knowledge (SNARKs)[27]. Assuming differing-inputs obfuscation for circuits, both constructions, in fact, also achieve differing-inputs obfuscation for Turing machines. Regrettably, the implausibility result for general purpose differing-inputs obfuscation (see Section 5.6.2) also applies to the above constructions and thus, it is not clear how much confidence we should have in the existence of iO for Turing machines.

Very recently, further progress towards the construction of Turing machine obfuscation was made by Koppula, Lewko and Waters [KLW14] as well as Ishai, Pandey, and Sahai [IPS15]. Koppula et al. [KLW14] show how to construct iO for Turing machines with unbounded memory but still with a restriction on the input size, i.e., a program obfuscation can only be evaluated on fixed-length inputs. For this, they rely only on iO for circuits, one-way functions and injective pseudorandom generators. Ishai et al. [IPS15] in turn present a weaker form of differing inputs obfuscation called *public-coins differing inputs obfuscation* for which the known implausibility results for full diO do not seem to carry over. Ishai et al. further show that the construction of indistinguishability obfuscation for Turing machines due to Ananth et al. [ABG+13] can be adapted to also work in the public-coins differing-input setting.

5.9 On the Plausibility of General-Purpose Obfuscation

For the remainder of this chapter we give a brief abstract over the history of general-purpose obfuscation candidates focusing on plausibility of their underlying security assumptions. For further details, we refer to the recent survey on indistinguishability obfuscation candidates by Horváth [Hor15] as well as the candidate papers themselves.

The first candidate construction for a general-purpose indistinguishability obfuscator was presented less than two years ago by Garg et al. [GGH+13b]. Their construction is based on a novel assumption called the *Equivalent Program Indistinguishability Assumption*. For the obfuscation of an NC^1 circuit C, Garg et al. create an "encoded matrix branching program" for the universal circuit $\mathsf{UC}(\cdot,\cdot)$. That is, they first obfuscate $\mathsf{UC}(\cdot,\cdot)$ with the techniques highlighted in Section 5.8.1 and call this an "encoded matrix branching program" for UC. Let us denote this encoded matrix program for the universal circuit by $\overline{\mathsf{uC}}$. Then, to obtain an obfuscation of circuit C Garg et al. fix the first input of $\overline{\mathsf{uC}}$ to C by removing the respective matrices from the branching program. The equivalent program indistinguishability assumption now informally says the following:

Assumption 5.2 (Equivalent Program Indistinguishability Assumption [GGH+13b]—informally)**.** *Given the "encoded matrix branching program" of a circuit $\overline{C}(\cdot,\cdot)$ taking two inputs, as well as two different assignments a_1 and a_2 for the first input such such that the resulting programs are functionally equivalent, that is,*

$$\forall x : \overline{C}(a_1, x) = \overline{C}(a_2, x),$$

[27] SNARKs are short and publicly verifiable arguments of knowledge for NP that give the security guarantee that if a prover generates a correct (and short) proof then with overwhelming probability it "knows" a witness to the statement.

then we have that

$$\overline{C}(a_1, \cdot) \approx_c \overline{C}(a_2, \cdot).$$

Here, the distributions are over the construction of the encoded matrix branching program for $\overline{C}$*.*

On closer inspection, we find that the assumption closely models the intended application: if we define our obfuscator for a circuit C to be the encoded matrix branching program for the universal circuit UC with its first input fixed to C then the assumption tells us that for any different representation of C the obfuscations are indistinguishable. In order to strengthen the confidence in the above assumption Garg et al. further validate the assumption in an idealized model called the *generic colored matrix model*.

In subsequent works, Brakerski and Rothblum [BR14] and Barak et al. [BGK+14] have simplified the construction further and showed that it is secure against all generic multi-linear attacks. While previous works use Barrington's Theorem as an intermediate step, Ananth et al. [AGIS14] identified this as a major factor that can be optimized to obtain more efficient obfuscation schemes[28] and showed how to obfuscate general branching programs. In [MSW14], Miles, Sahai, and Weiss go beyond the generic graded encoding model and allow adversaries to perform unlimited additions across different levels (note that the graded encoding model usually forbids such operations) and they show that also in this *arithmetic* setting the candidate construction yields VBB security.

Complementary, Pass, Seth and Telang [PST14] and Gentry et al. [GLSW14] set out to develop techniques on proving the security of indistinguishability obfuscators down to standard model assumptions. Pass et al. [PST14] show how to base an adapted construction on a novel meta-assumption they call *Semantically-Secure Multilinear Encodings* and Gentry et al. [GLSW14] show that iO can be based on instance-independent assumptions giving a construction based on the *Multilinear Subgroup Elimination Assumption.*[29]

While previous works mostly build on the breakthrough candidate construction of Garg et al. [GGH+13b], Zimmerman [Zim15] as well as Applebaum and Brakerski [AB15] develop obfuscators which do not use branching programs as an intermediate step but, instead, take a more direct approach. However, what they have in common with their branching program relatives is the use of multilinear graded encoding schemes.

Multilinear maps. All candidates discussed above base their security mostly on the existence of multilinear graded encoding schemes: [GGH+13b, BR14, BGK+14, AGIS14, MSW14, Zim15, AB15] analyze their constructions in idealized graded encoding models and [PST14, GLSW14] consider strong standard model assumptions for graded encoding schemes.

Graded encoding schemes were first proposed by Garg, Gentry, and Halevi [GGH13a] as an approximation of multilinear maps only a year before the first candidate indistinguishability obfuscator was presented. The construction presented by Garg et al. [GGH13a] is based on ideal lattices and, shortly after, a construction over the integers was proposed by Coron, Lepoint, and Tibouchi [CLT13].

[28] We note that, currently, obfuscation techniques can only be termed efficient in a theoretical setting as the constants hidden in the Landau notation for multilinear map candidates are huge. However, given time we might see speed-ups akin to those in multi-party computation or fully homomorphic encryption schemes.

[29] Gentry et al. [GLSW14] require subexponential security and Pass et al. [PST14] note that if they similarly assume subexponential security of the underlying primitives that then their construction can also be based on a single, efficiently falsifiable assumption [Nao03].

A second construction from lattices was recently presented by Gentry, Gorbunov, and Halevi [GGH15] which, however, could not yet be based on "'nice' hardness assumptions" [GGH15].

While it was known that low-level encodings of zeros allow for so-called "zeroizing" attacks on [GGH13a] (which there, for example, leads to the analog of the decision-linear problem ([BBS04]) not being hard), it was not clear if such attacks can also be mounted on the other candidates. Furthermore, it was unclear if low-level encodings of 0 could be obtained by the way in which the obfuscation schemes use the graded encoding schemes. For this note that, when using the branching program as described in Section 5.8, high-level encodings of zero can be obtained—an honest evaluation of a program can lead to an encoding of the identity matrix—but other than that no low-level encodings of zero need to be given to the adversary.

Earlier this year, Cheon et al. [CHL+15] presented a new and devastating "zeroizing" attack on [CLT13] which leads to a complete break of the cryptosystem, i.e., it allows to recover all secret parameters in polynomial time. Proposed countermeasures [BWZ14, GGHZ14] were shown to be ineffective [CLT14]. It was first hoped that the restricted use of graded encoding schemes in obfuscation candidates does not allow an adversary to learn low-level encodings of zeros. For this note that in the obfuscation setting all the elements are generated at setup time and unlike in the (full) multilinear maps scenario the adversary cannot request additional encodings of its choices. However, shortly after the first attacks were presented by Cheon et al., these were further extended by Gentry et al. [GHMS14] who showed that the absence of low-level encodings of zero does not necessarily protect against the attacks.

Post-zeroizing. Even though various candidate constructions for obfuscation are not know to be broken [BR14, BGK+14, AGIS14, MSW14, Zim15, AB15] (for this also note that various candidates have only been analyzed in the ideal graded encoding model) it is alarming that many security arguments either reduce to assumptions which are known to be false for the current candidates or are in an ideal model which does not model well actual real world constructions. To mitigate the latter, Gentry et al. [GHMS14] propose a new idealized graded encoding model where in particular the zero-test functionality leaks additional information beyond the fact whether or not the tested element is an encoding of zero. To again strengthen our confidence in obfuscation candidate constructions, Badrinarayanan et al. [BMSZ15] discuss *post-zeroizing obfuscation* and present a candidate construction which provably (in a generic model) does not allow an adversary to obtain encodings of zero which are at the heart of all known attacks. However, to truly gain confidence in the existence of general-purpose obfuscation, we require constructions based on primitives other than multilinear maps. First steps in this direction were made by Ananth and Jain [AJ15] and, independently, by Bitansky and Vaikuntanathan [BV15] who construct indistinguishability obfuscation from a general-purpose functional encryption scheme that has succinct ciphertexts and sub-exponential security. Furthermore, Canetti and Vaikuntanathan [CV13] construct iO based on black-box pseudo-free groups. Although these three constructions are based on primitives other than multilinear maps the downside is that they are based on primitives which we so far only know how to construct from iO itself.

Conclusion. All in all, the study of indistinguishability obfuscation and, in particular, the study of constructions for indistinguishability obfuscation is still in its early stages—the first candidate construction is roughly two years old—and, thus, it is too early to say how the chips may fall. At

the time of writing, the general belief seems to be that indistinguishability obfuscation is a plausible notion and that, given time, indistinguishability obfuscation may be constructed from a more diverse set of primitives. While this thesis is primarily a study of the (un)instantiability of random oracles, it is as much (if not more so) a study of indistinguishability obfuscation. We show many positive and negative implications of the existence of indistinguishability obfuscation and believe that especially results of the form "$\mathsf{iO} \implies \neg X$" (and its contrapositive $X \implies \neg\mathsf{iO}$), where X is some interesting primitive, allow us to gain a better understanding of indistinguishability obfuscation in its whole.

Chapter 6

Point-Function Obfuscation

> "Whenever possible, substitute constructions out of known entities for inferences to unknown entities."
>
> Bertrand Russell

Summary. In this chapter we introduce special-purpose obfuscators for the specific class of point functions which are zero everywhere except on a single point. We present various definitions, focusing on point-function obfuscation in the presence of auxiliary information.

Chapter content

6.1 Introduction

In the previous chapter we introduced powerful general-purpose obfuscation schemes that work for any polynomial sized circuit (or even any polynomial-time Turing machine). In this chapter, we consider special-purpose obfuscation schemes for a class containing only very simple functions, so called *point functions*. A *plain point function* (sometimes also referred to as *password function*) p_x for some value $x \in \{0,1\}^*$ is defined as:

$$p_x(s) := \begin{cases} 1 & \text{if } s = x \\ \perp & \text{o/w} \end{cases}$$

Such plain point functions—we often will drop the attribute plain and simply speak of point functions—can, for example, be used in an authentication scenario, given that the obfuscation is sufficiently strong and does not reveal the *point address* x. Consider x to be a user's password. Then, to authenticate one could use a point obfuscation of x and, thus, does not need to store x in the clear. For a different scenario, consider you publish a puzzle with a unique solution and in addition you

want to allow anyone to check their solution on their own without having to send it in. One way to implement this would be to obfuscate the point function p_{solution} which outputs 1 on input the correct solution and $\perp$ otherwise. If sufficiently obfuscated (for example, with a VBB obfuscator) this program should leak nothing of the solution unless the solution is already known.

Multi-bit output point functions. Besides plain point functions we also consider an extension that on the first glance does not seem to have a huge impact (which it, however, does at least once we consider the auxiliary information setting). A *multi-bit output point function* is identical to a plain point function except that it does not only return a single bit but rather on input the correct point we allow it to return a bit-string, usually of polynomial length. A multi-bit output point function $p_{x,m}$ where we call x the *point address* and m the *point message* (or *point value*) is defined as:

$$p_{x,m}(s) := \begin{cases} m & \text{if } s = x \\ \perp & \text{o/w} \end{cases}$$

Assume that we have an obfuscator MBPO for a multi-bit output point function such that on input x and m it returns an obfuscated point function

$$\overline{p}_{x,m} \leftarrow\!\!\$\, \mathsf{MBPO}(x, m).$$

Intuitively, this setting is closely related to symmetric encryption, because if $\overline{p}_{x,m}$ hides x and m, then one can recover m only with knowledge of the point address (secret key) x. Canetti, Tauman-Kalai, Varia, and Wichs [CKVW10] study the relationship between obfuscation for multi-bit output point functions and symmetric encryption and show tight connections between these primitives. In particular, they show how symmetric encryption of various strength (CPA security, KDM security, fully weak keys, auxiliary input, etc.) lead to constructions of obfuscators for multi-bit output point functions and vice versa. One can define the encryption operation of a symmetric encryption scheme for a key k and message m as $\mathsf{Enc}_{\mathsf{k}}(m) := \mathsf{MBPO}(\mathsf{k}, m)$. Correspondingly, decryption interprets a ciphertext c as a circuit and runs it on the key, that is, $\mathsf{Dec}_k(c) := c(k)$. Similarly, to obfuscate a point (k, m) given a symmetric encryption scheme one encrypts $c \leftarrow\!\!\$\, \mathsf{Enc}_{\mathsf{k}}(m)$ and additionally constructs a "decryption program" with ciphertext c hard-coded and which on input a key tries to decrypt c with that key.[1] Note that while, ideally, we would like a point obfuscator to be "good" for any point x a symmetric encryption scheme only needs to offer security if keys are chosen uniformly at random. Additionally, symmetric encryption schemes do not necessarily offer security if the message is chosen to depend on the key and hence for a point obfuscation derived from such an encryption scheme we must ensure that point address x and point message m are chosen from specific distributions: for example, x could be the uniform distribution and m should not depend on x. As mentioned, one option is to consider stronger requirements on the encryption scheme. Another is to derive point obfuscation techniques via different means. In the following we present a brief abstract on the history of point obfuscation and informally introduce the settings that we are most interested in this thesis: point obfuscation in the presence of auxiliary information and composable point obfuscation. Then,

[1] In order to obtain a correct obfuscation scheme one needs to start from an encryption scheme which allows for detection of decryption with invalid keys. Canetti et al. show that this property is without loss of generality and can be added generically to any semantically secure encryption scheme [CKVW10].

in Section 6.3 we give formal definitions for various notions of point obfuscators and in Section 6.5 discuss candidate constructions.

A note for the impatient reader. As in the previous chapter with indistinguishability obfuscation, we try to present point-function obfuscation in a broader context. We note that the most important definitions are that of AIPO (point obfuscation in the presence of auxiliary information) and its multi-bit output variant MB-AIPO. We present these in Section 6.3.

6.2 An Introduction to Obfuscating Point Functions

The study of point obfuscation begins with Canetti [Can97], although in a slightly different context. Canetti studied random oracles and tried to identify and define the useful properties of a random oracle, to then find realizations in the standard model for these properties. As a first step he proposed a primitive called *oracle hashing* that, similarly to a random oracle, should hide all partial information on an input. As a candidate construction for such a hash function H, Canetti proposed the following: let $\mathbb{G}_\lambda$ be a group of prime order $q_\lambda \in (2^{\lambda-1}, 2^\lambda)$. Then, on input $x \in \mathbb{Z}_q$ and randomness $r \in \mathbb{G}_\lambda$ we define the output of $\mathsf{H}(x; r)$ to be:[2]

$$\mathsf{H}(x; r) := (r, r^x).$$

Given Canetti's oracle hashing function H we can construct a point obfuscation scheme PO which stores value (r, r^x) and on input s computes r^s and compares this to r^x. As Canetti was interested in showing that H hides all partial information on an input, he, along the way, showed that the point obfuscation constructed from it is a good point obfuscation and we will later give precise security definitions.

A second prominent construction of point obfuscators is due to Wee [Wee05] who presented his construction based on the existence of strong one-way permutations in 2005. Both obfuscation schemes achieve a variant of VBB obfuscation—note that while VBB obfuscation is impossible in general it may exist for restricted classes of functions, in particular, it may exist for point functions. Note also that for the moment, when we speak of VBB obfuscation for point functions we consider this to be without auxiliary information, that is, the auxiliary information aux given to adversary and simulator in Definition 5.2 is always the empty string.[3] However, under sufficient assumptions, both obfuscation schemes can be shown to achieve security in the presence of auxiliary information, that is, the point obfuscator hides point x even if an adversary is given certain information about x.

Obfuscating multi-bit output point functions. The study of multi-bit output point obfuscation was started in 2003 by Lynn, Prabhakaran, and Sahai [LPS04] who showed how to obfuscate multi-bit output point functions in the random oracle model. As seen already in Section 4.6.1 we can obfuscate a multi-bit output point function (x, m) by computing

[2] In his original paper Canetti did not consider point obfuscation but hashing and in fact considered a slight variation of the construction where point x is first hashed with a collision resistant hash function, that is, he defined $\mathsf{H}(x; r) := (r, r^{h(x)})$ for some collision resistant function h. We note that the security proofs given are, however, foremost for the presented construction without an additional application of a collision resistant hash function.

[3] Note that for VGB obfuscation the auxiliary information setting is identical to the setting without auxiliary information (see Proposition 5.4 on page 76).

$\underline{\mathsf{MBPO}(x, m)}$

$\overline{x} \leftarrow \mathsf{RO}(0\|x)$

$\overline{m} \leftarrow \mathsf{RO}(1\|x) \oplus m$

return $(\overline{x}, \overline{m})$

Given $(\overline{x}, \overline{m})$ and knowing value x allows to recover m by computing $\overline{m} \oplus \mathsf{RO}(1\|x)$. Furthermore to check whether x is correct one can test whether $\mathsf{RO}(0\|x)$ is equal to $\overline{x}$. In contrast to the point obfuscations by Canetti and Wee this multi-bit point obfuscation in general is not perfectly correct unless we assume that the random oracle is injective.

Multi-bit point obfuscators and composability. Canetti and Dakdouk [CD08] study multi-bit point obfuscators in the standard model and show that composable plain point obfuscators (i.e., obfuscators that remain secure even if an adversary sees multiple obfuscations) imply composable multi-bit point obfuscators (and vice versa). In particular, Canetti and Dakdouk give a generic construction of a MBPO from a simple point obfuscator: to obfuscate a point function $p_{x,m}$ with $|m| = t$ we construct $t + 1$ point obfuscations, that is, we obfuscate $p_{x,m}$ bit by bit:

$\underline{\mathsf{MBPO}(x, m)}$

$\overline{\mathbf{p}}[1] = \mathsf{PO}(x)$

for $i = 1 \ldots, |m|$ **do**

  if $m[i] = 0$ **then**

    $y \leftarrow\!\!\$\ \{0,1\}^{|x|} \setminus \{x\}$

    $\overline{\mathbf{p}}[i+1] \leftarrow\!\!\$\ \mathsf{PO}(y)$

  else

    $\overline{\mathbf{p}}[i+1] \leftarrow\!\!\$\ \mathsf{PO}(x)$

return $\overline{\mathbf{p}}$

For each bit of m that is set to 1 an obfuscation of x is generated and for any 0-bit an obfuscation of a random value is output. Knowing the point address x then allows to easily recover m.

One might think that a good point obfuscator directly yields a good multi-bit output point obfuscator via the above construction but this intuition is flawed. A good point obfuscator should be one-way (we formally prove this in Section 6.4) as otherwise given $\overline{p} \leftarrow\!\!\$\ \mathsf{PO}(x)$ for a random x one can recover a preimage which by the functionality requirement contains x and hence the obfuscation does not properly hide x. Let PO be a point obfuscator. We define an adapted obfuscator PO' which in addition to the original point obfuscation outputs a hardcore bit of the input

$\underline{\mathsf{PO}'(x)}$

$\overline{x} \leftarrow\!\!\$\ \mathsf{PO}(x)$

$r \leftarrow\!\!\$\ \{0,1\}^{|x|}$

$b \leftarrow \langle x, r \rangle$

return $(\overline{x}, r, b)$

It is easily seen that the above is a good point obfuscator for points chosen at random: we have already argued that PO must be one way and thus what we leak in addition is nothing but the

Goldreich-Levin hardcore bit (also see Section 4.3.2). Canetti and Dakdouk even prove the stronger statement that if PO is a VBB point obfuscator then so is PO' [CD08]. Plugged into in the above construction of Canetti and Dakdouk we do not end up with a secure MBPO even if we require that the point address x is chosen uniformly at random. For this note, that if m is long enough and has sufficiently many 1 bits then with high probability one is able to uniquely reconstruct x.

Composable obfuscators. Composability of obfuscators is a strong requirement that was first studied by Lynn et al. [LPS04] who also showed that in the random oracle model there exists an obfuscator which is not even 2-(self)-composable—self-composability refers to obfuscators which are still secure if an adversary sees multiple obfuscations of the same point (resp. circuit). Furthermore, they conjectured that this property carries over to the standard model. Indeed, while we do have candidate constructions for strong (VBB-like) point obfuscators in the standard model we do not have constructions for composable point obfuscators that meet the same security guarantee. Jumping ahead, in this thesis we show that composable VBB-point obfuscation with auxiliary information (more precisely, composable AIPO with computationally hard-to-invert auxiliary information) does not exist if indistinguishability obfuscation exists. However, if we are willing to leave the VBB setting and settle for weaker obfuscators we do have a candidate construction. Bitansky and Canetti [BC14] show that the point obfuscator due to Canetti is a t-composable virtual grey-box (VGB; cf. Definition 5.4) point obfuscator under the (non-standard) t-Strong Vector Decision Diffie–Hellman assumption. They go on to show that their obfuscator can be used together with the construction of Canetti and Dakdouk [CD08] to also obtain a VGB multi-bit output point obfuscator and if composability is not necessary, then that this construction even yields a VBB-secure MBPO.

Point obfuscation in the presence of auxiliary inputs. The notion of point obfuscation that we are mostly interested in is (composable) point obfuscation in the presence of auxiliary information, that is, we consider point obfuscators that construct point obfuscations $\overline{p}_x$ (resp. multi-bit output point obfuscations $\overline{p}_{x,m}$) which hide x even if some information on point x (and possible message m) is given. Naturally, if sufficient information on x is given such that x can simply be guessed, we cannot hope to hide x. Thus, what we require is that if the auxiliary information aux itself hides x, or in other words, if x is unpredictable given the auxiliary information aux then an obfuscation $\overline{p}_x$ should also hide x in the presence of aux. We consider two variations of the above: auxiliary information which is *computationally* hard-to-invert and auxiliary information that is *statistically* hard-to-invert. Here computationally hard-to-invert means that no efficient predictor can successfully guess x with non-negligible probability given auxiliary information aux. Statistically hard-to-invert considers a stronger requirement where the predictor is allowed to run in unbounded time. While computationally hard-to-invert auxiliary information, thus, might information-theoretically contain x (for example, an encryption of x) in the statistical setting we require that x has still super-logarithmic min-entropy given the auxiliary information.

As with general-purpose obfuscators we can extend any point obfuscation setting with auxiliary inputs. While the VBB and VGB obfuscation notions are simulation based—anything an adversary can do with the code and the auxiliary information an (unbounded) simulator can do with access to the auxiliary information and oracle access to the functionality—we consider a weaker indistinguishability based notion introduced by Canetti [Can97] and later termed *distributional AIPO* (Auxiliary Input Point Obfuscator) by Bitansky and Paneth [BP12].

Instead of using distributions we choose a formalization with explicit sample algorithms for AIPOs. If Sam is a (statistically/computationally) unpredictable auxiliary-input point sampler—Sam is a PPT algorithm that on input the security parameter samples a point x together with auxiliary information aux—then we require from an obfuscator that the following distributions are computationally indistinguishable:

$$\left(\begin{array}{l}(x,\mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)\\ \\ \overline{p} \leftarrow\!\!\$\ \mathsf{PO}(x)\\ \textbf{return}\ (\overline{p},\mathsf{aux})\end{array}\right) \approx_c \left(\begin{array}{l}(x,\mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)\\ u \leftarrow\!\!\$\ \{0,1\}^{|x|}\\ \overline{p} \leftarrow\!\!\$\ \mathsf{PO}(u)\\ \textbf{return}\ (\overline{p},\mathsf{aux})\end{array}\right)$$

That is, on the one hand we consider the distribution which obfuscates the point as output by sampler Sam and on the other we consider the obfuscation of a random point.

Composable point obfuscation and auxiliary inputs. Putting it all together we arrive at composable AIPO either with computationally or with statistically hard-to-invert auxiliary information. For the computational case we show that composable AIPO does not exist assuming that general-purpose indistinguishability obfuscators exist (Chapter 7). However, even if only non-composable AIPO for computationally hard-to-invert auxiliary information exists (and we have several candidate constructions) this allows for interesting applications: we construct the first universal hardcore function with long output in Chapter 11. If we in turn consider statistically hard-to-invert auxiliary information then it can be shown that such AIPOs are implied by composable VGB point obfuscation [MH14a] and again we have a candidate construction due to Bitansky and Canetti [BC14]. We will use such composable point obfuscators in various constructions, for example, to achieve the first construction of fully secure q-query correlation-input secure hash functions (also see Section 4.2).

6.3 Defining Point Obfuscation

We start defining the functionality requirements for point (and multi-bit output point) obfuscators. For simplicity we assume the circuit model of computation, that is, that the program that the obfuscator outputs is a circuit description. If we now consider point circuits C_x for a point $x \in \{0,1\}^*$ (resp. multi-bit output point circuits $C_{x,m}$) where the circuits are given in a *canonical* encoding that allows to efficiently extract x (resp. x and m) we can define point obfuscators (PO) and multi-bit output point obfuscators by restricting the class of circuits in Definition 5.1 (page 71) to such canonical point circuits.

For this, as a first step, we formally define point functions as well as point circuits and fix such a canonical encoding.

Definition 6.1 (Point functions and point circuits with a canonical encoding)**.** *A binary function $p : \{0,1\}^* \to \{0,1\}$ is called a point function if there exists exactly one $x \in \{0,1\}^*$ such that*

$$p(s) := \begin{cases} 1 & \text{if } s = x \\ 0 & \text{o/w} \end{cases}$$

We call x the point address (or point) of point function p.

A circuit C is called a point circuit *if it computes a point function. We fix a canonical encoding for point circuits and identify with the binary string $x \in \{0,1\}^*$ the point circuit C that computes a point function with point address x by applying its input to a single equality gate.*[4] *Let $\mathcal{C}^{\mathsf{p}} = \{\mathcal{C}^{\mathsf{p}}_\lambda\}_{\lambda\in\mathbb{N}}$ denote the ensemble of all point circuits presented in the canonical encoding where $\mathcal{C}^{\mathsf{p}}_\lambda$ contains all point circuits for points of length λ.*

This now allows us to capture point obfuscators by restricting the class of circuits in Definition 5.1 to the canonical class of point circuits $\mathcal{C}^{\mathsf{p}}$.

Definition 6.2 (Point obfuscator)**.** *A* PPT *algorithm* PO *is called a* point obfuscator *(PO) if it satisfies the functionality preserving and polynomial slowdown requirements of Definition 5.1 for the class of point circuits $\mathcal{C}^{\mathsf{p}}$.*

Similarly we can define multi-bit output point functions $p_{x,m}$ as

$$p(s) := \begin{cases} m & \text{if } s = x \\ 0 & \text{o/w} \end{cases}$$

and multi-bit output point circuits with a canonical encoding. Let the ensemble $\mathcal{C}^{\mathsf{mbp}} = \{\mathcal{C}^{\mathsf{mbp}}_\lambda\}_{\lambda\in\mathbb{N}}$ contain all poly-sized multi-bit output point circuits, represented in a canonical encoding.[5] For multi-bit output point circuits we may further want to restrict the class to contain only circuits where point address length and message match a given length. If $\mathsf{al} : \mathbb{N} \to \mathbb{N}$ and $\mathsf{ml} : \mathbb{N} \to \mathbb{N}$ are two polynomials representing point address and point message length, then we denote by $\mathcal{C}^{\mathsf{mbp}}\big|_{\mathsf{al},\mathsf{ml}}$ the restricted class of point circuits defined as

$$\mathcal{C}^{\mathsf{mbp}}\big|_{\mathsf{al},\mathsf{ml}} := \left\{ C : C \in \mathcal{C}^{\mathsf{mbp}}_\lambda \wedge \exists! x \in \{0,1\}^{\mathsf{al}(\lambda)} : \exists! m \in \{0,1\}^{\mathsf{ml}(\lambda)} : C(x) = m \right\}_{\lambda\in\mathbb{N}}.$$

In other words $\mathcal{C}^{\mathsf{mbp}}\big|_{\mathsf{al},\mathsf{ml}}$ denotes the class of multi-bit output point circuits restricted to match a given bound on address and message length. Note that while for plain point obfuscators we may require that an obfuscator works for any point $x \in \{0,1\}^*$, multi-bit output point obfuscators may only work for pairs (x, m) that are of appropriate lengths, for example, $|x| = |m|$. For this consider the examples from the introduction: the random oracle MBPO of Lynn et al. [LPS04] requires that the point message matches the output length of the random oracle. Canetti and Dakdouk's compiler [CD08], on the other hand, can cope with arbitrarily long messages given that a composable plain point obfuscator exists. Jumping ahead, as we will present a negative result on the existence of multi-bit point obfuscators in this thesis, the result only gets stronger the more restricted the construction that we rule out.

Definition 6.3 (Multi-bit output point obfuscator)**.** *A* PPT *algorithm* MBPO *is called a* multi-bit output point obfuscator *(MBPO) if it satisfies the functionality preserving and polynomial slowdown requirements of Definition 5.1 for the class of multi-bit output point circuits $\mathcal{C}^{\mathsf{mbp}}$.*

[4] We can implement such an equality gate with a tree-structured AND-circuit.

[5] Technically, poly-sized is not well defined in this case since we require both a bound on the address and the message. We would like that $\bigcup_\lambda \mathcal{C}^{\mathsf{mbp}}_\lambda$ contains any possible multi-bit output point function and could, thus, for example, require that $\mathcal{C}^{\mathsf{mbp}}_\lambda$ contains all multi-bit output point functions where both address and message have a length that is smaller or equal to λ.

We call MBPO *an MBPO with point address length* $\mathsf{al} : \mathbb{N} \to \mathbb{N}$ *and point message length* $\mathsf{ml} : \mathbb{N} \to \mathbb{N}$*, where* al *and* ml *are polynomials, if it satisfies the functionality preserving and polynomial slowdown requirements of Definition 5.1 for the class of all multi-bit output point circuits* $\mathcal{C}^{\mathsf{mbp}}\big|_{\mathsf{al},\mathsf{ml}}$.

Remark. To make explicit that the obfuscator gets a representation that allows to recover the plain point (or point and message) we write $\mathsf{PO}(x)$ and $\mathsf{MBPO}(x, m)$ instead of $\mathsf{PO}(C_x)$ and $\mathsf{MBPO}(C_{m,x})$. In other words we require from a point obfuscator that on input a point x (resp. point address x and message m) it outputs a description of a point circuit $\overline{p}$ (resp. multi-bit output point circuit) and that the size of the obfuscated circuit is polynomially bounded by the size of the input point (resp. point and message).

Definition 5.1 and thus similarly our above definitions of point obfuscators capture what we would like functionality-wise from point obfuscation. It does not capture any form of security guarantee, i.e., an obfuscator that simply echos the input point fulfills the definition. Having defined point obfuscators in this way, we can, however, directly capture security by restricting general-purpose security notions such as the simulation-based VBB (see Definition 5.2) or VGB (see Definition 5.4) notions. While VBB and VGB are very strong notions of obfuscation we consider weaker indistinguishability based notions that were first formalized by Canetti [Can97] and later termed *distributional AIPO* (Auxiliary Input Point Obfuscator) by Bitansky and Paneth in their study of three-round weak zero-knowledge protocols for NP [BP12]. We introduce AIPOs[6] in Section 6.5 but first discuss composability of obfuscators.

6.3.1 Composable Obfuscation

So far all our obfuscation notions only considered the obfuscation of a single circuit or a single point function. A natural question to ask is whether the scheme remains secure even if the adversary is allowed to see multiple obfuscations, possibly of related points. This leads to the study of composition of obfuscators. The version we consider in this work is composition by concatenation formalized by Lynn, Prabhakaran, and Sahai [LPS04] which we define here in the auxiliary inputs setting and for general circuits rather than for point functions only:

Definition 6.4 (t-composable obfuscation [LPS04])**.** *A* PPT *machine* O *is a t-composable obfuscator for a circuit ensemble $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ if it satisfies the functionality and polynomial slow-down requirements, as in Definition 5.1, and for any* PPT *distinguisher* D *and polynomial* p*, there is a simulator* Sim*, such that for any list of circuits $C^1, \ldots, C^t \in C_\lambda$ (where $t = \mathsf{poly}(\lambda)$), any $\mathsf{aux} \in \{0,1\}^{\mathsf{poly}(\lambda)}$, and sufficiently large λ:*

$$\Big|\Pr\big[\mathcal{A}(\mathsf{O}(C^1), \ldots, \mathsf{O}(C^t), \mathsf{aux}) = 1\big] - \Pr\Big[\mathsf{Sim}^{C^1,\ldots,C^t}(1^{|C^1|}, \ldots, 1^{|C^t|}, \mathsf{aux}) = 1\Big]\Big| \leq \frac{1}{\mathsf{p}(\lambda)}.$$

We speak of a t-composable VBB *obfuscator if the above holds for simulators restricted to be* PPT*. We speak of a t-composable* VGB *obfuscator when restricting the number of oracle queries a simulator can make to be polynomial but otherwise allow the simulator to run in unbounded time. If an obfuscator is t-composable for any t we call it* composable*.*

[6] Note that we also defined VBB and VGB in the auxiliary input setting. Even though the name does not capture this, but the main difference is that it is not a simulation based notion but a weaker indistinguishability based notion.

Remark. Note that the above formalization of composable VBB is somewhat weaker than our definition of VBB security for general-purpose obfuscators (Definition 5.2) as here we allow the simulator to depend on the distinguishing probability, that is, for any polynomial there exists a simulator, whereas Definition 5.2 requires a simulator to exist for all polynomials. This is a relaxation often seen for point function constructions mainly because we have constructions that achieve this level of security (for example, the constructions of Canetti and Wee [Can97, Wee05] can be shown to achieve this level of non-composable VBB without auxiliary information). We note that both forms imply the notion of point obfuscation that we mostly consider in this thesis, an indistinguishability-based notion called AIPO. We show this implication in Section 6.4.

While [LPS04] consider t-composability in the virtual black-box setting, we here only require the relaxed VGB setting, that is, we allow the simulator to run in unbounded time. A nice side effect of working in the VGB setting is that this gives us auxiliary input security for free as (worst-case) VGB security implies (worst-case) VGB security in the presence of auxiliary information [BC14] (also see Section 5.4.1).

Constructing composable point obfuscation. Bitansky and Canetti show that the point obfuscation scheme of Canetti [Can97] is a t-composable VGB point obfuscator under the t-Strong Vector Decision Diffie–Hellman assumption [BC14]. Note that as we can first compose and then introduce auxiliary input, this implies that under the t-Strong Vector Decision Diffie–Hellman assumption Canetti's obfuscation scheme is also a VGB-AI point obfuscator. We recall the scheme by Canetti [Can97] in Section 6.5.

6.3.2 Point Obfuscation in the Presence of Auxiliary Information

We now present the definitions of point obfuscation that we will be using in this thesis. These are somewhat weaker than VBB or VGB obfuscation-based definitions and can come either in a composable or in a non-composable setting. Figure 6.1 provides an overview over the various notions and their relationship to one another.

We consider point obfuscators that on input a point $x \in \{0,1\}^*$ need to construct a point circuit $\overline{p}_x$ such that if the point was sampled by an unpredictable auxiliary-input point sampler then $\overline{p}_x$ hides x. We have already encountered unpredictable distributions, for example, in our discussion of correlated-input secure hash functions (see Definition 4.1 on page 48). Here we consider an extended variant where the distribution is not only over points x but also over dependent auxiliary information aux.[7] To make this more explicit we consider sample algorithms that on input some randomness and the security parameter sample a point x and auxiliary information aux together. For simplicity we also restrict samplers to be efficient.

We consider two types of unpredictability: *computational* unpredictability and *statistical* unpredictability. The first requires that given the auxiliary information no efficient predictor can predict point x while the latter allows the predictor to even run in unbounded time. For statistical unpredictability an equivalent definition is that the point x must have super-logarithmic min entropy conditioned on the auxiliary information. We here define the game based variant with a bounded/unbounded predictor as we feel that this presentation more explicitly captures the intuition.

[7] The definition can be traced back to [Can97] and the name distributional AIPO (Auxiliary Input Point Obfuscator) was introduced by [BP12]. We will drop the prefix distributional and simply speak of AIPO.

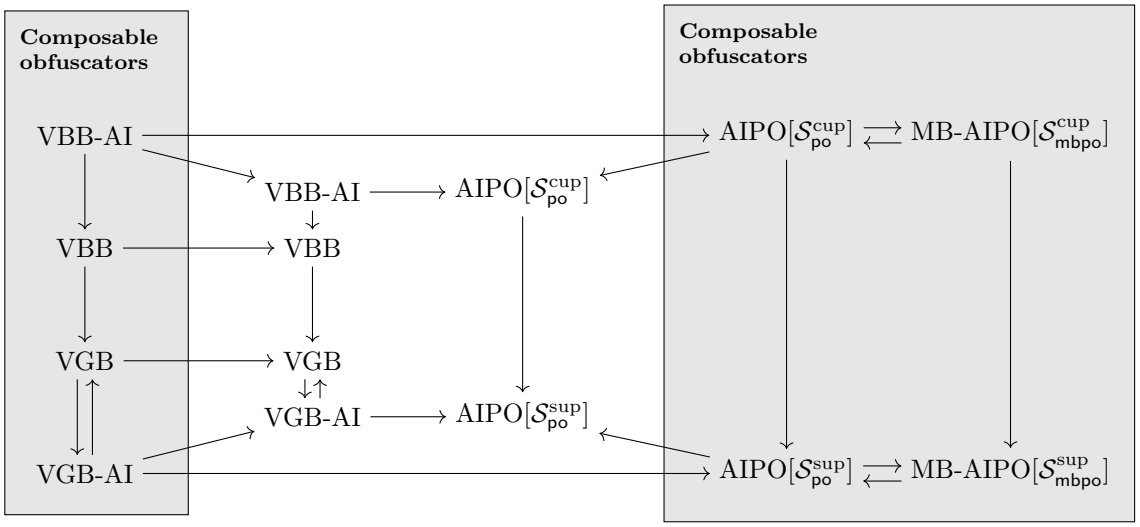


Figure 6.1: The picture provides an overview over various notions of point obfuscation and how they are related. The notions on highlighted background are composable notions. An arrow from notion A to notion B denotes that the existence of A implies the existence of B. For example, composable VBB obfuscation in the presence of auxiliary information (composable VBB-AI) implies non-composable VBB-AI as well as composable AIPO for computationally hard-to-invert auxiliary information (AIPO[$\mathcal{S}_{\mathsf{po}}^{\mathrm{cup}}$]).

Definition 6.5 (Unpredictable auxiliary-input point sampler)**.** *An algorithm* Sam *that on input the security parameter* 1^λ *outputs two strings* (x, aux) *is called a* computationally unpredictable *(resp.* statistically unpredictable*)* auxiliary-input point sampler *if no* PPT *(resp. unbounded) algorithm can predict* x *from* aux*. That is, for every* PPT *(resp. unbounded) algorithm* P *and for all large enough* λ*:*

$$\Pr_{(x,\mathsf{aux}) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda)} \left[\mathsf{P}(1^\lambda, \mathsf{aux}) = x\right] \leq \mathsf{negl}(\lambda)\,.$$

We let $\mathcal{S}_{\mathsf{po}}^{\mathrm{cup}}$ *denote all efficient computationally unpredictable auxiliary-input point samplers and denote by* $\mathcal{S}_{\mathsf{po}}^{\mathrm{sup}}$ *all efficient statistically unpredictable auxiliary-input point samplers.*

For simplicity we will often refer to a *unpredictable auxiliary-input point sampler* as simply an unpredictable sampler or unpredictable distribution when it is clear from the context what is sampled.

Next we define point obfuscators secure in the presence of auxiliary inputs relative to a class of samplers $\mathcal{S}_{\mathsf{po}}$. A point obfuscator AIPO is called secure for a sampler $\mathsf{Sam} \in \mathcal{S}_{\mathsf{po}}$ if no efficient distinguisher D can distinguish between being given an obfuscation $\overline{p}_x$ and auxiliary information aux for (x, aux) as sampled by Sam, or if it is given $\overline{p}_u$ and aux for a uniformly random point u. We write $\mathsf{AIPO}[\mathcal{S}_{\mathsf{po}}]$ to capture the class of obfuscators that are secure against all samplers in $\mathcal{S}_{\mathsf{po}}$.

Definition 6.6 (Auxiliary-input point obfuscation (AIPO))**.** *A* PPT *point obfuscator* AIPO *is called* secure under auxiliary input for auxiliary-input point samplers in class $\mathcal{S}_{\mathsf{po}}$ *if it satisfies the following secrecy property: for any sampler* $\mathsf{Sam} \in \mathcal{S}_{\mathsf{po}}$ *it holds for any* PPT *distinguisher* D *that the advantage* $\mathsf{Adv}^{\mathrm{po}}_{\mathsf{AIPO},\mathsf{Sam},\mathsf{D}}(\lambda)$ *is negligible:*

$$\mathsf{Adv}^{\mathrm{po}}_{\mathsf{AIPO},\mathsf{Sam},\mathsf{D}}(\lambda) = 2{\cdot}\Pr\left[\mathsf{PO}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{AIPO}}(\lambda) = 1\right] - 1 \leq \mathsf{negl}(\lambda)$$

$\underline{\mathsf{PO}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{AIPO}}(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\}$
$(x_0, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
$x_1 \leftarrow\!\!\$\ \{0,1\}^{|x_0|}$
$\overline{p} \leftarrow\!\!\$\ \mathsf{AIPO}(x_b)$
$b' \leftarrow\!\!\$\ \mathsf{D}(1^\lambda, \overline{p}, \mathsf{aux})$
return $b = b'$

The probability is over the coins of adversary $(\mathsf{Sam}, \mathsf{D})$*, the coins of* AIPO*, and the choices of* x_1 *and* b*. We denote by* $\mathsf{AIPO}[\mathcal{S}_{\mathsf{po}}]$ *the class of all* PPT *point obfuscators secure against samplers in* $\mathcal{S}_{\mathsf{po}}$*.*

Composable AIPOs. We can define a composable variant of the above definition by considering samplers that output a vector $\mathbf{x}$ rather than a point x. Computational (resp. statistical) unpredictability then requires that for all PPT (resp. unbounded) algorithms P and for all large enough λ:

$$\Pr\left[\mathbf{x}[i] = x \ :\ (\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda);\ (x, i) \leftarrow\!\!\$\ \mathsf{P}(1^\lambda, \mathsf{aux})\right] \leq \mathsf{negl}(\lambda)\,.$$

Hence, the AIPO security game would change accordingly to

$\underline{\text{Composable } \mathsf{PO}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{AIPO}}(\lambda)}$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$
for $i = 1, \ldots, |\mathbf{x}|$ **do**
  if $b = 1$ **then**
    $\mathbf{x}[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{x}[i]|}$
  $\mathbf{p}[i] \leftarrow\!\!\$\ \mathsf{AIPO}(\mathbf{x}[i])$
$b' \leftarrow\!\!\$\ \mathsf{D}(1^\lambda, \mathbf{p}, \mathsf{aux})$
return $b = b'$

6.3.3 Indistinguishable Point Obfuscation

The AIPO notions can be naturally weakened to obtain an *indistinguishability-style* notion of point obfuscation. For this we simply require that the point sampler outputs the empty string ε as auxiliary information. Formally we capture this class by $\mathcal{S}^{\not{\mathsf{aux}}}_{\mathsf{po}}$.

Definition 6.7. *An algorithm* Sam *that on input the security parameter* 1^λ *outputs one string and the empty string* (x, ε) *is called* empty auxiliary information point sampler*. We denote the class of all such point samplers by* $\mathcal{S}^{\not{\mathsf{aux}}}_{\mathsf{po}}$*.*

This allows us to capture indistinguishable point obfuscation by considering the class of $\mathrm{AIPO}[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\not{\mathsf{aux}}}_{\mathsf{po}}]$ secure obfuscators.

6.3.4 Obfuscation for Point Functions with Multi-Bit Output

Similar to AIPO we can define an analogous notion MB-AIPO for multi-bit output point functions via an unpredictable sampler—the notion was introduced by Matsuda and Hanaoka [MH14a] in an

average-case formulation called AIND-δ-cPUAI—where sampler Sam now samples a tuple (x, m, aux) defining MBPF $p_{x,m}$ and providing auxiliary information on x and m. Again, we require that it is computationally/statistically infeasible to recover the point address x given auxiliary information aux. In particular this means that we allow aux to contain the point message m in the clear. From an MB-AIPO obfuscator we now require that the obfuscation of $p_{x,m}$ is indistinguishable from an obfuscation with a changed point value m' where m' is chosen uniformly at random. Intuitively this captures that the obfuscation does not reveal any information about the point value m. For this note that the auxiliary information aux may contain m and still it should be infeasible to learn if the obfuscation $\overline{p}$ has point value m or a different point value m'. If we think of $\overline{p}$ as an encryption of a message m under a key x then this requirement captures the indistinguishability of encryptions. We note that also other definitional choices are possible here, which we discuss after presenting the formal definition.

Definition 6.8 (Unpredictable auxiliary-inputs multi-bit output point sampler)**.** *An algorithm* Sam *that on input the security parameter* 1^λ *outputs* (x, m, aux) *is called a* computationally unpredictable *(resp.* statistically unpredictable*)* auxiliary-inputs multi-bit output point sampler *if no* PPT *(resp. unbounded) algorithm can predict* x *from* aux*. That is, for every* PPT *(resp. unbounded) algorithm* P *and for all large enough* λ*:*

$$\Pr_{(x,m,\mathsf{aux}) \leftarrow\$ \mathsf{Sam}(1^\lambda)} \left[\mathsf{P}(1^\lambda, \mathsf{aux}) = x\right] \leq \mathsf{negl}(\lambda)\,.$$

We let $\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$ *denote all efficient computationally unpredictable auxiliary-input multi-bit output point samplers and denote by* $\mathcal{S}^{\mathrm{sup}}_{\mathsf{mbpo}}$ *all efficient statistically unpredictable auxiliary-input multi-bit output point samplers.*

As discussed in Section 6.3, multi-bit output point obfuscators may be restricted in terms of what point message lengths they can process (see also Definition 6.3). We next define multi-bit output AIPOs in the restricted setting as we will later show an impossibility result and the more we can restrict what a construction needs to achieve the stronger such an impossibility result will be.

Definition 6.9 (Auxiliary-inputs multi-bit output point obfuscation (MB-AIPO))**.** *Let* MB-AIPO *be a* PPT *multi-bit output point obfuscator with point address length* $\mathsf{al} : \mathbb{N} \to \mathbb{N}$ *and message length* $\mathsf{ml} : \mathbb{N} \to \mathbb{N}$*, where* al *and* ml *are polynomials. The scheme* MB-AIPO *is called* secure under auxiliary input for auxiliary-input multi-bit output point samplers in class $\mathcal{S}_{\mathsf{mbpo}}$ *if it satisfies the following secrecy property: for any sampler* $\mathsf{Sam} \in \mathcal{S}_{\mathsf{mbpo}}$ *it holds for any* PPT *distinguisher* D *that the advantage* $\mathsf{Adv}^{\mathrm{mbpo}}_{\mathsf{MB\text{-}AIPO},\mathsf{Sam},\mathsf{D}}(\lambda)$ *is negligible:*

$$\mathsf{Adv}^{\mathrm{mbpo}}_{\mathsf{MB\text{-}AIPO},\mathsf{Sam},\mathsf{D}}(\lambda) = 2 \cdot \Pr\left[\mathsf{MBPO}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{MB\text{-}AIPO}}(\lambda)\right] - 1 \leq \mathsf{negl}(\lambda)$$

$\underline{\mathsf{MBPO}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{MB\text{-}AIPO}}(\lambda)}$

$b \leftarrow\$ \{0,1\}$

$(x, m_0, \mathsf{aux}) \leftarrow\$ \mathsf{Sam}(1^\lambda)$

$m_1 \leftarrow\$ \{0,1\}^{\mathsf{ml}(\lambda)}$

$\overline{p} \leftarrow\$ \mathsf{MB\text{-}AIPO}(x, m_b)$

$b' \leftarrow\$ \mathsf{D}(1^\lambda, \overline{p}, \mathsf{aux})$

return $b = b'$

The probability is over the coins of adversary $(\mathsf{Sam}, \mathsf{D})$*, the coins of* MB-AIPO*, and the choices of* m_1 *and* b*. We denote by* $\mathsf{MB\text{-}AIPO}[\mathcal{S}_{\mathsf{mbpo}}]$ *the class of all* PPT *point obfuscators secure against distributions in* $\mathcal{S}_{\mathsf{mbpo}}$*.*

Definition 6.9 was used in [MH14a] and is implied by the notion of composable AIPO [CD08] that is used in [BP12].[8] Interestingly, an MB-AIPO as by the above definition is not necessarily also AIPO secure. For this note that given an obfuscation scheme MB-AIPO we can construct an $\mathsf{MB\text{-}AIPO}'$ such that $\mathsf{MB\text{-}AIPO}'(x, m)$ works like MB-AIPO, but additionally, outputs the first bit of x. In other words, we define

$$\mathsf{MB\text{-}AIPO}'(x, m; r) := x[1] \| \mathsf{MB\text{-}AIPO}(x, m; r)$$

Leaking a single bit if x does not hurt the security of $\mathsf{MB\text{-}AIPO}'$ as given in the above definition. However, the "natural" construction of an AIPO from $\mathsf{MB\text{-}AIPO}'$ defined as

$$\mathsf{AIPO}(x; r) := \mathsf{MB\text{-}AIPO}'(x, 1; r)$$

does not yield a secure AIPO obfuscator. For this consider the adversary $(\mathsf{Sam}, \mathsf{D})$ where Sam samples only strings x that begin with 1. Given an obfuscation $\mathsf{AIPO}(x')$ for some point x' distinguisher D now learns the first bit of x' and outputs it. If x' is sampled according to Sam then D will always output 1. In case x' is random, distinguisher D outputs 1 only with probability $\frac{1}{2}$ and thus adversary $(\mathsf{Sam}, \mathsf{D})$ has an advantage of $\frac{1}{2}$ in the the AIPO security game.

Alternative definitions. There are two immediate alternative definitions of MB-AIPO that come to mind. One could modify the MB-AIPO definition by requiring that the obfuscation of point function $p_{x,m}$ is indistinguishable from an obfuscation of $p_{x',m'}$. That is, instead of only adapting the point value, we could conceive a notion in which the honest obfuscation should be indistinguishable from one where both the point address and the point value are chosen uniformly at random. A second alternative would be to require that the obfuscation of point function $p_{x,m}$ is indistinguishable from an obfuscation of $p_{x',m}$. This gives us three alternatives:

$$(x, m) \quad \text{vs.} \quad (x, m') \qquad \text{(Definition 6.9)} \tag{6.1}$$

$$(x, m) \quad \text{vs.} \quad (x', m') \tag{6.2}$$

$$(x, m) \quad \text{vs.} \quad (x', m) \tag{6.3}$$

where the first alternative corresponds to our Definition 6.9. We note that the second and third alternative are easily seen to imply also AIPO. However, the third alternative allows the obfuscation to leak large portions of m which goes against the intuition of what MB-AIPO wants to model.

Remark. As we will see later, assuming indistinguishability obfuscation exists, neither of the three definitions can be met for samplers that sample *computationally* hard-to-invert auxiliary inputs. Throughout this thesis we thus always refer to Definition 6.9 when speaking of an MB-AIPO, unless explicitly stated otherwise.

[8] Canetti and Dakdouk [CD08] show that composable point functions (without auxiliary input) imply multi-bit point functions (without auxiliary input). Their result carries over to obfuscation in the presence of auxiliary inputs.

Composable MB-AIPO. Analogously to plain AIPOs we can define a composable variant of MB-AIPOs. As we show that already non-composable MB-AIPOs cannot exist assuming the existence of indistinguishability obfuscators we do not provide a formal treatment of composable MB-AIPOs.

6.3.5 Average-Case Point Obfuscation

The notions for point obfuscation as defined above are over arbitrary distributions of super-logarithmic min-entropy over the point address, that is, we require samplers to sample points (x, aux) (resp. (x, m, aux)) such that no unbounded predictor can predict x (when it is not given the auxiliary information) or in other words, we require that

$$\mathrm{H}_\infty\left(x \;\middle|\; x \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)\right) \in \omega(\log(\lambda)).$$

Note that this is a necessary condition for both computational and statistical AIPO (and MB-AIPO).

A weaker form of (point) obfuscation is to require that the point address is sampled according to the uniform distribution (or a high min-entropy distribution) and thus less under the control of sampler Sam. Such a variant is usually referred to as *average-case* as the obfuscator only needs to be good on average rather than to be good for any choice of x (which is usually referred to as worst-case obfuscation). Note that the same terminology is also used for general-purpose obfuscators such as VBB and VGB obfuscators. Here a *worst-case* obfuscator needs to be *good* for any choice of circuit while an *average-case* obfuscator only needs to be good over a random choice of circuit. (Also see Section 5.3.1.)

Remark. Although all our definitions are defined in the *worst-case* scenario resulting in stronger requirements for corresponding obfuscators we note that our impossibility result (Chapter 7) for MB-AIPO$[\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}]$ rules out also the average-case variant where point address x is sampled uniformly at random.

6.4 V(B|G)B Implies AIPO Implies One-Way Functions

In the following we present three auxiliary results. First we show that VBB point obfuscation implies the indistinguishability-based AIPO notion of point obfuscation. If we consider VBB with auxiliary inputs this yields the strongest form of AIPO$[\mathcal{S}^{\mathrm{cup}}_{\mathsf{po}}]$ obfuscator, secure in the presence of *computationally* hard-to-invert auxiliary information. Furthermore, if we start from a composable VBB obfuscator we obtain a composable AIPO obfuscator. If we reduce the assumption on general-purpose obfuscation and only consider VGB obfuscation, then we obtain a weaker form of AIPO, namely, AIPO$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$ secure against *statistically* hard-to-invert auxiliary information. Both implications have also been shown by Matsuda and Hanaoka [MH14a] but we note that our direct proofs are simpler and shorter.

Finally, we show that both notions of AIPO (and even the indistinguishability-based notion without auxiliary information AIPO$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\not{\mathsf{aux}}}_{\mathsf{po}}]$) imply the existence of one-way functions.

6.4.1 V(G|B)B-(AI) Obfuscation Implies (AI)PO

In the following we show that both VBB and VGB obfuscators imply AIPO obfuscators for different sampler classes. We begin with the VBB result.

Lemma 6.1. *Let* O *be a secure VBB-AI point obfuscator. Then* O *is also AIPO*$[\mathcal{S}^{\mathrm{cup}}_{\mathsf{po}}]$*- as well as MB-AIPO*$[\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}]$*-secure.*

We show the result for plain AIPO security. The result follows analogously for the multi-bit output variant.

Proof. Let $(\mathsf{Sam}, \mathsf{D})$ be an adversary against the AIPO security of obfuscator O. We can rewrite the advantage of $(\mathsf{Sam}, \mathsf{D})$ in game PO (see Definition 6.6) as

$$\left|\Pr\left[\mathsf{D}(1^\lambda, \overline{p}_x, \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, \overline{p}_u, \mathsf{aux}) = 1\right]\right|, \tag{6.4}$$

where on the left sampler Sam samples (x, aux) and $\overline{p}_x$ is an obfuscation of x and on the right $\overline{p}_u$ is an obfuscation of a uniformly random point u of the same length as x. By the VBB security we know that for any polynomial p there exists a simulator Sim such that for sufficiently large λ we have that

$$\left|\Pr\left[\mathsf{D}(1^\lambda, \overline{p}, \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^{\overline{p}}(1^\lambda, \mathsf{aux}) = 1\right] \leq \frac{1}{\mathsf{p}(\lambda)}\right|.$$

Note that here we use the weaker definition of VBB obfuscation where the simulator may depend on the distinguishing probability (cf. Definition 6.4 and following remark). This allows us to rewrite the difference in Equation (6.4) as

$$\Bigg|\Pr\left[\mathsf{D}(1^\lambda, \overline{p}_x, \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^{\overline{p}_x}(1^\lambda, \mathsf{aux}) = 1\right] + \tag{6.5}$$

$$\Pr\left[\mathsf{Sim}^{\overline{p}_u}(1^\lambda, \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, \overline{p}_u, \mathsf{aux}) = 1\right] + \tag{6.6}$$

$$\Pr\left[\mathsf{Sim}^{\overline{p}_x}(1^\lambda, \mathsf{aux}) = 1\right] - \Pr\left[\mathsf{Sim}^{\overline{p}_u}(1^\lambda, \mathsf{aux}) = 1\right]\Bigg| \tag{6.7}$$

The difference in the first two lines (Equations (6.5) and (6.6)) can be upper bounded by $\frac{1}{\mathsf{p}}$ by the security of the VBB obfuscator. The final difference in Equation (6.6) can be upper bound by the unpredictability of sampler Sam. For this note that u is chosen as a uniformly random value and independent of aux and thus simulator Sim queries its oracle on u only with negligible probability. Similarly, we can upper bound the probability of simulator Sim querying its oracle on x by observing that doing so yields a predictor against sampler Sam which by assumption only has negligible advantage. Using the triangle inequality, we thus can upper bound the above by

$$\frac{2}{\mathsf{p}(\lambda)} + \mathsf{negl}(\lambda).$$

Thus, we have shown that for any polynomial poly there exists a simulator such that for sufficiently

large λ, Equation (6.4) is upper bounded by

$$\frac{2}{\mathsf{poly}(\lambda)} + \mathsf{negl}(\lambda)\,,$$

which yields the desired result, that is, $\mathsf{Adv}^{\mathrm{po}}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda)$ is negligible. □

It is easy to see that the above similarly holds for composable obfuscators, that is, any composable VBB obfuscator is also a composable AIPO[$\mathcal{S}^{\mathrm{cup}}$] obfuscator.

As for virtual grey-box obfuscation we note that the only difference in the above proof is the bound for Equation 6.7 as a VGB simulator runs in unbounded time. Thus, if aux hides point x only computationally then a VGB simulator may query its oracle on x as it can recover it from aux. In order to avoid this we need to require that the auxiliary information *statistically* hides point x which yields the following lemma.

Lemma 6.2. *Let* O *be a secure VGB point obfuscator. Then* O *is also AIPO*[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}$]*- as well as MB-AIPO*[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{mbpo}}$]*- secure.*

Also note that VGB implies VGB-AI and, hence, we do not need to state this as an extra assumption.

6.4.2 AIPO Implies One-Way Functions

In this section we show that, like most cryptographic primitives, AIPO implies one-way functions.

Lemma 6.3 (AIPO implies one-way functions)**.** *Point-function obfuscation (that is secure under auxiliary inputs) implies one-way functions.*

Proof. Two distributions that are computationally close and statistically far imply a distributional one-way function [Gol90], and a distributional one-way function implies a standard one-way function [IL89].[9] The security property of AIPO[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\not{\mathsf{aux}}}_{\mathsf{po}}$] implies that the obfuscation of p_x for a random x, where $x[1] = 0$ (i.e., the first bit of x is 0) is indistinguishable from the obfuscation of p_u for a random u. Hence, we have two computationally indistinguishable distributions. Let us argue that they are statistically far. With probability $\frac{1}{2}$, the first bit of u does not equal 0 and hence, the obfuscation of u is outside of the support of the distributions over p_x due to the functionality preservation requirement of obfuscators. Hence, the two random variables have statistical distance at least $\frac{1}{2}$ which concludes the proof. □

6.5 Constructions of Point-Function Obfuscation Schemes

In this final section of the chapter we review constructions of point obfuscators as well as corresponding assumptions. The study of point-function obfuscation started with Canetti [Can97] who proposed a point obfuscation scheme as the underpinnings of an *oracle hashing scheme* which, similarly to a random oracle, should hide all partial information on an input but unlike a random oracle be an

[9] The definition of a distributional one-way function asks that it is hard to find a preimage of a random image, that is, the sampling is over the image.

efficiently computable function. An obfuscator is parameterized by a cyclic group $\mathbb{G}$ of prime order. To obfuscate a point x we choose a generator $g \leftarrow\!\!\$\, \mathbb{G}$ at random and output (g, g^x). Given the tuple (g, g^x) one can then easily check if a target value τ is equal to x be computing g^τ and comparing the result to g^x.

Construction 6.1 (Point Obfuscator due to [Can97])**.** *Let $\mathcal{G} := \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a group ensemble, where each $\mathbb{G}_\lambda$ is a group of prime order $\mathsf{p}(\lambda) \in (2^{\lambda-1}, 2^\lambda)$. We define an obfuscator AIPO for points in the domain $\mathbb{Z}_{\mathsf{p}(\lambda)}$ as follows: $p(x) \overset{\mathsf{AIPO}}{\mapsto} C(r, r^x)$, where $r \leftarrow\!\!\$\, \mathbb{G}_\lambda$ is a random generator of $\mathbb{G}_\lambda$, and $C(r, r^x)$ is a circuit which on input τ, checks whether $r^x = r^\tau$.*

Canetti analyzes his construction down to various flavors of the decisional Diffie–Hellman assumption (DDH; see Appendix A.3 for a formal definition). Depending on the strength of the considered DDH variant different levels of security can be shown. If only assuming the "standard" DDH assumption then already the obfuscator achieves a form of virtual-black box obfuscation (without auxiliary information and where the points need to be sampled uniformly at random) where we, however, allow the simulator to depend on the distinguishing probability (see also the remark after Definition 6.4). If one considers a stronger variant of the DDH assumption where the points are not sampled uniformly at random but come from a distribution of only super-logarithmic min-entropy then the corresponding obfuscator can now also handle any inputs (and not just uniformly random inputs).

A second construction achieving the same sort of security, that is, VBB without auxiliary information and where the simulator may depend on the distinguishing probability is given by Wee [Wee05]. For his construction, Wee assumes the existence of a strong one-way permutation. Specifically, he requires that there exists a permutation such that no polynomial-size circuit can invert the permutation on more than polynomially many points. Wee also extends his construction to obtain a mild form of a multi-bit output point obfuscation for point messages which have logarithmic length.

High min-entropy obfuscators. Canetti, Micciancio, and Reingold (CMR) renamed Canetti's *oracle hashing schemes* [Can97] into *perfectly (probabilistic) one-way hash functions* (POW) [CMR98] and gave further constructions based on one-way permutations or collision-resistant hash functions. With their constructions CMR focused on high min-entropy distributions (i.e., distributions with min-entropy at least n^δ for $0 < \delta < 1$ and with n denoting the point length) and hence achieve weaker security properties when viewed as point-function obfuscators. Fischlin [Fis99] as well as Dodis and Smith [DS05] further optimize the constructions given by CMR. Dodis and Smith, furthermore, show how to construct proximity point obfuscators (point obfuscators which output 1 on inputs "close" to x) [DS05]. Their construction, however, still crucially depends on the underlying distributions having high min-entropy. Hofheinz, Malone-Lee, and Stam [HMLS07] study average-case and approximate obfuscation and also provide a construction for point-function obfuscation which, however, requires points to be chosen from the uniform distribution.

Random oracle obfuscation. While the constructions of Canetti and Wee [Can97, Wee05] achieve a weaker form of VBB obfuscation where the runtime of the simulator may depend on the distinguishing probability, Lynn, Prabhakaran, and Sahai [LPS04] show how to obfuscate point functions and multi-bit output point functions (and even more complicated access mechanisms) without any limitations. Their construction is, however, given in the programmable random oracle

model (cf. Section 3.3.2) and our negative result for MB-AIPOs presented in Chapter 7 can, thus, also be seen as an uninstantiability result for the random oracle in their scheme as their obfuscations of point functions can be seen to be also secure in the presence of auxiliary information.

6.5.1 Candidate AIPOs for Computationally Hard-To-Invert Auxiliary Info

While the above obfuscation schemes are defined in settings where adversaries are not given auxiliary information about the obfuscated points we next consider the stronger auxiliary inputs setting. One of the settings considered by Canetti for his original point obfuscation scheme [Can97] was the auxiliary inputs setting and he showed that under a strong auxiliary-inputs variant of the DDH assumption his obfuscator satisfies the requirements of an AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] obfuscator, that is, an AIPO for computationally hard-to-invert auxiliary inputs (see Definition 6.6). We here present the underlying assumption in the style of [BP12] adapted to our notation.

Assumption 6.1 ([Can97],[BP12])**.** *There exists an ensemble of prime order groups* $\mathcal{G} := \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ *of order* p *such that for any unpredictable auxiliary-input point sampler* $\mathsf{Sam} \in \mathcal{S}^{\text{cup}}_{\mathsf{po}}$ *it holds that for all* PPT *algorithms* $\mathcal{A}$ *there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for large enough* λ

$$\Big|\Pr\big[\mathcal{A}(r, r^x, \mathsf{aux}) = 1 \,:\, r \leftarrow\!\!\$\, \mathbb{G}_\lambda, (x, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)\big] - \Pr\big[\mathcal{A}(r, r^u, \mathsf{aux}) = 1 \,:\, r \leftarrow\!\!\$\, \mathbb{G}_\lambda, (x, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda), u \leftarrow\!\!\$\, \mathbb{Z}_{p(\lambda)}\big]\Big| \leq \mathsf{negl}(\lambda)\,.$$

Remark. Note that Assumption 6.1 basically assumes the security of Construction 6.1, that is, if we rewrite the above assumption in the game based notion of Definition 6.6 for Construction 6.1 then assumption and security game are identical.

A second candidate construction for AIPO is due to Bitansky and Paneth [BP12] who adapt the point obfuscation construction by Wee [Wee05] to allow for auxiliary input. Their construction is based on an assumption on the existence of strong pseudorandom permutations that remain one-way also in the presence of auxiliary information.

In a very recent work Bellare and Stepanovs [BS15] present generic constructions for AIPO starting from different primitives. They give a construction from strong forms of deterministic public-key encryption secure in the presence of auxiliary inputs as well as from auxiliary-input one-way functions (AI-OWF) and indistinguishability obfuscation.[10] One-way functions can be regarded as the weakest object in complexity-based cryptography and AI-OWF are thus one of the weakest primitives secure in the presence of computationally hard-to-invert auxiliary information. While the assumptions of Canetti and Bitansky and Paneth [Can97, BP12] are in some sense "fine-tuned" towards the intended result the assumption that a one-way function exists which is secure also in the presence of computationally hard-to-invert auxiliary information seems more tangible, both in terms of being able to construct such a primitive as well as to refute the existence of this primitive. On the downside we note that the notion considered by Bellare and Stepanovs ([BS15]; BS) is significantly

[10] They also present a third construction from Universal Computational Extractors, a framework that we discuss in detail in Part III.

stronger than what one would intuitively consider to be a AI-OWF. While the security definition of one-way functions considers preimages chosen uniformly at random, BS define the notion of AI-OWF[Sam] for an unpredictable auxiliary-input point sampler (cf. Definition 6.5) that jointly samples preimage x and auxiliary information aux and, hence, x is no longer sampled according to the uniform distribution but according to any high min-entropy distribution. The notion AI-OWF then considered by BS is a function that is secure with respect to any sampler $\mathsf{Sam} \in \mathcal{S}_{\mathsf{po}}^{\mathrm{cup}}$, that is, any sampler that samples computationally hard-to-invert auxiliary information.

Bellare and Stepanovs [BS15] construct an AIPO from indistinguishability obfuscation and an injective AI-OWF as follows:

Construction 6.2. *Let* F *be a keyed injective AI-OWF and* iO *an indistinguishability obfuscator.*

$\mathsf{AIPO}(x)$	$C[\mathsf{k}, y](x)$
$\mathsf{k} \leftarrow\!\!\$\ \mathsf{F.KGen}(1^\lambda)$	**if** $\mathsf{F.Eval}(\mathsf{k}, x) = y$ **then**
$y \leftarrow \mathsf{F.Eval}(\mathsf{k}, x)$	**return** 1
$\overline{p} \leftarrow\!\!\$\ \mathsf{iO}(C[\mathsf{k}, y])$	**return** 0
return $\overline{p}$	

Let us provide some intuition behind the security proof. In Section 5.6.1 we have seen a result due to Boyle, Chung and Pass (BCP) that puts indistinguishability obfuscation and differing-inputs obfuscation in relation [BCP14]. BCP show that any general-purpose indistinguishability obfuscator is a restricted differing-inputs obfuscator for circuits that differ only on polynomially many values (Theorem 5.13 on page 92).

Circuit $C[\mathsf{k}, y]$ in Construction 6.2 is zero everywhere except on the single input x for which $y = \mathsf{F.Eval}(\mathsf{k}, x)$. For this note that we required F to be injective. Thus, in order to apply BCP we must show that $C[\mathsf{k}, y]$ and the all-zero circuit Z are differing-inputs in the presence of auxiliary information aux. If so, we can then argue that the construction is secure since the constant zero circuit contains no information about point x. To complete the argument we need to argue that given $C[\mathsf{k}, y]$ one cannot extract the unique preimage x. But this is exactly the security of the AI-OWF which concludes the proof. We refer to [BS15] for further details.

6.5.2 Candidate AIPOs for Statistically Hard-To-Invert Auxiliary Information

While the constructions in the previous section achieve AIPO security for computationally hard-to-invert auxiliary information they obtain their auxiliary-input security by "pushing the auxiliary inputs into the assumption". For this note that, for example, Assumption 6.1 but also the definition of AI-OWF used by Bellare et al. [BS15] require security of the underlying primitive for *all* unpredictable auxiliary-input point samplers $\mathsf{Sam} \in \mathcal{S}_{\mathsf{po}}^{\mathrm{cup}}$. Dodis, Tauman-Kalai and Lovett (DKL; [DKL09]) present a construction based on a variant of the Learning Parity-with-Noise (LPN) assumption (a falsifiable assumption in the terminology of Naor [Nao03]) which, unlike the above assumptions, does not quantify over all auxiliary information samplers. They present a construction of point-function obfuscators as well as a construction for multi-bit output point obfuscators. On the downside the seemingly weaker assumption employed by DKL also achieves only a weaker form of AIPO security, namely, AIPO[$\mathcal{S}_{\mathsf{po}}^{\mathrm{sup}}$] (AIPOs secure in the presence of *statistically* hard-to-invert auxiliary

information). For their MB-AIPO construction the auxiliary information must furthermore hide point x even in presence of message m, that is, auxiliary information of, for example, $x \oplus m$ could not be tolerated.

A second construction is again the construction of Canetti (Construction 6.1). As discussed in Section 6.2 Bitansky and Canetti show that the construction is a t-composable VGB point obfuscator under the (non-standard) t-Strong Vector Decision Diffie–Hellman assumption which they show to hold in the generic group model [BC14]. The connection to AIPOs for statistically hard-to-invert auxiliary information was recently made by Matsuda and Hanaoka [MH14a] who show that composable VGB-AI point obfuscators imply the existence of composable AIPO with respect to statistically unpredictable distributions; we give a direct proof of this statement as Lemma 6.2 (see also Figure 6.1). Hence under the t-Strong Vector Decision Diffie–Hellman assumption Construction 6.1 is a composable AIPO$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$. For completeness we here present the assumption:

Assumption 6.2 (t-Strong Vector Decision Diffie–Hellman assumption [BC14]:)**.** *There exists an ensemble of prime order groups $\mathcal{G} := \{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ of order p_λ such that for any unpredictable auxiliary-input point sampler $\mathsf{Sam} \in \mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\cancel{\mathsf{aux}}}_{\mathsf{po}}$, such that Sam on input the security parameter 1^λ outputs $(\mathbf{x}, \mathsf{aux})$ where $\mathbf{x}$ is a vector of points and $\mathsf{aux} = \varepsilon$ is the empty string, it holds that the following distributions are computationally indistinguishable*

$$\left(\begin{array}{l}(\mathbf{x},\varepsilon) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)\\ \textbf{for } i = 1,\ldots,|\mathbf{x}| \textbf{ do}\\ \quad\\ \quad r \leftarrow\!\!\$\, \mathbb{G}_\lambda\\ \quad \mathbf{p}[i] \leftarrow\!\!\$\, (r, r^{\mathbf{x}[i]})\\ \textbf{return } \mathbf{p}\end{array}\right) \approx_c \left(\begin{array}{l}(\mathbf{x},\varepsilon) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)\\ \textbf{for } i = 1,\ldots,|\mathbf{x}| \textbf{ do}\\ \quad \mathbf{x}[i] \leftarrow\!\!\$\, \{0,1\}^{|\mathbf{x}[i]|}\\ \quad r \leftarrow\!\!\$\, \mathbb{G}_\lambda\\ \quad \mathbf{p}[i] \leftarrow\!\!\$\, (r, r^{\mathbf{x}[i]})\\ \textbf{return } \mathbf{p}\end{array}\right)$$

We note that a crucial difference between the previous assumption and Assumption 6.1 is that here we do not quantify over auxiliary inputs. However, due to the VGB property that auxiliary input security is "for free" (see Section 5.4.1) we still get an auxiliary-inputs secure point obfuscator.

PART II

RANDOM-ORACLE UNINSTANTIABILITY FROM INDISTINGUISHABILITY OBFUSCATION

Part Summary

In the following chapters we show that indistinguishability obfuscation can be used to show that various random oracle based schemes are uninstantiable. We begin our study by looking at strong forms of point-function obfuscation which have simple and efficient constructions in the random oracle model [LPS04]. Assuming the existence of indistinguishability obfuscation we show that we cannot hope to build such obfuscators in the standard model. Note that this result is quite different from the seminal result of Canetti, Goldreich, and Halevi [CGH98] who show that a specifically crafted scheme secure in the random oracle model cannot be be instantiated. In contrast, we show that an interesting security notion cannot be met in the standard model. We then turn to random oracle transformations which turn schemes with weak security (in the standard model) into schemes with strong security (in the random oracle model). Examples are the Encrypt-with-Hash transformation which allows to transform randomized IND-CPA public-key encryption schemes into IND-secure deterministic public-key encryption schemes or the Fujisaki–Okamoto transform to obtain CCA-secure encryption. We show that a large class of transformations (including the two aforementioned ones) are uninstantiable if indistinguishability obfuscators exist. To this end we show the existence of secure standard-model schemes which, when transformed, become uninstantiable.

Chapter Summary

In Chapter 7 we show that indistinguishability obfuscation and strong forms of multi-bit output point obfuscation secure in the presence of auxiliary information is mutually exclusive. In particular we show that if indistinguishability obfuscation exists then MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] cannot exist which in turn means that also composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] and composable VBB-AI obfuscation for point functions cannot exist. The results in this chapter are mostly based on [BM14a].

In Chapter 8 we extend the techniques of Canetti, Goldreich, and Halevi [CGH98] and show that many prominent random-oracle transformations are uninstantiable if indistinguishability obfuscation exists. In particular, we show that this holds for the popular Encrypt-with-Hash and Fujisaki–Okamoto transform to construct deterministic public-key encryption and CCA-secure hybrid encryption respectively. The results in this chapter are based on [BFM15].

Chapter 7

General-Purpose Obfuscation versus Point Obfuscation with Auxiliary Input

"[...] and either must die at the hand of the other for neither can live while the other survives [...]"

Sybill Trelawney, Harry Potter and the Order of the Phoenix

Summary. We begin our discussion of random oracle uninstantiability by exploring the relation between general-purpose obfuscation and point-function obfuscation secure in the presence of computationally hard-to-invert auxiliary information. We show that indistinguishability obfuscation and MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] (and thus composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]) are mutually exclusive and explain how this can be interpreted as a random oracle uninstantiability result. This result was published at Asiacrypt 2014 [BM14a]. For completeness, we present a recent extension due to Bellare, Stepanovs, and Tessaro [BST15] who show that virtual grey-box obfuscation is mutually exclusive with *verifiable* (and not necessarily composable) AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$].

Chapter content

7.1 Introduction

Random oracle uninstantiability results can come in various forms and in this chapter we examine the role the random oracle plays in a construction of a point function obfuscation scheme by Lynn, Prabhakaran, and Sahai [LPS04] who were amongst the first to give positive results for obfuscation for multi-bit output point functions as well as for more complex access control functionalities, in the random oracle model. Lynn et al. construct a point obfuscation for a MBPF $p_{x,m}$ relative to a random oracle RO as

$\underline{\mathsf{MBPO}^{\mathsf{RO}}(x, m)}$
$\overline{x} \leftarrow \mathsf{RO}(0\|x)$
$\overline{m} \leftarrow \mathsf{RO}(1\|x) \oplus m$
return $(\overline{x}, \overline{m})$

where the result $(\overline{x}, \overline{m})$ can be thought of as a point obfuscation $\overline{p}_{x,m} := (\overline{x}, \overline{m})$ which is evaluated as

$\underline{\overline{p}^{\mathsf{RO}}_{x,m}(x')}$
if $\overline{x} = \mathsf{RO}(0\|x')$ **then**
  return $\overline{m} \oplus \mathsf{RO}(1\|x')$
return $\perp$

In order to yield a correct obfuscation we need to consider a fixed input-length random oracle that is injective as otherwise there could be multiple and distinct x' that pass the if-branch in the first line and potentially yield different results. Lynn et al. show that their construction yields a virtual black-box obfuscator for multi-bit output point functions without auxiliary-input security. Auxiliary-input security in the context of obfuscation was introduced by Goldwasser and Tauman-Kalai [GK05], two years after Lynn et al. published their construction and it is easily seen that the above construction remains VBB also in this stronger setting in case we do not allow the auxiliary information to depend on the random oracle. For completeness we here present a proof of this statement and subsequently discuss what needs to be changed in order to obtain a point obfuscation scheme that is VBB secure even in the presence of random-oracle dependent auxiliary information.

Theorem 7.1 ([LPS04] Theorem 2 (extended))**.** *The above construction due to Lynn et al. [LPS04] yields an auxiliary-input virtual black-box (VBB-AI) multi-bit output point-function obfuscator in the random oracle model if the auxiliary input is independent from the random oracle.*

For the following we simplify and assume that the above construction obfuscates points in $\{0,1\}^{\lambda-1}$ with point messages in $\{0,1\}^{\lambda}$. Consequently, we consider a fixed input-length and injective random oracle $\mathsf{RO} : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$.

Proof. For any adversary $\mathcal{A}$, we need to show the existence of a simulator Sim that on input the security parameter 1^{λ}, auxiliary information aux and with oracle access to a point obfuscation $\overline{p}_{x,m}$ can simulate the behavior of adversary $\mathcal{A}$ on input $(1^{\lambda}, \overline{p}_{x,m}, \mathsf{aux})$. As we are in the random oracle setting, adversary $\mathcal{A}$ expects access to a random oracle RO. We construct simulator Sim as follows: it chooses two random strings $\overline{x}, \overline{m} \in \{0,1\}^{\lambda}$ and runs adversary $\mathcal{A}$ on input $(1^{\lambda}, (\overline{x}, \overline{m}), \mathsf{aux})$. Without loss of generality we assume that $\mathcal{A}$ never repeats a query to its oracle. To answer an oracle query q from adversary $\mathcal{A}$ to random oracle RO, simulator Sim strips off the first bit of q and then queries its own oracle $\overline{p}$ on $q[2..|q|]$. Simulator Sim distinguishes two cases: If the result is $\perp$, then simulator Sim chooses a uniformly random value in $\{0,1\}^{\lambda}$ which it returns. Otherwise, if the result is m, then it considers the first bit $q[1]$ of query q. If $q[1] = 0$, then simulator Sim returns $\overline{x}$. If, on the other hand, $q[1] = 1$, then it returns $\overline{m} \oplus m$. When adversary $\mathcal{A}$ stops, simulator Sim stops and outputs whatever $\mathcal{A}$ outputs.

The simulation of the random oracle for adversary $\mathcal{A}$ is perfect: the distribution simulated by Sim is the uniform distribution and the randomly chosen obfuscation $(\overline{x}, \overline{m})$ together with the simulation

of the random oracle matches the obfuscated point function that simulator Sim is given as an oracle. For this note that if adversary $\mathcal{A}$ recovers x and queries the random oracle on $0\|x$—the simulator notices this when its own oracle returns m instead of $\perp$ on this query—it expects $\overline{x}$. Similarly, if $\mathcal{A}$ queries $1\|x$ the expected result is $\overline{m} \oplus m$. □

In case the auxiliary information aux may depend on the random oracle, the above simulation no longer works. Consider, for example, the case that aux contains the first half of $\mathsf{RO}(0\|x)$. Then adversary $\mathcal{A}$ immediately knows that when it is presented with the "fake" point obfuscation $(\overline{x}, \overline{m})$ that the obfuscation is "incorrect" since with overwhelming probability the first half of $\overline{x}$ does not match the information in aux. The problem is that the LPS scheme is deterministic and, thus, aux can contain information about how the "correct" obfuscation will look like. To remedy this, we need to make the obfuscation scheme probabilistic as aux may not depend on the randomness of the obfuscator. A straight forward adaption of the LPS scheme is the following which we call the *randomized LPS scheme*:

$\underline{\mathsf{MBPO}^{\mathsf{RO}}(x, m; r)}$

$\overline{x} \leftarrow \mathsf{RO}(0\|r\|x)$

$\overline{m} \leftarrow \mathsf{RO}(1\|r\|x) \oplus m$

return $(\overline{x}, \overline{m}, r)$

Theorem 7.2. *The above construction randomized LPS construction yields an auxiliary-input virtual black-box (VBB-AI) multi-bit output point-function obfuscator in the random oracle model even in case the auxiliary input depends on the random oracle.*

The proof is similar to before and we discuss the high-level idea next.

Proof sketch. The simulator works analogous to before with the exception that it also generates randomness r which is passed on together with $\overline{x}, \overline{m}$ to adversary $\mathcal{A}$. Note that since the obfuscation is now randomized (i.e., $\overline{x} = \mathsf{RO}(0\|r\|m)$) and auxiliary information aux is independent of the randomness of the obfuscator we have that

$$\mathrm{H}_\infty(\mathsf{RO}(0\|r\|m) \mid r) \approx \mathrm{H}_\infty(\mathsf{RO}(0\|r\|m) \mid \mathsf{aux}, r).$$

In other words aux does not significantly reduce the uncertainty about a "correct" value $\overline{x}$ (and the same holds for $\overline{m}$). Thus, with overwhelming probability adversary $\mathcal{A}$ cannot distinguish between being run on input aux and a "matching" point obfuscation or being run on aux and a random point obfuscation. Thus, the first step of the simulation where simulator Sim starts evaluating $\mathcal{A}$ on input $((\overline{x}, \overline{m}, r), \mathsf{aux})$ where $(\overline{x}, \overline{m}, r)$ are three random values is a perfect simulation of the environment that $\mathcal{A}$ expects to be in.

Our previous simulator answered all random oracle queries of $\mathcal{A}$ with a uniformly random value, unless adversary $\mathcal{A}$ queried on value $b\|x$ (now $b\|r\|x$) where x denotes the point address of the point function that Sim has access to. In order to keep up a perfect simulation of the random oracle for $\mathcal{A}$ even in case $\mathcal{A}$ has certain information about RO (embedded in aux) we, thus, let Sim forward any oracle query that is different from $b\|r\|x$ to its own RO. (For this note, that also Sim has access to RO.) In case $\mathcal{A}$ never queries its oracle on value $b\|r\|x$ the simulation is again perfect. In case $\mathcal{A}$ makes a query $b\|r\|x$ simulator Sim learns both x and m (by querying its point function oracle on x).

It can then generate a *correct* obfuscation for point (x,m), rewind adversary $\mathcal{A}$ and rerun $\mathcal{A}$ relative to its own random oracle on input the correct obfuscation and auxiliary information aux. □

A simple corollary of the above result is that, in the random oracle model, the construction of Lynn et al. also achieves MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] security, that is, MB-AIPO with respect to computationally hard-to-invert auxiliary information (that may depend on the random oracle). Our first result shows that, if indistinguishability obfuscation exists, then the random oracle in the above construction cannot be instantiated by any standard model hash function. We do, however, go a step further and not only rule out MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] security via instantiating the random oracle in the Lynn et al.–construction (or randomized LPS construction) but we show that in the presence of iO no standard model construction that achieves MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] security can exist. A corollary of this rules out also the existence of *composable* (plain) AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]: as we have seen in the previous chapter, composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] allows the construction of MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]. Similarly, as virtual black-box obfuscation in the presence of auxiliary information implies MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]-security this also yields an impossibility result for VBB-AI obfuscation of multi-bit output point functions. In summary, we derive the following negative results.

Theorem [informal]. *If indistinguishability obfuscation exists, then MB-AIPO*[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] *and hence composable AIPO*[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] *as well as VBB-AI obfuscation for multi-bit output point functions do not exist.*

Let us sketch a proof which is inspired by the impossibility result of Barak et al. [BGI+12] who show that VBB obfuscation cannot exist in general. As shown in Section 5.3.2, the first step taken by Barak et al. is to show that two particularly chosen circuits (resp. Turing machines) cannot be obfuscated together. Interestingly for us is that one of the circuits is a multi-bit output point function $p_{x,m}$ with a randomly chosen point address x and point value m. The second function, on the other hand, is a *test function* $T_{x,m}$ that takes as input a circuit C and tests whether $C(x)$ is equal to m. Barak et al. show that if an adversary is given only oracle access to $p_{x,m}$ and $T_{x',m'}$ then it cannot check whether the two functions "match", i.e., whether $(x', m') = (x, m)$. In turn, when given a circuit C that computes $p_{x,m}$, the adversary can run $T_{x,m}$ on C and simply check whether $T_{x,m}(C)$ returns 1. Hence, the obfuscation of $p_{x,m}$ and the obfuscation of $T_{x,m}$ leak more information than two oracles for $p_{x,m}$ and $T_{x,m}$, thus establishing a counterexample for VBB obfuscation. We note that although the starting point of Barak et al.'s result is a point function $p_{x,m}$, they actually construct an unobfuscatable function that is a combination of the point function $p_{x,m}$ together with test function $T_{x,m}$ and thus their result is an impossibility result for general VBB obfuscation rather than an impossibility result for point-function obfuscation.

In order to strengthen our impossibility result we attack a notion of obfuscation as weak as possible and settle for MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] instead of VBB-AI point obfuscation. In order to attack MB-AIPO, we need to define an adversarial *auxiliary-input multi-bit output point sampler* Sam (cf. Definition 6.9) that produces computationally unpredictable auxiliary information (i.e., $\mathsf{Sam} \in \mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$). On the other hand, sampler Sam must choose auxiliary information aux such that it allows distinguisher D to distinguish between obfuscations of $p_{x,m}$ and obfuscations of $p_{x,m'}$. For this, our adversarial sampler Sam draws random values x and m to specify point function $p_{x,m}$ and, as auxiliary information aux, it will output a specially devised obfuscation that approximates the behavior of test function $T_{x,m}$.

Given a circuit aux that approximates $T_{x,m}$ and a multi-bit point function p, in the second stage, distinguisher D, outputs whatever circuit aux outputs when run on input p. It distinguishes

successfully between an obfuscation $\overline{p}$ of the "matching" multi-bit point function $p_{x,m}$ and the obfuscation $\overline{p}$ of a non-matching multi-bit point function $p_{x,m'}$ in case aux correctly computes test function $T_{x,m}$.

To conclude the argument it remains to show how $T_{x,m}$ can be approximated such that its description hides value x. For this, the idea is to use the zero-circuit technique for indistinguishability obfuscation (cf. Section 5.5.5). That is, we obfuscate the test function via an indistinguishability obfuscator and prove that the result is indistinguishable from an obfuscation of the constant zero circuit Z; the circuit that returns 0 on all inputs. As the zero circuit does not contain any information about x, indistinguishability obfuscation guarantees that likewise, an obfuscation of the test function $T_{x,m}$ hides x computationally. In more detail, let y be the output of a pseudorandom generator PRG when applied to m. Auxiliary information aux is set to be an indistinguishability obfuscation of the following circuit $C[x, y]$ with parameters x and y hard-coded. Circuit $C[x, y]$ gets as input a circuit $\overline{p}$, runs $\overline{p}$ on x and checks whether $\mathsf{PRG}(\overline{p}(x))$ is equal to y. If yes, it outputs 1. Else, it outputs 0.

For simplicity, let us assume that the PRG is injective. Then, $C[x, y]$ behaves exactly like the test function $T_{x,m}$. Interestingly, and that is the key idea, we do not actually use m to compute the circuit $C[x, y]$, we only need $y = \mathsf{PRG}(m)$. In particular, as PRG is a one-way function, y does not leak m. Moreover, as PRG is a pseudorandom generator, y does not even leak whether a preimage m exists.

We will now use the PRG property to argue that an indistinguishability obfuscation of $C[x, y]$ does not leak anything about x. (This technique of using a PRG within an obfuscated circuit was first introduced by Sahai and Waters [SW14].) Namely, if y is in the image of the PRG, then $C[x, y]$ is equal to the test function $T_{x,m}$. In turn, when y is not in the image of the PRG, then $C[x, y]$ is the all-zero function. Due to the PRG security, these two distributions—$C[x, y]$ when y is drawn as an output from the PRG and $C[x, y]$ when y is drawn at random—are computationally indistinguishable. Moreover, when the PRG has enough stretch, then with overwhelming probability, a random y is not in the image of the PRG, and hence, with overwhelming probability over a random y, the circuit $C[x, y]$ is the all-zero circuit Z. For the two functionally equivalent circuits $C[x, y]$ and Z, it holds that $\mathsf{iO}(C[x, y])$ is computationally indistinguishable from $\mathsf{iO}(\mathsf{Z})$. As Z leaks nothing about x, we can argue that also $\mathsf{iO}(C[x, y])$ leaks nothing about x and hence, x is unpredictable given the auxiliary information aux as produced by Sam. Consequently, $\mathsf{Sam} \in \mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$ as required by the definition of MB-AIPO$[\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}]$.

Remark. Note that unpredictability in the MB-AIPO definition only requires that x is unpredictable from auxiliary information aux. Therefore, hiding m might seem unnecessary. Interestingly, it turns out that this is not merely an artifact of our proof. In Chapter 12, we define a strong notion of unpredictability where x needs to be unpredictable from the pair (aux, m), and we show that MB-AIPO can be achieved under this definition, assuming plain (non-composable) AIPO in conjunction with indistinguishability obfuscation.

Finally, let us remark upon that our proof crucially relies on auxiliary information aux to hide point x only computationally as we embed an indistinguishability obfuscation which is only computationally secure unless the polynomial hierarchy collapses (see also Theorem 5.7, Section 5.5.2 on page 81).

Extensions to (plain) AIPO. Our negative results do not carry over to the setting of obfuscating plain point functions in the presence of computationally hard-to-invert auxiliary information, that is, to plain AIPO (assuming they are not composable[1]). This is due to the fact that we cannot apply the PRG to a function that only outputs a single bit and the use of a PRG seems crucial in the zero-circuit argument. Analogously, it looks unlikely that the impossibility result of Barak et al. [BGI+12] carries over to plain point functions, because it seems crucial that the point function $p_{x,m}$ has a multi-bit output m. Imagine that T_x takes the circuit C as input and returns 1 if and only if $C(x) = 1$. Then, an adversary can perform binary search and recover x, even when only given access to T_x and p_x as oracles.[2] Consequently, it seems that also their result does not carry over to standard point functions.

Very recently, Bellare, Stepanovs, and Tessaro ([BST15]; BST) show that one can extend our results to cover also a large class of non-composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] if one is willing to assume a stronger general-purpose obfuscator, namely, virtual grey-box obfuscation. VGB obfuscation, similarly, to VBB obfuscation is simulation-based, that is, it considers simulators that only get black-box access to the functionality. BST show that given a VGB obfuscator one can build an obfuscation of test function T_x which hides x but allows to test whether a point function $p_{x'}$ is a point function for x or not. Above we argued that such a test function is problematic as it allows an adversary to recover x, for example, by employing binary search. However, if the test function can *verify* that the circuit that it gets as input is a point circuit, then oracle access to T_x is not better than oracle access to a point function p_x. As a result, BST show that any point obfuscation scheme that is *verifiable* cannot be AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] secure where verifiability means that it can be efficiently (and deterministically) checked whether or not a circuit is in the support of the obfuscator (i.e, could be output by the obfuscator).

Outline. We begin presenting our negative result on MB-AIPOs in Section 7.2 following with a discussion on implications for point-function obfuscation and thoughts on how this result relates to the *random oracle controversy* (Section 7.3). We go on to present potential counter-measures and, finally, in Section 7.4 we present the BST result on AIPO vs. VGB obfuscation.

7.2 IO Implies the Impossibility of MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]

In the following we present our negative result, namely that indistinguishability obfuscation and multi-bit output point function obfuscation in the presence of computationally hard-to-invert auxiliary information (MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]) are mutually exclusive. We discuss implications of this result in greater detail in Section 7.3.

7.2.1 IO and MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] are Mutually Exclusive

Multi-bit point obfuscation with auxiliary inputs is a powerful primitive and has, for example, been used to construct CCA-secure encryption schemes [MH14a] and to circumvent black-box impossibility results for three-round weak zero-knowledge protocols for NP [BP12]. Our following result says that,

[1] Canetti and Dakdouk [CD08] show that composable AIPO already implies MB-AIPO.

[2] Access to test function T_x alone suffices to recover x. For this consider an adversary that recursively prepares circuits $C[\mathsf{prefix}]$ which return 1 on inputs that start with prefix and 0 otherwise. With $|x|$ queries the adversary can recover x bit by bit.

if indistinguishability obfuscation and pseudorandom generators exist, then MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] (as defined in Definition 6.9) cannot exist.

Theorem 7.3. *Let* $\mathsf{al} : \mathbb{N} \to \mathbb{N}$ *and* $\mathsf{ml} : \mathbb{N} \to \mathbb{N}$ *be two polynomials. If indistinguishability obfuscation exists for all circuits in* P/poly*, then MB-AIPO*[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] *with address length* al *and message length* ml *does not exist.*

Remark. It is worth mentioning that the result remains valid even if we consider average case MB-AIPOs (where point address x is chosen uniformly at random). Furthermore, it covers all the three variants of MB-AIPO that we discuss below Definition 6.9. We discuss a further restriction in Section 7.2.2 where we allow the obfuscation to output erroneous circuits with certain probability, that is, the obfuscator only needs to be correct most of the time. Finally, note that by restricting the obfuscator to a specific address and message length makes the impossibility result stronger.

To prove Theorem 7.3 we use indistinguishability obfuscation to construct a computationally unpredictable sampler $\mathsf{Sam} \in \mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$ together with a distinguisher D that, given auxiliary information aux from Sam can distinguish point obfuscations for point-message pairs as specified by sampler Sam from obfuscations for point messages coming from the uniform distribution.

Let us first present the intuition behind sampler Sam. Sampler Sam takes as input the security parameter 1^λ and outputs two values x, m together with some auxiliary information aux. Auxiliary information aux will be the indistinguishability obfuscation of a predicate circuit that takes as input a description of a circuit $\langle C \rangle$, evaluates the circuit on the hard-coded value x, runs the result through a pseudorandom generator PRG and finally compares this result with some hard-coded value y. That is, we consider the circuit $T[x, y]$ that has values x and y hard-coded and which is defined as:

$T[x, y](\langle C \rangle)$

$m' \leftarrow \mathsf{UC}(\langle C \rangle, x)$

if $\mathsf{PRG}(m') = y$ **then**

  return 1

return 0

Here, UC denotes the universal circuit taking as input a circuit description $\langle C \rangle$ (of an a priori fixed length) together with a value x (also of a fixed length) and which outputs $C(x)$. This use of a pseudorandom generator PRG allows us later to argue that if value y is chosen uniformly at random that it will be outside the image of PRG with high probability. For this note that at most a $\frac{2^{\mathsf{PRG.il}(\lambda)}}{2^{\mathsf{PRG.ol}(\lambda)}}$-fraction of the image of the PRG is used and, thus, if we choose the stretch to be sufficiently high, for example, we choose the output length to be twice the input length, then only a negligible fraction of the image is used. Finally, if y is such that it is outside the image then circuit $T[x, y]$ returns 0 on every input and is thus equivalent to the constant zero circuit Z. The theorem then follows with the security of the indistinguishability obfuscator. We next make this intuition formally.

Proof of Theorem 7.3. Let MBPO be a multi-bit output point obfuscator with address length $\mathsf{MBPO.al}$ and message length $\mathsf{MBPO.ml}$. Let further $\mathsf{MBPO.ol}$ denote the polynomial that describes the output length of MBPO that is, it describes the size of point obfuscations as generated with MBPO. Let $\mathsf{PRG} : \{0,1\}^{\mathsf{PRG.il}(\lambda)} \to \{0,1\}^{2\mathsf{PRG.il}(\lambda)}$ be a pseudorandom generator with stretch 2 where we

set $\mathsf{PRG.il} = \mathsf{MBPO.ml}$. Note that as point obfuscators imply the existence of one-way functions (Lemma 6.3 in Section 6.4) this is without loss of generality.

Fix a circuit encoding and let $\mathsf{UC}(\cdot,\cdot)$ be the universal circuit that on input a description $\langle C\rangle$ of a circuit C and value x outputs $C(x)$. We define sampler Sam as follows: it chooses random values m and x and computes $y \leftarrow \mathsf{PRG}(m)$. It then constructs circuit $T[x,y]$ with values x and y hard-coded. It creates an obfuscation $\mathsf{aux} \leftarrow\!\!\$\, \mathsf{iO}(T[x,y])$ and outputs (x, m, aux). Once more as pseudocode:

$\mathsf{Sam}(1^\lambda)$	$T[x,y](\langle C\rangle)$
$m \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PRG.il}(\lambda)}$	$m' \leftarrow \mathsf{UC}(\langle C\rangle, x)$
$x \leftarrow\!\!\$\, \{0,1\}^{\mathsf{MBPO.al}(\lambda)}$	**if** $\mathsf{PRG}(m') = y$ **then**
$y \leftarrow \mathsf{PRG}(m)$	**return** 1
$\mathsf{aux} \leftarrow\!\!\$\, \mathsf{iO}(T[x,y])$	**return** 0
return (x, m, aux)	

Note that we can choose UC such that it can process circuit descriptions of length $\mathsf{MBPO.ol}(\lambda)$ and values $x \in \{0,1\}^{\mathsf{MBPO.al}(\lambda)}$.

To complete the proof we need to show that there exists an efficient distinguisher D that breaks the security of the multi-bit point obfuscator when given auxiliary information aux as output by sampler Sam. Additionally, we need to argue that $\mathsf{Sam} \in \mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$, that is, that Sam is a computationally unpredictable multi-bit point sampler. We begin with a definition of distinguisher D.

Distinguisher D gets values $\overline{p}$ and aux as input, where $\overline{p}$ is either a point obfuscation of $p_{x,m}$ sampled according to Sam or an obfuscation for $p_{x,u}$ for a uniformly random value u. Distinguisher D computes $\mathsf{aux}(\overline{p})$ and outputs the result. If $\overline{p}$ is an obfuscation of $p_{x,m}$, then D computes the predicate function

$$\mathsf{PRG}(p_{x,m}(x)) = y$$

where y was computed as $\mathsf{PRG}(m)$. Thus, it will always output 1. If, however, $\overline{p}$ is an obfuscation of $p_{x,u}$ then with overwhelming probability over the choice of u adversary D returns 0. For this we note that PRG-security implies that for any fixed value y, the probability that $\mathsf{PRG}(u)$ is equal to that value for a random u, is negligible. In other words, for all $y \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ it holds that

$$\Pr_{u \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PRG.il}(\lambda)}}[\mathsf{PRG}(u) = y] \leq \mathsf{negl}(\lambda)\,.$$

It follows that $(\mathsf{Sam}, \mathsf{D})$ break the security of the obfuscator with overwhelming probability. It remains to show that $(\mathsf{Sam}, \mathsf{D})$ is a valid pair of adversaries, that is, that $\mathsf{Sam} \in \mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$. We outsource this to Lemma 7.4 which is a Lemma of independent interest that we will reuse (in slightly adapted form) for later results. □

With the following lemma we capture that the auxiliary information aux as computed by sampler Sam hides point address x down to the security of the indistinguishability obfuscator iO and the pseudorandom generator PRG. For this we show that no efficient adversary can distinguish obfuscations of circuit $T[x,y]$ from obfuscations of the constant zero circuit Z when appropriately padded, that is, if we pad Z to the same length as circuit $T[x,y]$. As the constant zero circuit does, information theoretically, not contain any information about point x, we know that also an obfuscation of $T[x,y]$ must computationally hide point x.

Lemma 7.4. *Let ℓ, t be polynomials. For $x \in \{0,1\}^{\ell(\lambda)}$, let $\mathsf{UC}(\cdot, x)$ be a universal circuit with value x hard-coded, that on input a description $\langle C \rangle \in \{0,1\}^{t(\lambda)}$ of a circuit C outputs $C(x)$. If $\mathsf{PRG} : \{0,1\}^{\mathsf{PRG.il}(\lambda)} \to \{0,1\}^{2\mathsf{PRG.il}(\lambda)}$ is a pseudorandom generator and iO is a secure indistinguishability obfuscator for all circuits in P/poly, then for all efficient PPT distinguishers D it holds that*

$$\left|\Pr\left[\mathsf{D}\left(1^\lambda, \mathsf{iO}\left(\mathsf{PRG}(\mathsf{UC}(\cdot,x)) = \mathsf{PRG}(m)\right)\right) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, \mathsf{iO}(\mathsf{Z})) = 1\right]\right| \leq \mathsf{negl}(\lambda) \tag{7.1}$$

is negligible. Here Z denotes the constant zero circuit padded to the same length as circuit $(\mathsf{PRG}(\mathsf{UC}(\cdot,x)) = \mathsf{PRG}(m))$. The first probability is over the random choice of x and m and the coins of iO and D and the second probability is over the coins of iO and D.

Remark. We note that the stretch of the PRG in the above Lemma can be reduced further, that is, it suffices if the difference $\mathsf{PRG.ol}(\lambda) - \mathsf{PRG.il}(\lambda)$ is super-logarithmic in the security parameter.

Proof. We bound the distinguishing probability in Lemma 7.4 with the security of the PRG and the indistinguishability obfuscator iO. We proof the statement via a sequence of three games where the first is identical to the left probability in Equation (7.1) and the last game describes an identical distribution as the right hand side of Equation (7.1). The statement then follows by showing that the differences between each two consecutive games in negligible. We depict the games in Figure 7.1 and next give a textual description. The arrows from game to game (both in text and picture) indicate the reduction target for that particular step.

$\mathsf{Game}_1(\lambda)$**:** The game is identical to the left hand side of Equation (7.1). That is, the game chooses random values m and x and computes $y \leftarrow \mathsf{PRG}(m)$. It constructs the predicate circuit $T[x,y] \leftarrow (\mathsf{PRG}(\mathsf{UC}(\cdot,x)) = y)$ and an obfuscation $\overline{C} \leftarrow\!\!\$\, \mathsf{iO}(T[x,y])$. It then calls distinguisher D on input $\overline{C}$ and outputs whatever D outputs.

PRG ↓

$\mathsf{Game}_2(\lambda)$**:** As before, except that the game chooses a uniformly random value y. It constructs the predicate circuit $T[x,y] \leftarrow (\mathsf{PRG}(\mathsf{UC}(\cdot,x)) = y)$ and an obfuscation $\overline{C} \leftarrow\!\!\$\, \mathsf{iO}(T[x,y])$. It then calls distinguisher D on input $\overline{C}$ and outputs whatever D outputs.

iO ↓

$\mathsf{Game}_3(\lambda)$**:** As before, except that now an obfuscation of the constant zero-circuit (padded to the length of program $T[x,y]$) $\overline{C} \leftarrow\!\!\$\, \mathsf{iO}(\mathsf{Z}_{|T[x,y]|})$ is constructed. We have reached the target game, that is, the game is identical to the right hand side of Equation (7.1).

What is left to show is that the distinguishing difference between any two consecutive games is negligible. The result then follows by noting that Equation (7.1) can be rewritten as

$$\left|\Pr\left[\mathsf{D}\left(1^\lambda, \mathsf{iO}\left(\mathsf{PRG}(\mathsf{UC}(\cdot,x)) = \mathsf{PRG}(m)\right)\right) = 1\right] - \Pr\left[\mathsf{D}(1^\lambda, \mathsf{iO}(\mathsf{Z})) = 1\right]\right| \leq \sum_{i=2}^{3} \left|\Pr\left[\mathsf{Game}_i^{\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{i-1}^{\mathsf{D}}(\lambda) = 1\right]\right|$$

and that the sum of constantly many negligible functions is again negligible. In the following we consider the difference between each two consecutive games in turn.

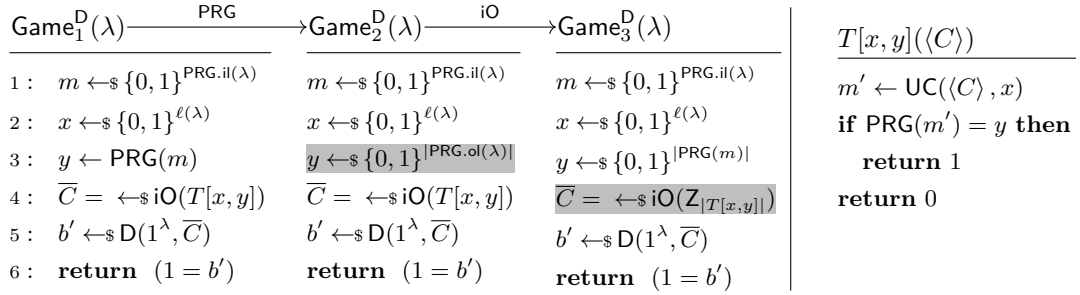


Figure 7.1: The hybrids for the proof of Lemma 7.4 on the left together with the test circuit $T[x,y]$. We highlight the changes between the games with a light-grey background.

Game_1 to Game_2. We bound the distinguishing probability between Game_1 and Game_2 by the security of the pseudorandom generator. For this note that the only difference between Game_1 and Game_2 is the choice of y which is chosen as $\mathsf{PRG}(m)$ (for a uniformly random m) in Game_1 while it is chosen uniformly at random in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ in Game_2. Thus, a distinguisher D between the two games directly yields a distinguisher against the PRG. The distinguisher against the PRG gets as input a value y. It selects a random x in $\{0,1\}^{\ell(\lambda)}$ and creates an obfuscation of circuit $T[x,y]$. It then runs distinguisher D on input the security parameter and the obfuscated circuit and outputs whatever it outputs. If y is chosen as a uniformly random seed to which the PRG was applied then the distinguisher perfectly simulates Game_1 and if y is chosen as a uniformly random value in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ then it perfectly simulates Game_2. Thus, we can bound the distance between the two games as:

$$\left|\Pr\left[\mathsf{Game}_2^{\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathsf{D}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathsf{D}}(\lambda).$$

Game_2 to Game_3. We bound the difference between Game_2 and Game_3 by the security of the indistinguishability obfuscator iO. For this we construct an equivalence circuit sampler $\mathsf{Sam}_{\mathsf{iO}}$ together with a distinguisher D_{iO}. Sampler $\mathsf{Sam}_{\mathsf{iO}}$ runs the steps of Game_2 up-to but not including line 4. It then constructs circuit $T[x,y]$ and zero circuit $\mathsf{Z}_{|T[x,y]|}$ padded to length $|T[x,y]|$. It outputs both circuits. Corresponding distinguisher D_{iO} gets as input an obfuscated circuit $\overline{C}$ and runs game distinguisher D on input the security parameter 1^λ and obfuscated circuit $\overline{C}$. It outputs whatever D outputs.

If $\overline{C}$ is chosen as an obfuscation of circuit $T[x,y]$ then together $\mathsf{Sam}_{\mathsf{iO}}$ and D_{iO} perfectly simulate Game_2 and if it is chosen as an obfuscation of Z they perfectly simulate Game_3. It remains to argue that $\mathsf{Sam}_{\mathsf{iO}}$ is a valid equivalence circuit sampler. For this note that pseudorandom generator PRG is stretching, that is, $\mathsf{PRG.ol}(\lambda) \geq 2 \cdot \mathsf{PRG.il}(\lambda)$. It follows that a random value $y \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ is in the image of the pseudorandom generator only with probability $2^{-\mathsf{PRG.il}(\lambda)}$ and, thus, the difference between Game_2 and Game_3 is upper bounded as:

$$\left|\Pr\left[\mathsf{Game}_3^{\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_2^{\mathsf{D}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam}_{\mathsf{iO}},\mathsf{D}_{\mathsf{iO}}}(\lambda) + \frac{1}{2^{\mathsf{PRG.il}(\lambda)}}.$$

Summing up, we can upper bound the difference in Equation (7.1) by

$$(7.1) \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathsf{D}}(\lambda) + \frac{1}{2^{\mathsf{PRG.il}(\lambda)}} + \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam}_{\mathsf{iO}},\mathsf{D}'}(\lambda)$$

which, by assumption on the security of the pseudorandom generator and indistinguishability obfuscator is negligible. □

7.2.2 On Approximate Obfuscation

An interesting question asked in [BST15] is whether we can construct MB-AIPO[$\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$] if we relax the correctness criterion, that is, if we allow the obfuscator on input (x, m) to output a circuit which is not functionally equivalent to point function $p_{x,m}$. At the extreme, we can require that the obfuscator only needs to be correct on random values (x, m) and with non-negligible probability.

It is not too difficult to see that the above proof carries over to this setting. For this note that the unpredictability analysis is independent of whether or not the obfuscator is correct. For the advantage of distinguisher D we note that if hidden bit $b = 0$ (that is, $\overline{p}$ matches the sampled point message) then D will output 1 with at least the probability that the obfuscator is correct, that is

$$\Pr[\,\mathsf{D}(\overline{p}, \mathsf{aux}) = 1 \mid b = 0\,] \geq \Pr[\mathsf{correct}]\,.$$

In case $b = 1$ the obfuscator gets as input a point message m_1 which is independently chosen of message m_0 which was used to compute $y \leftarrow \mathsf{PRG}(m_0)$ for circuit $T[x, y]$. As any fixed value collides with $\mathsf{PRG}(m_0)$ only with negligible probability (over the choice m_0) down to the PRG probability we have that

$$\Pr[\,\mathsf{D}(\overline{p}, \mathsf{aux}) = 1 \mid b = 1\,] \leq \mathsf{negl}(\lambda)$$

is negligible. This yields the desired result as by assumption $\Pr[\mathsf{correct}]$ is non-negligible.

7.3 On Implications and Circumventions

Average case MB-AIPO[$\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$] is implied by virtual-black-box obfuscation in the presence of auxiliary input (see Section 6.4). Consequently our impossibility result also shows that VBB-AI obfuscation of multi-bit point functions secure in the presence of auxiliary input cannot exist if indistinguishability obfuscation exist:

Corollary 7.5. *If indistinguishability obfuscation exists, then VBB-AI multi-bit output point obfuscation secure with auxiliary input does not exist.*

An immediate second implication is due to the study of Canetti and Dakdouk [CD08] who show that composable AIPO implies the existence of composable MB-AIPO. Applying our result we get the following corollary.

Corollary 7.6. *If indistinguishability obfuscation exists, then composable AIPO*[$\mathcal{S}^{\mathrm{cup}}_{\mathsf{mbpo}}$] *does not exist.*

7.3.1 Interpretation as a Random Oracle Uninstantiability Result

As discussed in the introduction, Lynn et al. [LPS04] construct VBB obfuscators for multi-bit output point functions in the idealized random oracle model and their result can easily be seen to encompass auxiliary information. Thus, assuming indistinguishability obfuscation exists our result rules out the existence of a standard model hash function that can instantiate the random oracle in their construction.

Corollary 7.7. *If indistinguishability obfuscation exists, then the multi-bit output point-function obfuscator by Lynn et al. [LPS04] cannot be instantiated in the standard model so that it achieves MB-AIPO*$[\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}]$ *or VBB-AI security.*

It is interesting to note that this uninstantiability result is very different from the result of Canetti, Goldreich, and Halevi [CGH98] that we discussed and presented in Section 3.4.2. CGH show that there exists a signature scheme in the random oracle model for which the random oracle cannot be instantiated. This, of course, does not imply that secure signature schemes without random oracles cannot exist but that only this particularly crafted signature scheme becomes insecure when the random oracle is replaced by a standard-model hash function. Our result is different in nature and can be better compared to the random oracle impossibility result of Nielsen [Nie02]: we show that the obfuscation notion MB-AIPO$[\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}]$ cannot be met in the standard model assuming indistinguishability obfuscation. However, we have also seen that the same notion, i.e. MB-AIPO$[\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}]$ exists in the random oracle model as the construction of Lynn et al. [LPS04] achieves it (see also Theorem 7.1). Similarly, Nielsen shows that the task of non-interactive, non-committing encryption is infeasible in the standard model but achievable in the random oracle model [Nie02].

This brings us back to our discussion of the *random oracle controversy* from Section 3.5. An often repeated argument in favor of the random oracle methodology is that all known counter examples are contrived, at least to some extent. In their discussion on *provable security* from 2004 [KM06] Koblitz and Menezes make a compelling argument for the use random oracles, reasoning along the lines of what we do to estimate parameters for real-world cryptosystems:

> *"If the top experts in algorithmic number theory at present can factor at most a 576-bit RSA modulus [54], then perhaps we can trust a 1024-bit modulus. If the best implementers of elliptic curve discrete logarithm algorithms have been able to attack at most a 109-bit ECDLP [19], then perhaps we can have confidence in a 163-bit group size. By the same token, if one of the world's leading specialists in provable security (and coauthor of the first systematic study of the random oracle model [6]) puts forth his best effort to undermine the validity for practical cryptography of the random oracle assumption, and if the flawed construction in [4] is the best he can do, then perhaps there is more reason than ever to have confidence in the random oracle model."* [KM04]

The main argument put forth is that the "construction in [6]"—a construction of a hybrid encryption system secure due to Bellare, Boldyreva, and Palacio [BBP04] which they show to be secure in the ROM but insecure with any standard-model hash function—is flawed since it "violates standard cryptographic practice" and similar observations can be made for many counter-examples to the general applicability of the random oracle model [CGH98, GK03, CGH03, GKMZ14]. For our example above (and similarly for the result of Nielsen [Nie02]) such an argument is, however, invalid

as we do not even provide a construction that could violate good cryptographic practice but instead show that an interesting cryptographic primitive that can be trivially constructed in the random oracle model may not exist in the standard model. Indeed, the simple construction of hiding a point by xoring the result of a random oracle query may easily be used (accidentally) within a reduction and in that case it is not clear what implications this may have.

Interestingly, the work of Nielsen is not discussed by Koblitz and Menezes, neither in their 2004 work [KM04] nor in their recent twenty-year retrospective of the random oracle model [KM15]. As we feel that a compelling argument in favor of the random oracle methodology should also be able to explain and interpret our above result as well as the result by Nielsen, we repeat our earlier conclusion that further understanding of the random oracle model and its consequences is necessary such that, hopefully, we can at some point base also the security of practical schemes, that today rely on the random-oracle methodology, on well-studied (standard-model) assumptions.

7.3.2 On Circumventing the Impossibility Result

Several results have been based on the existence of MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] (or composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]). Matsuda and Hanaoka [MH14a] present a CCA-secure public-key encryption scheme based on MB-AIPO and Bitansky and Paneth [BP12] give a three-round weak zero-knowledge protocol for NP based on composable AIPO.[3] We should thus explore whether we can find a plausible weakening for the notion of MB-AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{mbpo}}$] that, on the one hand, still admits for good applications while on the other circumvents our impossibility result. Similarly, an intriguing question is, whether we can instantiate the random oracle in the construction of Lynn et al. to achieve a weaker form of obfuscation, for example, VBB obfuscation without auxiliary information.

Besides their construction relying on MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$], Matsuda and Hanaoka [MH14a] also present a second construction of a CCA-secure PKE scheme based on the weaker assumption of MB-AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{mbpo}}$] that is secure only with respect to statistically unpredictable samples. Indeed, our techniques do not carry over to ruling out MB-AIPO for statistically unpredictable samplers, because the way in which we use indistinguishability obfuscation inherently relies on computational security (see also Theorem 5.7 on page 81). We also note that, based on indistinguishability obfuscation (and one-way functions) we can also show that the random oracle in the construction of Lynn et al. [LPS04] can be securely instantiated when aiming for this weaker notion of statistical MB-AIPO (Chapter 12). Jumping ahead, switching to a statistical notion of security will be a counter-measure also for other impossibility results that we present in this thesis and often allows to recover many of the original results. This will be the case, in particular, for a notion called *Universal Computational Extractors* which we present in detail in Part III.

Going in the same direction, we can also consider restricting the auxiliary information in other ways. Potentially, bounded auxiliary information—for example, if the amount of auxiliary information is restricted to be less than the size of an MB-AIPO—could also be used to circumvent our iO-based impossibility result while preserving a reasonably wide range of applications.

Finally, we note that in Chapter 12 we present a weakened notion of MB-AIPO that we deem to fall outside our impossibility result. Namely, we strengthen the assumption on unpredictable samplers: we require them to remain unpredictable even in case the point value m is given to the predictor

[3] We note that the construction of 3-message witness-hiding protocols from AIPO [BP12] as well as the construction of a CCA secure PKE scheme from lossy encryption schemes and MB-AIPO with statistically hard-to-invert information [MH14a] are not affected by our result.

alongside the auxiliary information aux. This allows us to bypass our impossibility result as we can no longer perform the PRG trick: if the preimage under the PRG is given to the predictor we can no longer replace the PRG value with a truly random value. We call this notion *strong unpredictability* and give a construction based on (non-composable) AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] and indistinguishability obfuscation. Whether or not such a weaker notion suffices for the applications in [BP12, MH14a] is an open question.

7.4 AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] versus Virtual Grey-Box Obfuscation

We have already seen that our impossibility result for MB-AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{mbpo}}$] extends to composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]. On the other hand, it is unclear if our techniques can be further extended to rule out an even larger class of AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] constructions or to even show a similar 1-out-of-2 result also for AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]. However, if one is willing to strengthen the assumption on the general-purpose obfuscator it becomes, indeed, possible to show that a large class of constructions, including the point obfuscation scheme of Canetti [Can97] (Construction 6.1 in Section 6.5) cannot be AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] secure. This extension to our result has, very recently, been presented by Bellare, Stepanovs, and Tessaro (BST; [BST15]) and we use the remainder of this chapter to discuss their ideas.

The idea behind our MB-AIPO impossibility result was to embed a test function $T[x, y]$ into the auxiliary information which, individually, hides point address x but which allows to test point obfuscations as being chosen according to the sampler's distribution or according to a uniform distribution. Let us extend this idea to the AIPO setting. As here point functions only consist of a point address x we would need to place a *test function* $T[x]$ into the auxiliary information that allows to test whether a point-function obfuscation $\overline{p}$ is for point x or not. The straight forward test function

$\underline{T[x](\overline{p})}$
if $\overline{p}(x) = 1$ **then**
  return 1
return $\perp$

is easily seen not work. For this note that function $T[x]$ as defined above is nothing but a universal function that evaluates the (binary) program that it gets as input on input x and returns the result. Thus, the program $\mathsf{P}[\mathsf{prefix}]$ defined as

$$\mathsf{P}[\mathsf{prefix}](x) = \begin{cases} 1 & \text{if } x \text{ begins with } \mathsf{prefix} \\ 0 & \text{otherwise} \end{cases}$$

allows us to recover x bit-by-bit from test function $T[x]$ even in case we only get oracle access to $T[x]$. The reason is, that $T[x]$ does not check whether the submitted program is a valid point function (obfuscation). Consider an adversary getting oracle access to an adapted version of test function $T[x]$ defined as follows

$T[x](\overline{p})$

if $\overline{p}$ is a point function **then**

  return $\overline{p}(x)$

return $\perp$

In order to learn anything from an oracle implementing $T[x]$ an adversary must construct a point function for point x which, intuitively, should be as hard as recovering point x. In particular, in the case where the adversary is given no extra information the two problems are essentially equivalent. Note that this even holds if the adversary is unbounded but only allowed polynomially many queries to its oracle.

These two observations are the starting point of the work of BST [BST15]: 1) we require a setting where the adversary is only given oracle access to the test function, and 2) the test function needs to ensure that it only processes inputs that are valid point functions. The first requirement can be met by strengthening the general-purpose obfuscation assumption to virtual grey-box obfuscation.[4] Remember that VGB obfuscation is stronger than indistinguishability obfuscation (see Propositions 5.9) and that, in contrast to iO, it is simulation based. That is, VGB obfuscation guarantees that for any adversary there exists an (unbounded) simulator that can simulate the adversary with only black-box access to the functionality (cf. Definition 5.4). The second observation yields a requirement on the class of AIPOs that can be attacked. BST introduce the notion of *verifiable obfuscators* which captures such obfuscation schemes that admit for an efficient and deterministic algorithm that allows to verify whether a given circuit is a valid obfuscation, that is, if there exists an input to the obfuscator such that it outputs the given circuit.[5] This is best explained with an example.

Canetti's original point obfuscation scheme [Can97] defines the obfuscation of a point x to be (g, g^x) for a uniformly random generator of some cyclic group $\mathbb{G}$. Given that the underlying group $\mathbb{G}$ is know it can easily be verified that a given obfuscation (g, g^y) is a valid obfuscation (even though it is difficult to learn value y). Consequently, Canetti's point obfuscation scheme is *verifiable*.

In summary, BST show the following result

Theorem 7.8 ([BST15])**.** *If general-purpose VGB obfuscation exists for all circuits in* P/poly*, then no verifiable obfuscator can be AIPO*[$\mathcal{S}^{\text{cup}}_{\text{po}}$] *secure.*

A simple corollary is that Assumption 6.1 (page 132)—the assumption underlying Canetti's point obfuscator for achieving AIPO[$\mathcal{S}^{\text{cup}}_{\text{po}}$]-security–cannot hold in case virtual grey-box obfuscators exist. However, we do not have candidate constructions for full VGB obfuscators but only for VGB obfuscators for NC^1 circuits [BCKP14]. Furthermore, Bellare et al. also present a slightly changed version of Assumption 6.1 which seems immune to their attack. Their idea is that group $\mathbb{G}$ is not fixed for all security parameters but that instead for each obfuscation a fresh group description is generated. This alone does not rule out the above attack if the group generator is *verifiable* where verifiability means that one can efficiently check whether a string d is a description that could be

[4] An interesting open question is whether similar result could also be derived from differing-inputs obfuscation which, similarly to VGB, is stronger than plain indistinguishability obfuscation.

[5] A similar but weaker notion, called *recognizability* was introduced by Bitansky and Paneth [BP12]. Here the existence of an algorithm that given an obfuscation and an unobfuscated circuit can verify that the obfuscation matches the unobfuscated circuit, that is, that there exists some randomness such that the obfuscator on input the unobfuscated circuit and the randomness outputs the obfuscated circuit.

output by the group generator [BST15]. However, for group generators that are not verifiable the above attack seems not to work and for these the extended assumption might hold. Furthermore, the extension of Wee's point obfuscator [Wee05] due to Bitansky and Paneth [BP12] is not known to be verifiable and BST conclude that virtual grey-box obfuscation, at this point, does not seem to be in contention with AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] [BST15].

Chapter 8

Uninstantiability of Encrypt-with-Hash and Fujisaki–Okamoto

> "The universal affirmative, 'Everybody says he's a duck,' is crushed instantly by proving the particular negative, 'Peter says he's a goose'"
>
> Lewis Carroll, A Tangled Tale

Summary. We switch the focus of our random oracle uninstantiability discussion towards random oracle transformations and in particular the Encrypt-with-Hash transformation that turns any randomized public-key encryption scheme into a fully secure deterministic public-key encryption scheme. Assuming indistinguishability obfuscation exists we show that Encrypt-with-Hash is not sound, that is, there are randomized PKE schemes that when transformed with EWH are uninstantiable. We extend the result to a more general class of PKE transformations which, for example, contains the famous Fujisaki–Okamoto transformation for obtaining CCA-secure PKE encryption schemes. Finally, we discuss how our techniques can also be used with transformations for other primitives, for example, random-oracle transformations for symmetric encryption schemes. The results in this chapter are based on a work published at TCC 2015 [BFM15].

Chapter content

8.1 Introduction

In the previous chapter we studied the relation between strong (multi-bit output) point obfuscation schemes and indistinguishability obfuscation and showed a random oracle uninstantiability result for the point obfuscation construction of Lynn et al. [LPS04]. In this chapter we continue the study of uninstantiability of random oracles in the presence of indistinguishability obfuscation, but shift our focus to random oracle transformations. More specifically, we are interested in ROM transformations T^{RO} that take as input *any* standard-model scheme S which is guaranteed to satisfy a mild form of security, and convert S into a new scheme $\mathsf{T}^{\mathsf{RO}}[\mathsf{S}]$ in the random-oracle model that meets a stronger level of security. A fundamental question for such transformations is their instantiability, that is,

whether or not there exists an efficient hash function H such that $\mathsf{T}^{\mathsf{H}}[\mathsf{S}]$ is strongly secure for all mildly secure S. A weaker form of this question is obtained by switching the order of quantifiers, that is, we may ask whether for every scheme S there exists an efficient hash function H such that $\mathsf{T}^{\mathsf{H}}[\mathsf{S}]$ is secure. We show a number of negative results in this direction, which take the following form: there is a mildly secure scheme S^* such that no matter which hash function H is picked, scheme $\mathsf{T}^{\mathsf{H}}[\mathsf{S}^*]$ is provably insecure.

While for the results of the previous chapter indistinguishability obfuscation for circuits was sufficient, our results here come in two flavors depending on the class of programs that the indistinguishability obfuscator supports. Assuming iO for circuits, then for any polynomial p there exists a scheme S_{p} such that for any hash function H of description size at most p the scheme $\mathsf{T}^{\mathsf{H}}[\mathsf{S}_{\mathsf{p}}]$ is insecure. This, in particular, yields an uninstantiability result for any fixed and finite set of hash functions. This result, however, does not rule out instantiating the oracle with hash functions which have larger description size and are in some sense "more complex" than the base scheme. By assuming the existence of iO for *Turing machines* we are able to further strengthen this result to one which rules out instantiations with respect to any, possibly scheme-dependent hash function.

Deterministic Encryption and Encrypt-with-Hash

Our first result establishes the uninstantiability of the Encrypt-with-Hash (EwH) transform of Bellare, Boldyreva, and O'Neill [BBO07]. For a formal introduction to deterministic public-key encryption and the Encrypt-with-Hash transformation we refer to Section 4.5 and here repeat the most important intuitions. The EwH transformation converts a randomized IND-CPA public-key encryption scheme into a deterministic public-key encryption (D-PKE) scheme $\mathsf{D\text{-}PKE}$ by extracting the randomness needed for encryption via hashing the message and the public key; that is, the encryption algorithm $\mathsf{D\text{-}PKE}^{\mathsf{RO}}.\mathsf{Enc}(m, (\mathsf{hk}, \mathsf{pk}))$ first computes random coins $r \leftarrow \mathsf{RO}(\mathsf{hk}, \mathsf{pk} \| m)$ and then invokes the base encryption algorithm on message m, public key pk and random coins r to generate a ciphertext. As shown by Bellare et al. [BBO07] the simple EwH-transformation meets the strongest notion of security that has been proposed for deterministic encryption in the ROM if the underlying encryption scheme is IND-CPA secure. Roughly speaking, a deterministic public-key encryption scheme is IND-secure if no adversary can distinguish the encryptions of two high min-entropy and pk-independent messages. We note that by high min-entropy we mean that the messages cannot be guessed for which super-logarithmic min-entropy is, in fact, sufficient. (In case a scheme is in the random oracle model then the min-entropy requirement is relative to the random oracle, that is, conditioned on the random oracle the min-entropy requirement must hold.) Known standard-model constructions, on the other hand, achieve only weaker levels of security, e.g., security against block sources [BFOR08, BFO08, FOR12].

While in Section 4.5 we presented the transformation in the unkeyed random oracle setting, we present the complete pseudocode here once more in the keyed random oracle setting (cf. Section 3.3.5) which we also assume for the rest of this chapter.

$\underline{\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{KGen}(1^\lambda)}$	$\underline{\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{Enc}((\mathsf{hk}, \mathsf{pk}), m)}$	$\underline{\mathsf{EwH}[\mathsf{PKE}]^{\mathsf{RO}}.\mathsf{Dec}(\mathsf{sk}, c)}$
$\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H}.\mathsf{KGen}(1^\lambda)$	$r \leftarrow \mathsf{RO}(\mathsf{hk}, \mathsf{pk} \| m)$	$m \leftarrow \mathsf{PKE}.\mathsf{Dec}(\mathsf{sk}, c)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE}.\mathsf{KGen}(1^\lambda)$	$c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}, m; r)$	**return** m
return $((\mathsf{hk}, \mathsf{pk}), \mathsf{sk})$	**return** c	

Remark. It is worth noting that we decided to treat the hash key hk as part of the public-key of the transformation. A different setup would be to treat the key as a public-parameter which is chosen independently of the keys. As the first choice is more restrictive towards the adversary we go for this setting as for a negative result the more we can restrict the adversary against the encryption scheme, the stronger the result. For this note that in the IND security notion for D-PKE the adversary must choose its messages independent of the public key, and when treating the hash key as part of the public key they must, in particular, also be chosen independent of the hash key.

We ask if any hash function can be used to instantiate the random oracle within the EwH transform to achieve IND security. Assuming iO for circuits/Turing machines, we build an IND-CPA-secure encryption scheme such that when the EwH transform is applied and the result is instantiated by any (p-bounded) hash function, then the resulting scheme is *not* secure, not even for block-sources or adversaries that are restricted to output only a single message.

In more details, starting with an arbitrary scheme PKE we consider a new scheme PKE^* which includes as part of its ciphertexts an indistinguishability obfuscation of the program

$\underline{\mathsf{P}[\mathsf{pk}, m, \mathsf{PRG}(r)](\langle \mathsf{H}_{\mathsf{hk}} \rangle)}$

$s' \leftarrow \mathsf{PRG}(\mathsf{UP}(\langle \mathsf{H}_{\mathsf{hk}} \rangle, \mathsf{pk} \| m))$

if $s' = \mathsf{PRG}(r)$ **then**

  return m

return 0

Here m is the plaintext message, pk the public key and r the randomness used for encryption (the program only stores the value $\mathsf{PRG}(r)$ but not r itself). Note that the program is very similar to the test function $T[x, y]$ used in the previous chapter but with two changes. On the one hand, the program takes as input the description of a hash function H_{hk} (denoting hash function H with key hk hard-coded) instead of a point function. This is, of course, only a semantic change since neither here nor in the results of the previous chapter do we test what type of function is provided as input.[1] The second change is that instead of returning a Boolean value, it returns the encrypted message m when the check passes. In other words, for program $\mathsf{P}[\mathsf{pk}, m, \mathsf{PRG}(r)]$ the public key pk, the message m and the PRGed randomness $\mathsf{PRG}(r)$ (i.e., the result of the computation) are parameters. The program takes as input a hash function H_{hk} (with a hard-coded key hk) and evaluates H_{hk} on $\mathsf{pk}||m$ to obtain some value s'. Then, it applies PRG to s' and checks whether $\mathsf{PRG}(s')$ is equal to the hard-wired value $\mathsf{PRG}(r)$. If this is the case, it returns the message m. Else, it returns 0.

We can use an obfuscation of this program to attack the security of $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ as follows: the adversary runs this program on the description H_{hk} of the hash function that is used in the instantiation (with hard-coded hk) to obtain the encrypted message. (Note that this (second-stage) adversary gets to see the public encryption key and hence also hash key hk.)

A corollary of this result is that under iO, no security assumption (single or multi-staged, falsifiable or not) is strong enough to build D-PKEs via EwH. We remark that our results are incomparable to those of Wichs [Wic13] who shows an unconditional unprovability result for D-PKEs using *arbitrary*

[1] We note that, indeed, for the extension due to Bellare et al. [BST15] which considers AIPO security in the presence of VGB obfuscation such a check is necessary (cf. Section 7.4).

techniques from single-stage assumptions. (Our results are conditional and show uninstantiability of EwH regardless of the assumptions used.) Finally, we note that our result naturally extends to the Randomized-Encrypt-with-Hash transform for building hedged PKEs [BBN+09].

Beyond EwH: The Fujisaki—Okamoto Transform

The above result generalizes to a wider class of (possibly randomized) transformations that use their underlying PKE schemes in a structured way and admit recovery algorithms that satisfy certain correctness properties. We call such transformations *admissible*. (See Section 8.3 for the details.) Somewhat surprisingly, the Fujisaki–Okamoto (FO) transform [FO99] for converting CPA into CCA security is admissible and, thus, suffers from uninstantiability. The FO transform, which dates back to the 1990s, is a simple and flexible technique to boost security of various schemes and has been widely used: for example, in identity-based encryption [BF01], its hierarchical and fuzzy variants [GS02, SW05], forward-secure encryption [CHK03], or certificateless and certificate-based encryption [ARP03, Gen03]. Our results, once again, come in two flavors depending on the strength of the underlying obfuscator. Our techniques can be further tweaked to show that one cannot instantiate the oracle used within the asymmetric component of the FO transform. This in particular means that the POW-encryption assumption of Boldyreva and Fischlin [BF05] used for partial instantiation of the oracles in FO is also uninstantiable if iO for Turing machines/circuits exists. In other words, the POW-encryption assumption may hold in the random oracle model but it does not hold for any standard model perfectly one-way (POW) hash function.

Comparison with CGH

Recall that Canetti, Goldreich, and Halevi (CGH) [CGH98] show the uninstantiability of certain ROM digital signature and encryption schemes without relying on iO. (We present their result in detail in Section 3.4.2.) Their technique is to give a (contrived) scheme that is secure in the random oracle model but behaves anomalously on certain inputs that are related to a compact description of the hash function. Our uninstantiability results share these features, that is, neither their nor our uninstantiability results apply to "natural" schemes. (For instance, it is not known if Encrypt-with-Hash when used with ElGamal is uninstantiable or not.) On the other hand, our results apply to natural transformations.

It is natural to ask if CGH-like techniques can be directly applied here so as to obtain uninstantiability results that do not rely on the iO machinery. For uninstantiability with respect to *unkeyed* hash functions, one can indeed construct anomalous PKE schemes which follow the CGH paradigm and give the desired uninstantiability result for Encrypt-with-Hash. For keyed hash functions, on the other hand, there seems to be an inherent limitation to CGH-like techniques. For instance, the security model for D-PKEs do *not* allow message distributions to depend on the hash key as this value is included in the public key and the latter is not given to the first-stage adversary. Consequently there is no way to generate messages which contain a *full* description of the hash function used, *including its key*, which seems to be necessary when applying CGH-like techniques. It might appear that this issue can be easily resolved by noting that the encryption routine *does* have access to the hash key and a full description of the hash function can be formed at this point. The caveat, however, is that such an uninstantiable scheme no longer falls under the umbrella of schemes arising from the Encrypt-with-Hash transform. More precisely, although we can freely modify the base PKE to prove

uninstantiability, the transformation is *fixed* and it only allows black-box access to the hash function and denies encryption access to the hash key.[2] This observation applies to other transformations as well. For instance, in the FO transformation the message that is asymmetrically encrypted is chosen uniformly at random and thus cannot be set to the description of the hash function. To summarize, although the description of the hash function will be eventually made public, the adversarial scheme never gets to the hash function in full and to be successful it needs to coordinate the attack with the actual adversary (who does see the hash key). Indistinguishability obfuscation allows this distributed attack to be carried out.

Concurrent work

In concurrent and independent work, Green et al. [GKMZ14] use iO and techniques similar to ours to demonstrate the uninstantiability of random-oracle schemes. Like us, they embed an obfuscated program into schemes in order to make them uninstantiable. Our results, however, rule out the instantiability of (existing) random-oracle *transformations* whereas Green et al. construct uninstantiable *schemes* for primitives which cannot be targeted with CGH-like techniques. For instance bit encryption falls outside the reach of CGH as its input space is too short. Green et al. show that indistinguishability obfuscation can be used to extend CGH to such constrained primitives when opting for weaker p-type uninstantiability results.

In a second concurrent and independent work Bellare and Hoang [BH15] study how to construct deterministic public-key encryption from *universal computational extractors*, a notion that we discuss in detail in Part III. Besides their positive results, they also present an uninstantiability result for Encrypt-with-Hash, namely they show that for any hash function H there exists a public-key encryption scheme PKE^* such that $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ is uninstantiable. Note that this form of uninstantiability is significantly weaker than p-uninstantiability.

Outline

We begin our study of uninstantiable transformations with Encrypt-with-Hash in Section 8.2 and show how the techniques can be generalized to capture a large class of transformations in Section 8.3; this class also contains the Fujisaki–Okamoto transformation (Section 8.4). In the following, Section 8.5, we ask whether our results can be circumvented by transformations and present an extension to Encrypt-with-Hash which intuitively works around the result. However, we can show that an extended attack also applies here. Finally, in Section 8.6 we show how to extend our techniques also to transformations for symmetric encryption schemes.

8.2 Uninstantiability of Encrypt-with-Hash

When the EwH transformation is instantiated with an *unkeyed* random oracle a CGH-style uninstantiability result can be directly established [CGH98]. (This, in particular, shows that the use of a keyed hash function is necessary to instantiate EwH.) Given an arbitrary PKE scheme PKE, consider a tweaked variant of it PKE' which first interprets parts of the message m as the description of a

[2] Despite this, CGH-like techniques render Encrypt-with-Hash uninstantiable when stronger notions of security are considered [RSV13].

hash function H (together with its single key) and checks if the provided random coins r match the hash value $\mathsf{H}(\mathsf{pk}\|m)$. If so, it returns $0\|m$ and else it returns $1\|\mathsf{PKE.Enc}(\mathsf{pk}, m; r)$. Scheme PKE' is still IND-CPA secure because the probability that a truly random value r matches $\mathsf{H}(\mathsf{pk}\|m)$ is negligible. On the other hand, when the random coins are generated deterministically by applying a hash function, an IND adversary which asks for encryptions of $m_i\|\mathsf{H}$ for any two high min-entropy messages m_0 and m_1 which differ, say, on their most significant bits can easily win the game.[3] The standard IND game, however, restricts the first-stage adversary not to learn the public key, and thus, it cannot guess the (high min-entropy) hash key.

We show how to use indistinguishability obfuscation to extend the above uninstantiability to keyed hash functions. As mentioned in the introduction, our result comes in weak and strong flavors depending on the programs that the obfuscator is assumed to support. Assuming iO for Turing machines we obtain a strong uninstantiability result: there exists an IND-CPA encryption scheme that cannot be securely used in EwH in conjunction with *any* keyed hash function. Assuming the weaker notion of iO for circuits, we get p-uninstantiability: for any polynomial bound p there exists an IND-CPA scheme that cannot be securely used in EwH for any hash function whose description size is at most p. The latter result is also quite strong as, in particular, it means that for any finite set of hash functions (e.g., those which are standardized), we can give a PKE scheme that when used within EwH yields an insecure D-PKE scheme for any choice of hash function from this set. We note that the adversarial PKE scheme that we construct depends only on an upper bound on description sizes and not on their implementation details.

Theorem 8.1 (Uninstantiability of EwH)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp. p-bounded circuits $\mathcal{C}_{\mathsf{p}}$), the* EwH *transform is uninstantiable (resp. p-uninstantiable) with respect to* IND *security and* IND-CPA *base schemes in the standard model.*

We start by giving a high-level description of the proof before presenting the details. We may assume, without loss of generality, that an IND-CPA-secure PKE scheme exists as otherwise uninstantiability trivially holds. This, in turn, implies that we can also assume the existence of a secure pseudorandom generator.

Now given an IND-CPA-secure PKE scheme PKE, we construct a tweaked scheme PKE^* that is also IND-CPA secure but the D-PKE scheme $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ fails to be IND secure. To construct the adversarial scheme PKE^* we follow a similar strategy to CGH. The fundamental difference here is that $\mathsf{PKE}^*.\mathsf{Enc}$ does not have access to the hash key. To overcome this problem, we consider the obfuscation of a program P' that implements a variant of the test circuit $T[x, y]$ used in the previous chapter, i.e., it takes as input the description of a hash function $\mathsf{H}(\mathsf{hk}, \cdot)$, with a hard-wired key, runs it on the concatenation of values m and pk embedded into P', and outputs m if the result matches a third hard-wired value r:

$$\mathsf{P}'[\mathsf{pk}, m, r]\left(\mathsf{H}(\mathsf{hk}, \cdot)\right) : \text{if } \mathsf{H}(\mathsf{hk}, \mathsf{pk}\|m) = r \text{ return } m \text{ else return } 0\ .$$

The tweak that we introduce in PKE^* is that the encryption operation appends obfuscations of $\mathsf{P}'[\mathsf{pk}, m, r]$ to its ciphertexts, where pk, m and r are the values input to the encryption routine.

[3] This attack generalizes to the setting where the first-stage adversary can guess the hash key with non-negligible probability and in particular, EwH is uninstantiable with respect to the stronger IND model, as proposed in [RSV13].

We need to argue that (1) this tweak allows an adversary to break the scheme whenever the hash function is instantiated and (2) outputting such an indistinguishability obfuscation of P' does not hurt the IND-CPA security of PKE^*.

For (1), note that given an obfuscation of $\mathsf{P}'[\mathsf{pk}, m, r]$ and a description of $\mathsf{H}(\mathsf{hk}, \cdot)$, an adversary can recover m by running the above circuit on a description of $\mathsf{H}(\mathsf{hk}, \cdot)$. Now the second stage of the IND adversary gets the public key and thus the description of the hash function $\mathsf{H}(\mathsf{hk}, \cdot)$. Furthermore, it also gets a ciphertext which contains an obfuscation of $\mathsf{P}'[\mathsf{pk}, m, r]$. Hence, the second-stage adversary has all the information needed to break the IND security of the deterministic encryption scheme $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$.

Now this insecurity might have nothing to do with the transform if the tweaked scheme PKE^* that we introduce is already insecure. Hence, to rule out this possibility, we also need to argue that PKE^*, as a randomized encryption scheme, is IND-CPA secure. For this, we would like to argue that for a *truly random* r—such an r is used in randomized encryption—P' implements the constant zero program Z. Indeed, if r is sufficiently longer than $|\mathsf{pk}| + |m|$ then for any fixed $\mathsf{H}(\mathsf{hk}, \cdot)$, over a random choice of r the check performed by P' would fail with all but negligible probability. This, however, does not necessarily mean that the circuit is functionally equivalent to Z as there could *exist* a hash function $\mathsf{H}(\mathsf{hk}, \cdot)$ which passes the check.

To resolve this issue, we consider a further tweak to the base scheme. We consider a scheme which has a much smaller randomness space and instead uses coins that are *pseudorandomly generated*. This ensures that the randomness space used by PKE is sparse within the set of all possible coins, allowing a counting argument to go through. We adapt the program above to cater for the new tweaks:

$$\mathsf{P}[\mathsf{pk}, m, \mathsf{PRG}(r)]\left(\mathsf{H}(\mathsf{hk}, \cdot)\right) : \text{if } \mathsf{PRG}(\mathsf{H}(\mathsf{hk}, \mathsf{pk} \| m)) = \mathsf{PRG}(r) \text{ return } m \text{ else return } 0 \ .$$

At this point it might appear that no progress has been made as the above program, for reasons similar to those given above, is not functionally equivalent to Z. We note, however, that for a truly random $s \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ the program $\mathsf{P}[\mathsf{pk}, m, s]$ has a *description* which is indistinguishable from that of $\mathsf{P}[\mathsf{pk}, m, \mathsf{PRG}(r)]$ down to the security of the pseudorandom generator PRG. Furthermore, for such an s, this program *can* be shown to be functionally equivalent to the zero circuit with overwhelming probability as s will be outside the range of the PRG with overwhelming probability. These two steps allow us to prove that obfuscations of P leak no information about m, and show that scheme PKE^* is IND-CPA secure.

Formally, program P will use a universal program evaluator to run its input hash-function descriptions. If the (obfuscated) program is a Turing machine, it can be run on arbitrary large descriptions and arbitrarily sized hash functions are ruled out. On the other hand, if the program is a circuit, it has an a priori *fixed* input length, and thus can only be run on hash functions that respect the input-size restrictions. We formalize this proof intuition next.

Proof of Theorem 8.1. Let PKE be an IND-CPA-secure public-key encryption scheme, PRG be a pseudorandom generator of appropriate stretch and iO be an indistinguishability obfuscator supporting either Turing machines or circuits. We define a modified PKE scheme PKE^* as follows. The key-generation algorithm is unchanged. The adapted encryption algorithm is defined as shown below by appending an obfuscated program $\overline{\mathsf{P}}$ to its outputs. UP denotes a universal program evaluator. The modified decryption algorithm ignores the $\overline{\mathsf{P}}$ component and decrypts as in the base scheme.

$\underline{\mathsf{PKE}^*.\mathsf{Enc}(\mathsf{pk}, m; r\|r')}$

$s \leftarrow \mathsf{PRG}(r)$

$c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}, m; s)$

$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[\mathsf{pk}, m, s](\cdot); r')$

return $(c, \overline{\mathsf{P}})$

$\underline{\mathsf{P}[\mathsf{pk}, m, s](\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle)}$

$r\|r' \leftarrow \mathsf{UP}(\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \mathsf{pk}\|m)$

$s' \leftarrow \mathsf{PRG}(r)$

if $(s' = s)$ **then return** m

return 0

When we consider the above construction with respect to circuits, we need to specify an extra parameter p that upper-bounds the size of the inputs to the universal circuit evaluator. This maximum size of programs that the universal circuit admits corresponds to the maximum size of the hash functions that our uninstantiability proof applies to. Note that when the construction is considered for Turing machines, the input size for program P is arbitrary.

We show that the above tweaked scheme PKE^* is IND-CPA secure via a sequence of four games that we describe next. We present the pseudocode in Figure 8.1.

PRG

$\mathsf{Game}_1(\lambda)$**:** This game is identical to the IND-CPA game for the randomized base scheme PKE^* and an arbitrary adversary $\mathcal{A}$.

$\mathsf{Game}_2(\lambda)$**:** In this game the randomness s used for encryption within the underlying PKE scheme is no longer generated via a PRG call, but is sampled uniformly at random.

iO

$\mathsf{Game}_3(\lambda)$**:** In this game the ciphertext component $\overline{\mathsf{P}}$ is generated as an indistinguishability obfuscation of the zero program (that is, Turing machine or circuit) Z padded to the appropriate length (and running time).

Remark. We note that the proof is analogous to the proof of Lemma 7.4 with the exception that we here, in the final step, give a reduction to the IND-CPA security of the underlying scheme. We note that care needs to be taken, when considering Turing machine obfuscation in order to make the runtime of the zero circuit equivalent to that of program P.

We now show that each of the above transitions negligibly changes the game's output with respect to any adversary $\mathcal{A}$ and that in the final setting (Game_3) the distribution is negligibly close to $\frac{1}{2}$.

Game_1 to Game_2. We bound the difference in these games by the security of pseudorandom generator PRG. Note that a PRG adversary that gets as input y, a PRG image under a uniformly random seed or a truly uniformly random value, can perfectly simulate games Game_1 and Game_2 for adversary $\mathcal{A}$ by using y in place of s. If y is a PRG image, then Game_1 is run and if y is uniformly random then Game_2 is run:

$$\left|\Pr\left[\mathsf{Game}_2^{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathsf{D}}(\lambda).$$

Here D denotes the induced PRG adversary.

Game_2 to Game_3. We show that this hop negligibly affects the winning probability of $\mathcal{A}$ down to the security of the indistinguishability obfuscator. We let Sam be the sampler which runs all the steps of Game_2 using the first phase of $\mathcal{A}$ up to the generation of $\overline{\mathsf{P}}$. It then sets $\mathsf{P}_0 := \mathsf{P}[\mathsf{pk}, m_b, s]$, $\mathsf{P}_1 := \mathsf{Z}_{|\mathsf{P}_0|}$ and aux to be the ciphertext component c and the internal state of the first phase of

$\mathsf{Game}_1^{\mathcal{A}}(\lambda)$	$\xrightarrow{\text{PRG}}$	$\mathsf{Game}_2^{\mathcal{A}}(\lambda)$	$\xrightarrow{\text{iO}}$	$\mathsf{Game}_3^{\mathcal{A}}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$		$b \leftarrow\!\!\$\ \{0,1\}$		$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$		$(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$		$(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
$(m_0,m_1) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$		$(m_0,m_1) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$		$(m_0,m_1) \leftarrow\!\!\$\ \mathcal{A}(\mathsf{pk})$
$r\|\|r' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$		$r\|\|r' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$		$r\|\|r' \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}$
$s \leftarrow \mathsf{PRG}(r)$		$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$		$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$
$c \leftarrow \mathsf{PKE}(\mathsf{pk}, m_b; s)$		$c \leftarrow \mathsf{PKE}(\mathsf{pk}, m_b; s)$		$c \leftarrow \mathsf{PKE}(\mathsf{pk}, m_b; s)$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[\mathsf{pk}, m_b, s]; r')$		$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[\mathsf{pk}, m_b, s]; r')$		$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{Z}_{\lvert\mathsf{P}[\mathsf{pk},m_b,s]\rvert}; r')$
$b' \leftarrow\!\!\$\ \mathcal{A}(c, \overline{\mathsf{P}})$		$b' \leftarrow\!\!\$\ \mathcal{A}(c, \overline{\mathsf{P}})$		$b' \leftarrow\!\!\$\ \mathcal{A}(c, \overline{\mathsf{P}})$
return $(b' = b)$		**return** $(b' = b)$		**return** $(b' = b)$

$\mathsf{P}[\mathsf{pk}, m_b, s](\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle)$

$r\|\|r' \leftarrow \mathsf{UP}(\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \mathsf{pk}\|\|m_b)$
$s' \leftarrow \mathsf{PRG}(r)$
if $(s' = s)$ **then return** m_b
return 0

Figure 8.1: Hybrids used in the proof of Theorem 8.1 (left) and the program P obfuscated in the first two games (right). The highlighted lines show the changes in game transitions.

the IND-CPA adversary. Algorithm D receives an obfuscation $\overline{\mathsf{P}}$ of either P_0 or P_1, and resumes the second phase of $\mathcal{A}$ on $(c, \overline{\mathsf{P}})$ using the state recovered from aux. When P_0 is obfuscated $\mathcal{A}$ is run according to the rules of Game_2 and when P_1 is obfuscated $\mathcal{A}$ is run according to the rules of Game_3. Hence,

$$\left|\Pr\left[\mathsf{Game}_3^{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_2^{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}}(\lambda).$$

We must show that sampler Sam outputs functionally equivalent circuits with overwhelming probability. Assuming that the stretch of the PRG is sufficiently large, i.e., $\mathsf{PRG.ol}(\lambda) \geq 2 \cdot \mathsf{PRG.il}(\lambda)$, by the union bound the probability over a random choice of s that there *exists* an $r \in \{0,1\}^{\mathsf{PRG.il}(\lambda)}$ such that $\mathsf{PRG}(r) = s$ is upper bounded by $2^{\mathsf{PRG.il}(\lambda) - \mathsf{PRG.ol}(\lambda)} \leq 2^{-\mathsf{PRG.il}(\lambda)}$. Hence, the probability that P_0 is functionally inequivalent to the zero circuit is upper bounded by $2^{-\mathsf{PRG.il}(\lambda)}$, that is,

$$\Pr\left[\exists x\, \mathsf{P}_0(x) \neq 0 : (\mathsf{P}_0, \mathsf{P}_1, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)\right] \leq 2^{-\mathsf{PRG.il}(\lambda)}.$$

When working with Turing machines, we also need to ensure that the two programs used above are not just of equal size but have an identical runtime. We have several options to achieve this, the simplest being a variation of what we did in our proof of the CGH result in Section 3.4.2. That is, while we consider program P to be encoded as a Turing machine, we require that it takes as input descriptions of circuits with an encoding that is proportional to the circuit size. Consequently the program implements the universal circuit instead of a universal Turing machine in order to run the circuit description of hash function H_{hk}. The final tweak is to consider program P to be implemented as an oblivious Turing machine such that its runtime only depends on the size of the input circuit

description. Now, when constructing the zero circuit we can construct the very same program but also return 0 in case the check succeeds. That is, we consider a sampler that outputs the following two programs encoded as oblivious Turing machines:

$\mathsf{P}[\mathsf{pk}, m, s](\langle \mathsf{H}(\mathsf{hk}, \cdot) \rangle)$	$\mathsf{Z}[\mathsf{pk}, s](\langle \mathsf{H}(\mathsf{hk}, \cdot) \rangle)$
$r \Vert r' \leftarrow \mathsf{UC}(\langle \mathsf{H}(\mathsf{hk}, \cdot) \rangle, \mathsf{pk} \Vert m)$	$r \Vert r' \leftarrow \mathsf{UC}(\langle \mathsf{H}(\mathsf{hk}, \cdot) \rangle, \mathsf{pk} \Vert 0^{\mathsf{PKE.il}(\lambda)})$
$s' \leftarrow \mathsf{PRG}(r)$	$s' \leftarrow \mathsf{PRG}(r)$
if $(s' = s)$ **then return** m	**if** $(s' = s)$ **then return** 0
return 0	**return** 0

As the runtime of UC is only related to the (length of the) circuit description and not the additional inputs we have that the runtime of two programs, indeed, coincides on any input.

Game_3**.** We reduce the advantage of adversary $\mathcal{A}$ in Game_3 to the IND-CPA security of scheme PKE. The only difference between this game and the usual IND-CPA game for PKE is that an obfuscation of $\mathsf{Z}_{|\mathsf{P}[\mathsf{pk}, m_b, s]|}$ is attached to the ciphertexts. This program has a public description and hence its obfuscations can be perfectly simulated. It follows that

$$2 \cdot \Pr\left[\mathsf{Game}_3^{\mathcal{A}}(\lambda) = 1\right] - 1 \leq \mathsf{Adv}^{\text{ind-cpa}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$$

where $\mathcal{B}$ denotes the induced IND-CPA adversary.

Attacking IND security. To conclude the proof, we show there exists an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ that breaks the IND security of $\mathsf{D\text{-}PKE}^* := \mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ for any (p-bounded) function H. Adversary $\mathcal{A}_1$ chooses two values $x_0, x_1 \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$ uniformly at random and outputs messages $m_0 := x_0 \Vert 0$ and $m_1 := x_1 \Vert 1$. Observe that $\mathcal{A}_1$ adheres to the entropy requirements of admissible IND adversaries. Adversary $\mathcal{A}_2$ gets as input the public key $(\mathsf{hk}, \mathsf{pk})$ and a ciphertext $(c, \overline{\mathsf{P}})$. It then evaluates $\overline{\mathsf{P}}$ on a circuit description of hash function $\mathsf{H}(\mathsf{hk}, \cdot)$ with key hk recovered from the public key and hard-coded into the program description. (Note that if we are considering indistinguishability obfuscation only for circuits, then description of hash function H must have size at most $\mathsf{p}(\lambda)$.) Adversary $\mathcal{A}_2$ returns the least significant bit of P's output. This adversary and its operation within the IND game is shown in Figure 8.2. By the correctness of the obfuscator, $(\mathcal{A}_1, \mathcal{A}_2)$ always win IND with probability 1 irrespectively of the message that is encrypted:

$$\mathsf{Adv}^{\text{ind}}_{\mathsf{D\text{-}PKE}^*,\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1 \ .$$

□

8.2.1 Extension to Hedged PKEs

Hedged public-key encryption, introduced by Bellare et al. [BBN+09] models the security of public-key encryption schemes where the random coins used in encryption might have low entropy. Indistinguishability under chosen-distribution attacks (IND-CDA) shown in Figure 8.3 formalizes the security of hedged PKEs. This notion is similar to IND and the only difference is that the adversary additionally to the two message vectors also outputs a randomness vector. The high min-entropy

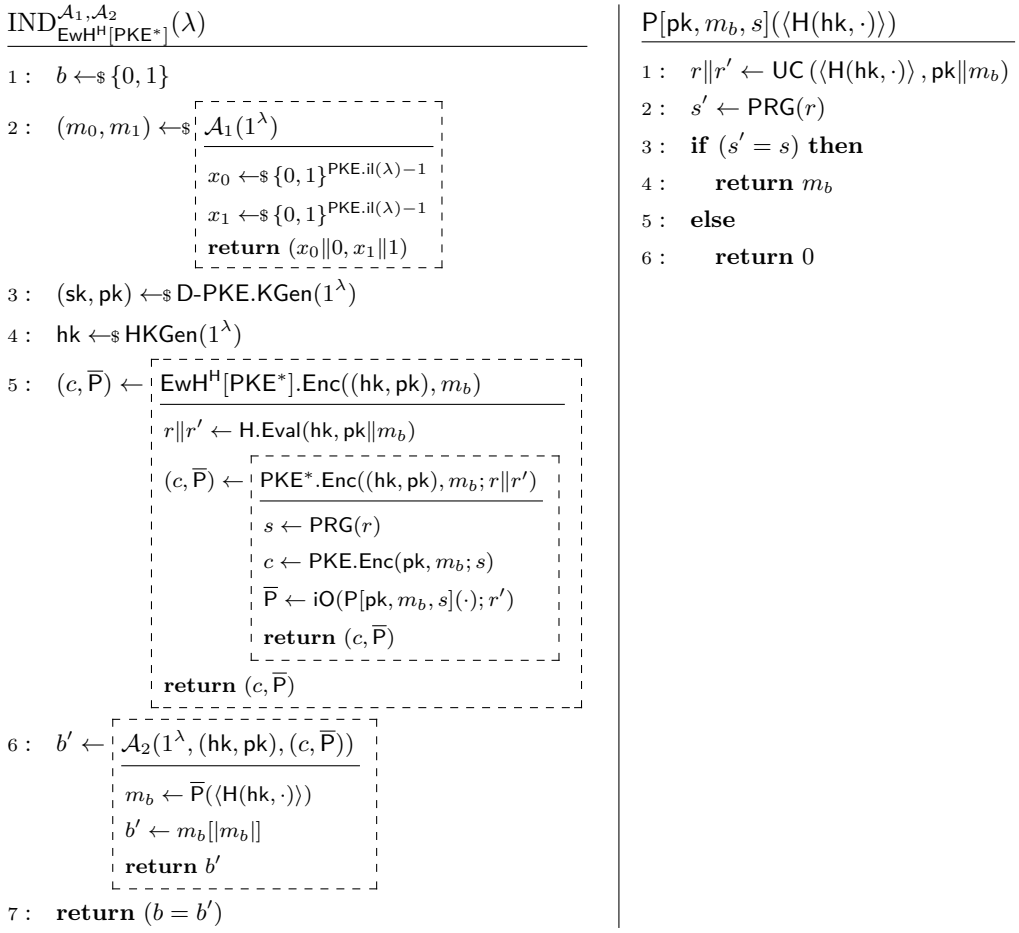


Figure 8.2: The IND security game for scheme $\mathsf{EwH}^{\mathsf{H}}[\mathsf{PKE}^*]$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 8.1. The boxed algorithms are to be understood as subroutines. Program P that is obfuscated as part of ciphertexts is given on the right.

restriction is spread over the message and randomness vectors. When the length of the randomness entries is 0, one recovers the IND model for D-PKEs. A transform similar to EwH, called Randomized Encrypt-with-Hash, can be defined for hedged PKEs [BBN+09]: hash the message, public key and the randomness to obtain new coins, and use them for the encryption operation. Our uninstantiability result can be immediately adapted to this transform as long as the message space has super-logarithmic size:

$\mathsf{P}[\mathsf{pk}, m, s](\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \rho)$
$r \leftarrow \mathsf{UP}(\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \mathsf{pk}\|m\|\rho)$
$s' \leftarrow \mathsf{PRG}(r)$
if $(s' = s)$ **then return** m
return 0

Observe that the program takes an additional input ρ that allows the attacker to specify the randomness. We note that this requires the adversary to choose the randomness in a predictable way,

$\underline{\text{IND-CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{H\text{-}PKE}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{H\text{-}PKE.KGen}(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $\mathbf{c}[i] \leftarrow \mathsf{H\text{-}PKE.Enc}(\mathsf{pk}, \mathbf{m}_b[i]; \mathbf{r}[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{pk}, \mathbf{c})$
return $(b' = b)$

Figure 8.3: The IND-CDA security game for hedged public-key encryption without initial adversaries. Our results carry over to a setting where an initial adversary that passes state to the first and second phase of the attack is present [RSS11].

which does not violate the min-entropy requirements as long as the min-entropy of the messages is sufficiently high. We note that if one strengthens the IND-CDA notion to require the randomness distribution to have super-logarithmic min-entropy, our attacks would no longer work. This in particular is the case if the message space of the scheme is small.

8.3 Beyond Encrypt-with-Hash

We show that our uninstantiability results can be further leveraged to rule out standard-model instantiations of a number of other known transformations. We generalize the iO attack of the previous section to what we call *admissible* transformations, and show that the classical and widely used Fujisaki–Okamoto transformation [FO99] falls under it. Later in Section 8.6 we show that our techniques can be used beyond transformations for public-key encryption schemes.

8.3.1 Generalizing the Attack

Let $\mathsf{GT}^{\mathsf{RO}}[\mathsf{PKE}]$ be a ROM transformation mapping PKE schemes to PKE schemes. Without loss of generality we assume that there is a single random oracle as multiple random oracles can be simulated via domain separation (see Section 3.3.3). When the oracle in $\mathsf{GT}^{\mathsf{RO}}[\mathsf{PKE}]$ is instantiated with a hash function H we write $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}]$. We say that transform GT is *structured* if it takes the following form for a (possibly randomized) oracle PPT machine T^{RO} and a public-key encryption scheme PKE.

$\underline{\mathsf{GT}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{Enc}(\mathsf{pk}, m; r)}$

$(\mathsf{pk}', m', r', c') \leftarrow \mathsf{T}^{\mathsf{RO}}(\mathsf{pk}, m; r)$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}', m'; r')$
return (c, c')

Note that in order to obtain a deterministic encryption routine, T needs to be deterministic (i.e., $r = \varepsilon$). For the sake of generality, however, we allow T to be randomized.

In addition to the above structural requirement on $\mathsf{GT}^{\mathsf{RO}}$, we require the existence of a deterministic *recovery* algorithm $\mathcal{R}^{\mathsf{RO}}$ such that experiment RECOVER returns true with probability 1 for any valid choice of message m.[4]

$$\begin{array}{l}\underline{\text{EXP. RECOVER}_{\mathsf{PKE},\mathsf{T},\mathcal{R},m}(\lambda)}\\ \mathsf{RO} \leftarrow\!\!\$\, \mathrm{Func}(\mathsf{kl}(\lambda), \mathsf{il}(\lambda), \mathsf{ol}(\lambda))\\ r \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PKE.rl}(\lambda)};\ (\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\, \mathsf{PKE.KGen}(1^\lambda)\\ (\mathsf{pk}', m', r', c') \leftarrow \mathsf{T}^{\mathsf{RO}}(\mathsf{pk}, m; r)\\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}', m'; r')\\ (m^*, r^*) \leftarrow \mathcal{R}^{\mathsf{RO}}(\mathsf{pk}', m', c, c')\\ \textbf{return } (m^* = m \wedge r^* = r').\end{array}$$

Note that $\mathcal{R}^{\mathsf{RO}}$ gets to see the adapted public key pk', the adapted message m' and the ciphertext (c, c') as computed by a structured transformation $\mathsf{GT}^{\mathsf{RO}}[\mathsf{PKE}]$ but it does *not* get to see the secret key sk nor the original randomness r (if present). Procedure Func in the first line of the experiment denotes that a random oracle with appropriate key, input, and output length is chosen. We call a transformation *admissible* if it is structured and admits a recovery algorithm.

As an example, let us check that the Encrypt-with-Hash transformation is admissible. The encryption operation of Encrypt-with-Hash is given by

$$\mathsf{EwH}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{Enc}(\mathsf{pk}, m) := \mathsf{PKE.Enc}(\mathsf{pk}, m; \mathsf{RO}(\mathsf{pk}\|m))\ .$$

This can be re-written in the above structured form as follows.

$$\begin{array}{l}\underline{\mathsf{EwH}^{\mathsf{RO}}[\mathsf{PKE}].\mathsf{Enc}(\mathsf{pk}, m; \varepsilon)}\\ (\mathsf{pk}', m', r', c') \leftarrow \boxed{\begin{array}{l}\underline{\mathsf{T}^{\mathsf{RO}}(\mathsf{pk}, m; \varepsilon)}\\ r' \leftarrow \mathsf{RO}(\mathsf{pk}\|m)\\ \textbf{return } (\mathsf{pk}, m, r', \varepsilon)\end{array}}\\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}', m'; r')\\ \textbf{return } (c, c')\end{array}$$

Note that EwH is deterministic, and so we set $r = \varepsilon$. The recovery algorithm $\mathcal{R}^{\mathsf{RO}}$ on input $(\mathsf{pk}', m', c, c')$ outputs

$$\Big(m', \mathsf{RO}(\mathsf{pk}'\|m')\Big)\ .$$

As $m' = m$ and $\mathsf{pk}' = \mathsf{pk}$ this is the required output in order to succeed in experiment RECOVER.

We now give our generalized uninstantiability result. Since we consider both randomized and deterministic transformations, in the former case we show that the scheme resulting from applying the transformation to an adversarial scheme is not IND-CPA secure, and in the latter case we show this for IND security.

[4]For our results it is, in fact, sufficient if the recovery algorithm succeeds only with non-negligible probability.

Theorem 8.2 (Uninstantiability of admissible transforms)**.** *Let* $\mathsf{GT}^{\mathsf{RO}}$ *be an admissible transformation. Assuming the existence of indistinguishability obfuscation for Turing machines* $\mathcal{M}$ *(resp.* p*-bounded circuits* $\mathcal{C}_{\mathsf{p}}$*), the* $\mathsf{GT}^{\mathsf{RO}}$ *transform is uninstantiable (resp.* p*-uninstantiable) with respect to* IND *security and* IND-CPA *base schemes if* GT *is deterministic, and uninstantiable (resp.* p*-uninstantiable) with respect to* IND-CPA *security and* IND-CPA *base schemes if* GT *is randomized.*

Proof. Similarly to EwH, we modify a scheme PKE to a tweaked variant PKE^* by attaching obfuscations of a program that can be used to win the IND game for $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}]$ in case $\mathsf{GT}^{\mathsf{RO}}$ yields a D-PKE scheme, or the IND-CPA game in case it yields a standard PKE scheme. The program that PKE^* obfuscates depends on the recovery algorithm $\mathcal{R}$. Roughly speaking, the program uses the recovery algorithm to recompute the randomness used by the transformation for the underlying PKE scheme and to check its well-formedness. If this check passes the program outputs the original message m. Otherwise it outputs 0. The important difference here is that P on top of a hash-function description also takes a *ciphertext component* as input. In other words, this program allows the adversary to also exploit the ciphertext that it has as its disposal.

$\mathsf{PKE}^*.\mathsf{Enc}(\mathsf{pk}, m; r\|r')$	$\mathsf{P}[\mathsf{pk}, m, s, c](\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, c')$
$s \leftarrow \mathsf{PRG}(r)$	$(m, r\|r') \leftarrow \mathcal{R}^{\mathsf{UP}(\langle \mathsf{H}(\mathsf{hk},\cdot)\rangle,\cdot)}(\mathsf{pk}, m, c, c')$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, m; s)$	$s' \leftarrow \mathsf{PRG}(r)$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[\mathsf{pk}, m, s, c](\cdot,\cdot); r')$	**if** $(s' = s)$ **then return** m
return $(c, \overline{\mathsf{P}})$	**return** 0

IND-CPA preservation. The tweaked scheme remains IND-CPA secure. The proof is analogous to that of Theorem 8.1. First we replace s with a truly random value in the generation of P. This is indistinguishable down to the security of the pseudorandom generator. Next we note that program P would only output a non-zero value in case s lies within the range of the pseudorandom generator. This occurs with only a negligible probability via the union bound. The proof then follows from the security of the indistinguishability obfuscator.

It remains to show that $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ is IND insecure assuming that T is deterministic and in case T is randomized that the instantiated scheme is not IND-CPA secure.

Breaking IND. The attack is similar to that for EwH. We define $\mathcal{A}_1$ to output two random messages that end in 0 and 1 respectively. The second-stage adversary $\mathcal{A}_2$ gets as input $(\mathsf{pk}, (c, \overline{\mathsf{P}}), c')$, that is, the public key pk, the ciphertext part $(c, \overline{\mathsf{P}})$ coming from our adapted scheme PKE^* and the second ciphertext part c' from the transformation GT (see definition of a structured transformation as described at the start of Section 8.3.1). Adversary $\mathcal{A}_2$ constructs a description of the hash function H with the hash key hk hard-coded, and runs $\overline{\mathsf{P}}$ on (H, c'). It terminates by returning the least significant bit of $\overline{\mathsf{P}}$'s output. We give the pseudocode of the attack in Figure 8.4.

When m_b is encrypted, the adversary gets an obfuscation of $\mathsf{P}[\mathsf{pk}', m', s, c]$ where pk' and m' are generated deterministically via T using input $(\mathsf{pk}, m_b; \varepsilon)$. (Note that we are currently considering deterministic transformations.) On input (H, c') program $\mathsf{P}[\mathsf{pk}', m', s, c]$ runs the recovery algorithm $\mathcal{R}$ on input $(\mathsf{pk}', c, c', m')$, giving it oracle access to H. The recovery algorithm outputs $(m_b, r\|r')$, where $r\|r'$ were the coins given to PKE^*. Program P then recomputes $\mathsf{PRG}(r)$. This matches s by

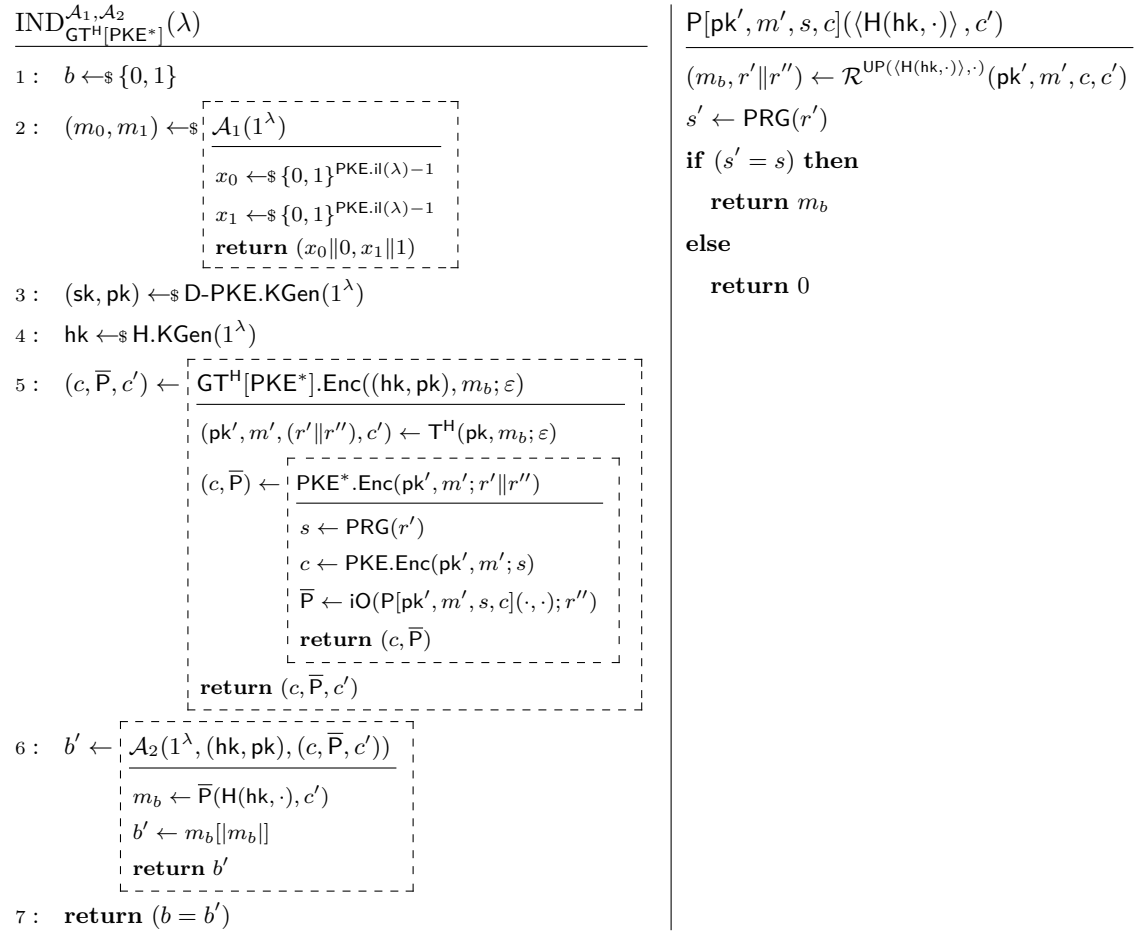


Figure 8.4: The IND game with respect to $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ and adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 8.2. For IND we assume that $\mathsf{GT}^{\mathsf{H}}[\mathsf{PKE}^*]$ is deterministic and thus $r = \varepsilon$.

the definition of GT, and hence the program outputs m_b. Thus, the adversary recovers the least significant bit of the encrypted message with probability 1.

Breaking IND-CPA. In this case, we consider an IND-CPA adversary $\mathcal{A}$ that outputs $m_0 := 0$ and $m_1 := 1$ as its chosen plaintexts and in its second phase launches the second stage of the IND attack above and checks which messages is recovered. The analysis is identical to the above case, noting that the recovery algorithm recovers the encrypted message as well as the random coins given to the PKE encryption operation. □

8.3.2 Transformation for Strong IND

The IND security game for deterministic PKEs only applies to public-key-independent distributions. Raghunathan, Segev and Vadhan [RSV13] strengthen this definition to allow the adversaries to adaptively choose messages after learning some information about the public key. They then give constructions in the standard model and present two new transformations in the random-oracle

model. The first scheme generates an additional random value u as part of the public key and sets the required randomness to $\mathsf{RO}(\mathsf{hk}, m\|u)$; that is, the entire public key is *not* used for randomness generation but only a specific part of it. The second scheme is parameterized by a polynomial Q and generates randomness as $\oplus_{i=1}^{Q+1}\mathsf{RO}(\mathsf{hk}, m\|i)$. Both of these schemes fall prey to our iO-based attacks as they can be shown to be admissible similarly to the EwH transformation.

8.4 The Fujisaki–Okamoto Transformation

The Fujisaki–Okamoto (FO) transformation [FO99] is a ROM technique to convert weak public-key encryption schemes, e.g., those which are indistinguishable (or even one-way) against chosen-plaintext attacks into strong ones which resist chosen-ciphertext attacks (i.e., are IND-CCA secure). In this transform a public-key encryption scheme PKE, a (deterministic) symmetric encryption scheme SE and two independent random oracles RO_1 and RO_2 are used. Under the FO transform, a ciphertext for a message m is generated by picking a fresh random value σ—FO is randomized—which will be hashed and then used as key for the symmetric scheme which in turn is used to encrypt the actual message m. The asymmetric scheme PKE is then used to encrypt σ in a checkable way: the randomness used to encrypt can be derived from message m and value σ. The encryption operation can, thus, be specified as:

$\underline{\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(\mathsf{pk}, m; \sigma)}$

$c_1 \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}, \sigma; \mathsf{RO}_1(\sigma\|m))$

$c_2 \leftarrow \mathsf{SE}.\mathsf{Enc}(\mathsf{RO}_2(\sigma), m)$

return (c_1, c_2)

In the standard model the random oracles are instantiated with keyed hash functions H and G. The hash keys are assumed to be a part of the public key. We denote such a standard-model instantiation by $\mathsf{FO}^{\mathsf{H},\mathsf{G}}[\mathsf{PKE},\mathsf{SE}]$. Similarly to EwH, the FO transformation is admissible and Theorem 8.2 implies its uninstantiability. We show that FO is uninstantiable even with respect to the weaker IND-CPA (rather than IND-CCA) security.

Corollary 8.3 (Uninstantiability of FO)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp. p-bounded circuits $\mathcal{C}_\mathsf{p}$), the FO transform is uninstantiable (resp. p-uninstantiable) with respect to* IND-CPA *security and* IND-CPA *base schemes in the standard model.*

Proof. We show that $\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}]$ is admissible, rewriting the FO transformation in *structured* form and providing a *recovery algorithm* $\mathcal{R}$. The result then follows with Theorem 8.2. The required transformation T and the recovery algorithm $\mathcal{R}$ are shown below.

$\underline{\mathsf{FO}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}].\mathsf{Enc}(\mathsf{pk},m;\sigma)}$

$(\mathsf{pk}',m',r',c') \leftarrow \boxed{\underline{\mathsf{T}^{\mathsf{RO}_1,\mathsf{RO}_2}(pk,m;\sigma)}\ m' \leftarrow \sigma;\ r' \leftarrow \mathsf{RO}_1(\sigma\|m);\ c \leftarrow \mathsf{SE}.\mathsf{Enc}(\mathsf{RO}_2(\sigma),m);\ \textbf{return}\ (pk,m',r',c')}$

$c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk}',m';r')$

return (c,c')

$\underline{\mathcal{R}^{\mathsf{RO}_1,\mathsf{RO}_2}[\mathsf{PKE},\mathsf{SE}](\mathsf{pk}',m',c,c')}$

$\sigma \leftarrow m'$

$m \leftarrow \mathsf{SE}.\mathsf{Dec}(\mathsf{RO}_2(\sigma),c')$

$r' \leftarrow \mathsf{RO}_1(\sigma\|m)$

return (m,r')

□

Partial instantiations of FO. Boldyreva and Fischlin [BF05] study the security of the FO transformation when only *one* of the two random oracles in the construction is instantiated.[5] They consider perfectly one-way hash functions (POW) [Can97], which, on a high-level, hide all information about preimages even given the hash key. Boldyreva and Fischlin [BF05] show that under an assumption which they call *POW encryption* one can securely instantiate the RO_1 oracle. The POW-encryption assumption asks that for any efficient message distribution $\mathcal{M}$ the following two distributions are computationally indistinguishable.

$$\left(\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE}.\mathsf{KGen}(1^\lambda)\\ \mathsf{k} \leftarrow\!\!\$\ \mathsf{POWHF}.\mathsf{KGen}(1^\lambda)\\ r \leftarrow\!\!\$\ \mathsf{POWHF}.\mathsf{coins}(\lambda)\\ (m,\mathsf{aux}) \leftarrow\!\!\$\ \mathcal{M}(\mathsf{pk},\mathsf{k},r)\\ \sigma \leftarrow\!\!\$\ \{0,1\}^\lambda\\ \omega \leftarrow \mathsf{POWHF}.\mathsf{Eval}(\mathsf{k},\sigma\|m;r)\\ c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk},\sigma;\omega)\\ \textbf{return}\ (\mathsf{pk},\mathsf{k},r,c,\mathsf{aux})\end{array}\right) \approx \left(\begin{array}{l}(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE}.\mathsf{KGen}(1^\lambda)\\ \mathsf{k} \leftarrow\!\!\$\ \mathsf{POWHF}.\mathsf{KGen}(1^\lambda)\\ r \leftarrow\!\!\$\ \mathsf{POWHF}.\mathsf{coins}(\lambda)\\ (m,\mathsf{aux}) \leftarrow\!\!\$\ \mathcal{M}(\mathsf{pk},\mathsf{k},r)\\ \sigma \leftarrow\!\!\$\ \{0,1\}^\lambda\\ \omega \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PKE.rl}(\lambda)}\\ c \leftarrow \mathsf{PKE}.\mathsf{Enc}(\mathsf{pk},\sigma;\omega)\\ \textbf{return}\ (\mathsf{pk},\mathsf{k},r,c,\mathsf{aux})\end{array}\right)$$

Looking at the proof of Corollary 8.3, and the obfuscated program P in particular (appearing in Theorem 8.2), we see that P uses both of its random oracles; that is, the recovery algorithm $\mathcal{R}$, which is a subroutine of P, first decrypts a ciphertext component using RO_2 and then recomputes the randomness using RO_1. We can modify this algorithm and obtain an impossibility result for partial instantiations as follows. Instead of using the decryption routine, we hard-wire the message $m_1 := 1$ into the circuit. For this, it is crucial that we operate in the setting of IND-CPA security where the adversary can always submit the same two messages, say $m_0 := 0$ and $m_1 := 1$. (In contrast, for IND security as considered before, the messages were required to be unpredictable.) We use a program similar to that used in the proof of Theorem 8.3 and use the following subroutine $\mathcal{R}$ as follows.

$\underline{\mathcal{R}^{\mathsf{RO}_1}[\mathsf{PKE},\mathsf{SE},m_1](\mathsf{pk}',m',c,c')}$

$\sigma \leftarrow m'$

$r' \leftarrow \mathsf{RO}_1(\sigma\|m_1)$

return (m_1,r')

[5] Note that the security analysis is still in the random-oracle model, as only one of the random oracles is instantiated.

The second phase of IND-CPA adversary $\mathcal{A}$, as before, runs $\overline{\mathsf{P}}$ and outputs (the last bit of) the result. Through these modifications we have removed the dependency on the second random oracle altogether. We can restate our result as follows.

Corollary 8.4 (Partial uninstantiability of FO)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp. p-bounded circuits $\mathcal{C}_\mathsf{p}$), the first random oracle in the* FO *transformation is uninstantiable (resp. p-uninstantiable) with respect to* IND-CPA *security and* IND-CPA *base schemes in the standard model. In particular the POW-encryption assumption is uninstantiable (resp. p-uninstantiable) with respect to* IND-CPA *schemes in the standard model assuming iO for Turing machines (resp. for $\mathcal{C}_\mathsf{p}$).*

In other words, while the POW-encryption assumption may hold for perfectly one-way hash functions in the random oracle model it cannot hold for any standard-model perfectly one-way hash function if indistinguishability obfuscation exists.

8.5 Careful with Conversion

In this section we explore new classes of D-PKE transformations that lie beyond those captured by admissible transformations. We present a candidate transformation that is specifically designed to foil our iO attack. We first show that this transformation is structurally sound by proving it secure in the ROM. We then show how to extend our techniques to this (and potentially other classes of) transformations. Our goal is to illustrate the flexibility of our main technique and show that it can be tweaked and extended in many ways.

8.5.1 Hybrid and Double Encrypt-with-Hash

The underlying idea behind this new transformation, which we term *Hybrid and Double Encrypt-with-Hash* (HD-EwH), is to fix the symmetric encryption scheme to a one-time pad (so that it cannot be modified adversarially) and "share out" the randomness and message given to the public-key scheme among two independent invocations so that the necessary information needed for an iO attack is not available for any single invocation. Formally, we define $\mathsf{HD\text{-}EwH}^{\mathsf{RO}}[\mathsf{PKE}]$ as follows. Key generation creates $(\mathsf{sk},\mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$ as well as four hash keys $\mathsf{hk}_1,\mathsf{hk}_2,\mathsf{gk}_1,\mathsf{gk}_2 \leftarrow\!\!\$\ \{0,1\}^{\mathsf{kl}(\lambda)}$. It returns $(\mathsf{sk},(\mathsf{hk}_1,\mathsf{hk}_2,\mathsf{gk}_1,\mathsf{gk}_2,\mathsf{pk}))$. An encryption of a message m consists of the following three components

$$\mathsf{PKE.Enc}\left(\mathsf{pk},\mathsf{RO}(\mathsf{hk}_1,\mathsf{pk}\|m);\mathsf{RO}(\mathsf{gk}_1,\mathsf{pk}\|m)\right),\mathsf{PKE.Enc}\left(\mathsf{pk},\mathsf{RO}(\mathsf{hk}_2,\mathsf{pk}\|m);\mathsf{RO}(\mathsf{gk}_2,\mathsf{pk}\|m)\right),$$
$$m\oplus\mathsf{RO}^2(\mathsf{hk}_1,\mathsf{pk}\|m)\oplus\mathsf{RO}^2(\mathsf{hk}_2,\mathsf{pk}\|m)\ ,$$

where $\mathsf{RO}^2(\mathsf{hk},x) := \mathsf{RO}(\mathsf{hk},\mathsf{RO}(\mathsf{hk},x))$. The decryption algorithm decrypts the asymmetric components of the ciphertext to get $\mathsf{RO}(\mathsf{hk}_1,\mathsf{pk}\|m)$ and $\mathsf{RO}(\mathsf{hk}_2,\mathsf{pk}\|m)$, hashes them under keys hk_1 and hk_2 respectively and xors them to compute the symmetric mask, and uses this to recover the message.

We next establish the soundness of the above transform by showing that it indeed results in a secure D-PKE in the random-oracle model. We postpone the formal proof until Section 9.4. This

allows us to use a tool called universal computational extractors which overall yields a cleaner proof than a direct proof in the random oracle model. (A direct proof of this result appears in [BFM14b].)

Proposition 8.5 (ROM security of HD-EwH)**.** *Let* PKE *be an* IND-CPA*-secure public-key encryption scheme. Then, in the random-oracle model and assuming that the probability of correctly guessing a public-key as generated by* PKE.KGen *on uniformly random coins is negligible, we have that scheme* $\mathsf{HD\text{-}EwH}^{\mathsf{RO}_1,\ldots,\mathsf{RO}_4}[\mathsf{PKE}]$ *is an* IND*-secure D-PKE scheme.*

Remark. We note that we, in fact, show a stronger result, that is, we only require the PKE scheme to be one-way. For this note that the PKE scheme is only used to encrypt randomly chosen messages (i.e., results from random oracle queries).

Uninstantiability of Hybrid and Double Encrypt-with-Hash

It is easy to see that this transformation falls outside the realm of our generalized result in Section 8.3, and this opens up the possibility of its standard-model instantiability. We show that our techniques can be extended to also cover HD-EwH. We will prove this for a slight generalization of HD-EwH where a fifth random oracle is used to generate a one-time-pad key for the ciphertext component:

$$m \oplus \mathsf{RO}\left(\mathsf{fk}, (\mathsf{RO}(\mathsf{hk}_1, \mathsf{pk}\|m), \mathsf{RO}(\mathsf{hk}_2, \mathsf{pk}\|m))\right) .$$

If the original scheme is instantiable, then so is this scheme: we first instantiate the first four oracles, and then replace the fifth one by the hash function $\mathsf{F.Eval}(\mathsf{fk}, (x_1, x_2)) := \mathsf{H}_1.\mathsf{Eval}(\mathsf{hk}_1, x_1) \oplus \mathsf{H}_2.\mathsf{Eval}(\mathsf{hk}_2, x_2)$, where $\mathsf{fk} := (\mathsf{hk}_1, \mathsf{hk}_2)$.

As before, we construct an adversarial PKE scheme PKE^* which outputs obfuscations as part of its ciphertext. In this case, however, the scheme will output the obfuscations of two programs P1 and P2 as shown below. We denote the messages passed to the two instances of the public-key encryption scheme by x and x', reserve m for the actual message encrypted under $\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2,\mathsf{F}}[\mathsf{PKE}^*]$.

$\underline{\mathsf{PKE}^*.\mathsf{Enc}(x, \mathsf{pk}; r\|r_1\|r_2)}$	$\underline{\mathsf{P1}[\mathsf{pk}, x, s](\mathsf{G}_1, \mathsf{G}_2, \mathsf{F}, c, \overline{\mathsf{P2}})}$	$\underline{\mathsf{P2}[\mathsf{pk}, x, s](\mathsf{G}_2, \mathsf{F}, x', c)}$
$s \leftarrow \mathsf{PRG}(r)$	$m \leftarrow \mathsf{UP}(\overline{\mathsf{P2}}, (\mathsf{G}_2, \mathsf{F}, x, c))$	$m \leftarrow c \oplus \mathsf{UP}(\mathsf{F}, (x, x'))$
$c \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x; s)$	$r\|r_1\|r_2 \leftarrow \mathsf{G}_1(\mathsf{pk}\|m)$	$r\|r_1\|r_2 \leftarrow \mathsf{G}_2(\mathsf{pk}\|m)$
$\overline{\mathsf{P1}} \leftarrow \mathsf{iO}(\mathsf{P1}[\mathsf{pk}, x, s](\cdot); r_1)$	**if** $(\mathsf{PRG}(r) = s)$ **then**	**if** $(\mathsf{PRG}(r) = s)$ **then**
$\overline{\mathsf{P2}} \leftarrow \mathsf{iO}(\mathsf{P2}[\mathsf{pk}, x, s](\cdot); r_2)$	**return** m	**return** m
return $(c, \overline{\mathsf{P1}}, \overline{\mathsf{P2}})$	**return** 0	**return** 0

The proof that PKE^* is IND-CPA secure is analogous to that of Theorem 8.1. We rely on the indistinguishability security of the obfuscator and the security of the pseudorandom generator to show that the obfuscations of the above programs are indistinguishable from those of the zero program. We first replace s with a truly random string. This change affects any adversary's advantage with only a negligible probability down to the security of PRG. Now unless s happens to be in the range of PRG, an unlikely event, both programs P1 and P2 implement the zero program. Hence we can replace the obfuscation of P1 and P2 by those of the (appropriately padded) zero program.

We now show that using scheme PKE^* in the HD-EwH transform yields an insecure scheme for any choice of hash functions H_1, H_2, G_1, G_2 and F for the five random oracles. Let us see how the adversarial scheme looks likes when plugged into HD-EwH with these hash functions.

$\underline{\mathsf{HD\text{-}EwH}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{G}_1,\mathsf{G}_2,\mathsf{F}}[\mathsf{PKE}^*](\mathsf{pk}, m)}$

1 : $r_1||r_1'||r_1'' \leftarrow \mathsf{G}_1.\mathsf{Eval}(\mathsf{gk}_1, \mathsf{pk}\|m)$;
2 : $r_2||r_2'||r_2'' \leftarrow \mathsf{G}_2.\mathsf{Eval}(\mathsf{gk}_2, \mathsf{pk}\|m)$
3 : $s_1 \leftarrow \mathsf{PRG}(r_1)$;
4 : $s_2 \leftarrow \mathsf{PRG}(r_2)$
5 : $x_1 \leftarrow \mathsf{H}_1.\mathsf{Eval}(\mathsf{hk}_1, \mathsf{pk}\|m)$;
6 : $x_2 \leftarrow \mathsf{H}_2.\mathsf{Eval}(\mathsf{hk}_2, \mathsf{pk}\|m)$
7 : $c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_1; s_1)$;
8 : $c_1' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, x_2; s_2)$
9 : $\overline{\mathsf{P1}} \leftarrow \mathsf{iO}(\mathsf{P1}[\mathsf{pk}, x_1, s_1](\cdot); r_1')$;
10 : $\overline{\mathsf{P1}}' \leftarrow \mathsf{iO}(\mathsf{P1}[\mathsf{pk}, x_2, s_2](\cdot); r_2')$
11 : $\overline{\mathsf{P2}} \leftarrow \mathsf{iO}(\mathsf{P2}[\mathsf{pk}, x_1, s_1](\cdot); r_1'')$;
12 : $\overline{\mathsf{P2}}' \leftarrow \mathsf{iO}(\mathsf{P2}[\mathsf{pk}, x_2, s_2](\cdot); r_2'')$
13 : $c \leftarrow m \oplus \mathsf{F.Eval}(\mathsf{fk}, x_1, x_2)$
14 : **return** $(c_1, \overline{\mathsf{P1}}, \overline{\mathsf{P2}}, c_1', \overline{\mathsf{P1}}', \overline{\mathsf{P2}}', c)$

We construct an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the IND security of our transformed scheme as follows. The first adversary $\mathcal{A}_1$ chooses two uniformly random values $d_0, d_1 \leftarrow\!\!\$\, \{0,1\}^{\mathsf{PKE.il}(\lambda)-1}$ and outputs messages $m_0 := d_0\|0$ and $m_1 := d_1\|1$. The second adversary $\mathcal{A}_2$ then receives as input a ciphertext $(c_1, \overline{\mathsf{P1}}, \overline{\mathsf{P2}}, c_1', \overline{\mathsf{P1}}', \overline{\mathsf{P2}}', c)$, where components $\overline{\mathsf{P1}}$ and $\overline{\mathsf{P1}}'$ are obfuscations of $\mathsf{P1}[\mathsf{pk}, x_1, s_1]$ and $\mathsf{P1}'[\mathsf{pk}, x_2, s_2]$ respectively, and $\overline{\mathsf{P2}}$ and $\overline{\mathsf{P2}}'$ are obfuscations of $\mathsf{P2}[\mathsf{pk}, x_1, s_1]$ and $\mathsf{P2}'[\mathsf{pk}, x_2, s_2]$ respectively. Adversary $\mathcal{A}_2$ then runs $\overline{\mathsf{P1}}[\mathsf{pk}, x_1, s_1]$ on input the descriptions of functions $\mathsf{G}_1(\mathsf{gk}_1, \cdot)$ and $\mathsf{G}_2(\mathsf{gk}_2, \cdot)$, a description of function $\mathsf{F}(\mathsf{fk}, \cdot)$, the ciphertext component c and the obfuscated program $\overline{\mathsf{P2}}'[\mathsf{pk}, x, s_2](\cdot)$. Note that the attack is running an obfuscated circuit on another obfuscated circuit.[6] It returns the least significant bit of the output as its guess.

To see that this attack is successful, observe that the program consisting of the composition of $\mathsf{P1}$ with $\mathsf{P2}$ as run by the adversary is, with overwhelming probability, functionally equivalent to program P^* below.

$\underline{\mathsf{P}^*[\mathsf{pk}, x_1, s_1, x_2, s_2](\mathsf{G}_1, \mathsf{G}_2, \mathsf{F}, c)}$

$m \leftarrow c \oplus \mathsf{UP}(\mathsf{F}, (x_1, x_2))$
$r_1||r_1'||r_1'' \leftarrow \mathsf{G}_1(\mathsf{pk}\|m)$
$r_2||r_2'||r_2'' \leftarrow \mathsf{G}_2(\mathsf{pk}\|m)$
if $(\mathsf{PRG}(r_1) \neq s_1)$ **then return** 0
if $(\mathsf{PRG}(r_2) \neq s_2)$ **then return** 0
return m

This program can be seen as the analogue of that presented for EwH adapted to the HD-EwH transform. Indeed, had we access to both (x_1, s_1) and (x_2, s_2) in one of the runs of the encryption algorithm, we could have directly attacked the scheme by obfuscating P^*. Since this access is (by design) denied to the scheme, we instead emulate the effect of the above program by constructing two obfuscated programs, each having access to only one of (x_1, s_1) or (x_2, s_2). As before, the above program returns the message m when run on correct hash descriptions and the last component of the ciphertext. Hence, by our choice of challenge messages, returning the least significant bit of the output message would match the hidden bit with probability one.

[6] Alternatively, it can also run $\overline{\mathsf{P1}}'[\mathsf{pk}, x_2, s_2]$ on the obfuscated program $\overline{\mathsf{P2}}[\mathsf{pk}, x_1, s_2]$ and hash descriptions. In either case, the modified PKE scheme must contain obfuscations of both P1 and P2.

$\underline{\text{PRV-CDA}^{\mathcal{A}_1,\mathcal{A}_2}_{\text{MLE}}(\lambda)}$

$\mathsf{pp} \leftarrow\!\!\$\ \text{MLE.}\mathsf{Pgen}(\lambda)$
$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1 \ldots |\mathbf{m}_0|$ **do**
  $\mathsf{k} \leftarrow \text{MLE.}\mathsf{KGen}(\mathsf{pp}, \mathbf{m}_b[i])$
  $\mathbf{c}[i] \leftarrow \text{MLE.}\mathsf{Enc}(\mathsf{k}, \mathbf{m}_b[i])$
$b' \leftarrow\!\!\$\ \mathcal{A}_2(\mathsf{pp}, \mathbf{c})$
return $(b = b')$

Figure 8.5: The MLE security games PRV-CDA of BKR [BKR13]. Here we have given slightly simpler variant where the adversaries are not allowed to share state via a zeroth-stage adversary. We note that this only strengthens our negative results.

8.6 ROM Transformations for Symmetric Encryption

Uninstantiability problems arising from the existence of indistinguishability obfuscators are not limited to public-key encryption schemes but can be similarly applied to random oracle transformations that transform (weak) symmetric encryption schemes into strong symmetric encryption schemes. As example, we consider the convergent encryption transformation for *message-locked security*, and discuss the intuition on how to adapt our techniques to this scenarios. For a more detailed description we refer to [BFM14b] where we show that also the *Randomized-Hash-then-Encrypt* introduced by Bellare and Keelveedhi (BK) [BK11] for obtaining key-dependent message security may be subject to uninstantiability.

8.6.1 Message-Locked Encryption

Message-locked encryption (MLE) is a form of deterministic symmetric encryption where the encryption key is deterministically derived from the message that is to be encrypted. This mechanism ensures that encryptions of identical plaintexts produce identical ciphertexts, allowing (cloud) storage providers to keep a single copy of the encrypted data (this is also referred to as *secure deduplication*). MLE was first formalized by Bellare, Keelveedhi and Ristenpart (BKR) [BKR13], who defined appropriate security models and constructed schemes that meet these definitions in the random-oracle and standard model.

BKR propose several security notions for MLEs. One is called PRV-CDA and is similar to the IND notion for D-PKE. In place of the public key, the public parameters of the scheme pp are now outside the reach of the first phase of the attack. These parameters are used by the encryption operation to derive the encryption key. In order to rule out trivial re-encryption attacks, each component of the message vectors are required to have high min-entropy (similarly, as in the case of IND). See Figure 8.5 for the details of the game.

Convergent encryption. One transformation which is formally studied by BKR and originates in the work of Douceur et al. [DAB+02] is *convergent encryption* (CE). The CE transform constructs a message-locked encryption scheme from a one-time-secure deterministic symmetric encryption

scheme SE in the (keyed) random-oracle model. Parameters pp are chosen during setup as a uniformly random string. The encryption key k for a message m and with public parameters pp and hash key hk is computed as $\mathsf{k} \leftarrow \mathsf{RO}(\mathsf{hk}, \mathsf{pp}\|m)$. Note that the range of the hash function must be a subset of the scheme's key space. The encryption and decryption algorithms of SE are used directly without change in the new scheme. We next present the pseudocode of the resulting scheme in the keyed random oracle model.

$\underline{\mathsf{CE}^{\mathsf{RO}}[\mathsf{SE}].\mathsf{Enc}(\mathsf{hk}, \mathsf{pp}, m)}$

$\mathsf{k} \leftarrow \mathsf{RO}(\mathsf{hk}, \mathsf{pp}\|m)$

$c \leftarrow \mathsf{SE}.\mathsf{Enc}(\mathsf{k}, m)$

return c

$\underline{\mathsf{CE}^{\mathsf{RO}}[\mathsf{SE}].\mathsf{Dec}(\mathsf{k}, c)}$

$m \leftarrow \mathsf{SE}.\mathsf{Dec}(\mathsf{k}, c)$

Under one-time key recovery[7] and a stronger variant of one-time IND-CPA security which requires ciphertexts are indistinguishable from random strings, the CE transformation is proved PRV-CDA secure in the random-oracle model [BKR13]. We note that the stronger requirement on IND-CPA is necessary since Bellare et al. show a stronger result, that is, they show that the resulting scheme is PRV$-CDA secure which strengthens PRV-CDA by requiring that an adversary cannot distinguish between ciphertexts or random strings.

Uninstantiability of Convergent Encryption

We show that the above random oracle transformation is not sound assuming that indistinguishability obfuscation exist. For this we present an IND-CPA secure symmetric encryption scheme which when transformed via the above transformation yields an insecure scheme. Jumping ahead, we note that if we fix the symmetric encryption scheme to implement a one-time pad then we can actually instantiate the random oracle via a hash function constructed from indistinguishability obfuscation. This positive result follows from our instantiations of UCEs in Chapter 11 and results obtained by Bellare et al. in [BHK13:2].

Attacking PRV-CDA**.** We now show that this transform yields an insecure scheme when starting from an adversarial one-time key-recovery and one-time IND-CPA-secure scheme SE^*. Our generalized result presented in Section 8.3 does not apply here as we are considering a transformation of (deterministic) symmetric encryption scheme.

Theorem 8.6 (Uninstantiability of convergent encryption)**.** *Assuming the existence of indistinguishability obfuscation for Turing machines $\mathcal{M}$ (resp. p-bounded circuits $\mathcal{C}_\mathsf{p}$) and a pseudorandom generator the convergent encryption transform CE is uninstantiable (resp. p-uninstantiable) with respect to* PRV-CDA *security and one-time key-recovery and one-time* IND-CPA*-secure base schemes in the standard model.*

We next present our one-time IND-CPA-secure symmetric encryption SE^* that breaks CE^{H}. The idea, as before, is to append an obfuscated circuited to the ciphertext. Since in MLE the scheme is not randomized, we obtain the necessary randomness for obfuscation directly from the secret key.

[7] One-time key recovery requires that in presence of at most one ciphertext no adversary can guess the key with non-negligible probability. The reason that it suffices to consider one-time security is that for each encryption a fresh key is chosen.

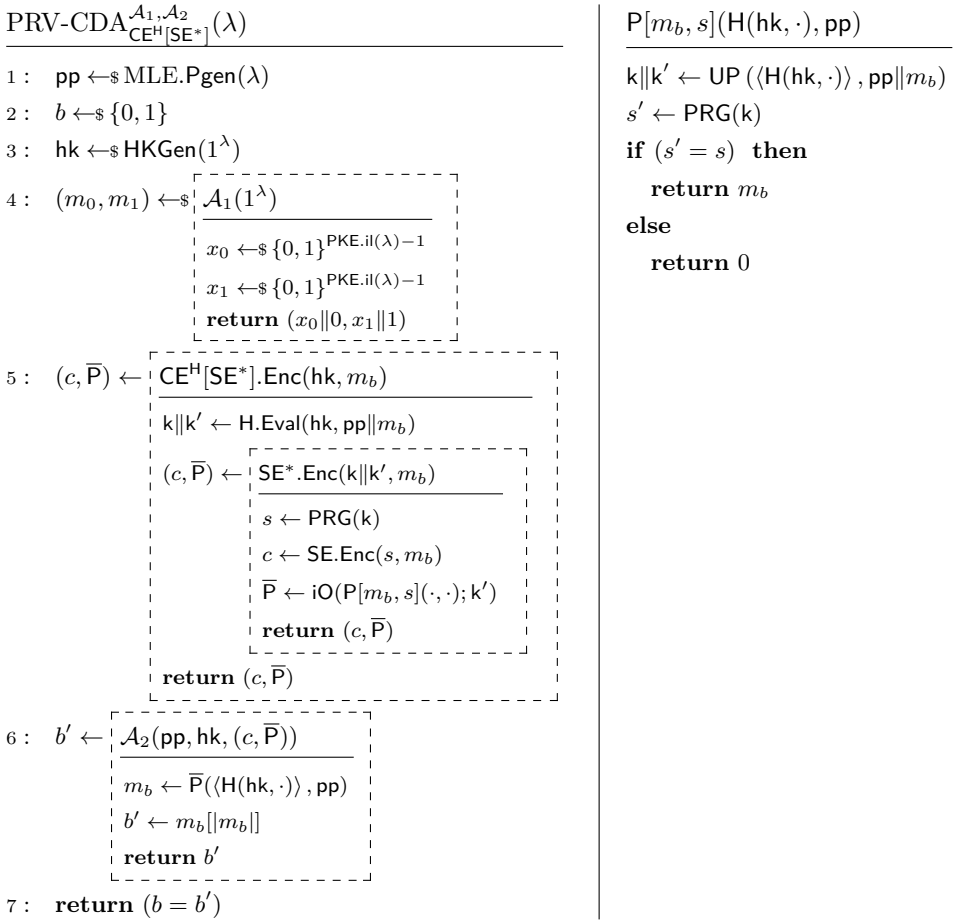


Figure 8.6: The PRV-CDA security game for scheme $\mathsf{CE}^{\mathsf{H}}[\mathsf{SE}^*]$ with our adversary $(\mathcal{A}_1, \mathcal{A}_2)$ as constructed in the proof of Theorem 8.6. The boxed algorithms are to be understood as subroutines. Program P that is obfuscated as part of ciphertexts is given on the right.

Given a one-time IND-CPA-secure and one-time key-recovery-secure scheme SE with uniform keys in $\{0,1\}^{\mathsf{SE.kl}(\lambda)}$ and a pseudorandom generator PRG of appropriate stretch we construct SE^* as follows. Key generation is left unchanged, and encryption is shown below. Decryption simply ignores the second component of the ciphertext and decrypts the first.

$\mathsf{SE}^*.\mathsf{Enc}(\mathsf{k}\|\mathsf{k}', m)$	$\mathsf{P}[m, s](\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \mathsf{pp})$
$s \leftarrow \mathsf{PRG}(\mathsf{k})$	$\mathsf{k}\|\mathsf{k}' \leftarrow \mathsf{UP}(\langle \mathsf{H}(\mathsf{hk}, \cdot)\rangle, \mathsf{pp}\|m)$
$c \leftarrow \mathsf{SE.Enc}(s, m)$	$s' \leftarrow \mathsf{PRG}(\mathsf{k})$
$\overline{\mathsf{P}} \leftarrow \mathsf{iO}(\mathsf{P}[m, s](\cdot,\cdot); \mathsf{k}')$	**if** $(s' = s)$ **then return** m
return $(c, \overline{\mathsf{P}})$	**return** 0

Proving that the above modifications do not affect the one-time IND-CPA and one-time key-recovery security properties of SE^* is analogous to that presented for D-PKEs. Since we consider one-time security, each encryption is run on a freshly sampled random key which can take the role of randomness used in the proof for D-PKEs. The attack against the PRV-CDA security of the

transformed scheme also works analogously to the EwH case. There is, however, a minor modification that needs to be taken care of as the symmetric component of the scheme does not have access to the public parameters pp of the MLE scheme (and, in particular, we cannot hardcode them into the obfuscated circuit). We address this issue by considering a circuit which takes the public parameters pp as an additional input. Note that according to the rules of the PRV-CDA game, pp will be available to the second-stage adversary. We present the pseudocode outline of the attack in Figure 8.6.

8.7 Circumventing Uninstantiability

We close our discussion of uninstantiability of random oracle transformations by briefly discussing potential ways around uninstantiability issues.

In the adapted scheme SE^* from the previous section we somewhat abused the fact that the base scheme only needs to be one-time secure, and derived arbitrary randomness from the key. It is an interesting question whether our attack can also be mounted if the base scheme is required to meet the standard IND-CPA notion. Removing the one-time restriction is intricate as one would have to introduce fresh randomness to avoid trivial attacks.

A second avenue that might allow to circumvent uninstantiability is to only allow short ciphertexts (such that an obfuscation cannot be embedded in the ciphertext) or to demand the stronger requirement that ciphertexts look random. This might not only be a way to circumvent uninstantiability results for random oracle transformations in the symmetric setting but might also allow to work around our uninstantiability results for public-key encryption schemes. For this note that our uninstantiability result crucially depends on including the obfuscation of a circuit into the ciphertext. Such an obfuscation is, however, heavily structured and it is not clear if indistinguishability obfuscation schemes exist that have an obfuscation which is indistinguishable from a uniformly random bit string. One straightforward distinguishing attack against any obfuscation scheme would be to simply execute the code and check the outputs. In particular, it cannot be the case that an obfuscation of both the zero circuit and the one circuit look like random strings. However, there could still exist an indistinguishability obfuscation scheme with the extra property that the obfuscations of the zero circuit look random. Although it sounds unlikely that such obfuscation schemes exist it seems an intriguing problem to formalize this intuition. We leave the study of such real-or-random indistinguishability obfuscators for future work.

PART III

UNIVERSAL COMPUTATIONAL EXTRACTORS

Part Summary

In the final part of this thesis we study *Universal Computational Extractors* (UCEs). The UCE framework was proposed by Bellare, Hoang, and Keelveedhi [BHK13:p] to provide a means to construct strong standard model assumptions for keyed hash functions which can be used to instantiate random oracles in a wide variety of applications. We show both positive and negative results, that is, we show that if indistinguishability obfuscation exists then some forms of strong UCEs cannot be constructed in the standard model. On the other hand, we construct several forms of interesting UCE notions from indistinguishability obfuscation and point obfuscation leading to the first standard model constructions for a large number of cryptographic primitives which before only had constructions in idealized models. This includes constructions of universal hardcore functions, q-query deterministic public-key encryption schemes and q-query correlated-input secure hash functions.

An artifact of constructions based on indistinguishability obfuscation is the use of padding which seems required within proofs but leads to our constructions only achieving q-query security. We initiate a formal study of padding in obfuscation based constructions and present a novel and strong assumption which may allow to lift the security of q-query construction to full security.

Chapter Summary

In Chapter 9 we introduce the UCE framework and discuss various proposed UCE notions. We show that the originally proposed UCE notions cannot be achieved in the standard model assuming that indistinguishability obfuscation exists. The results in this chapter are based on [BFM14a, BFM15, Mit14, BM14c, BM15b] as well as on [BHK13:p, BHK13:2] in which the UCE framework was introduced by Bellare, Hoang, and Keelveedhi.

In Chapter 10 we show that our negative results for strong forms of UCEs also carry over to more restricted forms of UCEs which are based on computationally unpredictable sources. The results in this chapter are based mainly on [BFM14a].

In Chapter 11 we show how to construct several different forms of UCE secure hash functions from indistinguishability obfuscation, puncturable pseudorandom functions and strong forms of point obfuscation. These constructions give rise to the first standard model constructions of various interesting cryptographic primitives such as universal hardcore functions, q-query deterministic public-key encryption and q-query correlated-input secure hash functions. The results in this chapter are based mainly on [BM14c, BM15b].

In Chapter 12 we study the relationship between UCEs and point-function obfuscation and show that the latter can be, indeed, constructed from UCEs. This shows that the assumptions on the existence of strong point-function obfuscation for our constructions of UCEs are, in fact, necessary. The results in this chapter are based mainly on [BM15b].

In Chapter 13 we initiate a formal study of padding in constructions using indistinguishability obfuscation. We present a framework of assumptions for stating that padding is only an artifact of proof techniques and study how reasonable such assumptions are. We show that even very restricted assumptions can be shown not to hold but that there is still some hope for showing that the q-query bound of our UCE constructions can be lifted. The results in this chapter are based mainly on [BM15b]. The counter-examples are due to Bitansky et al. [BCC$^+$14] and Holmgren [Hol15].

In Chapter 14 we show how to construct q-query deterministic public-key encryption schemes directly from indistinguishabiilty obfuscation and point-function obfuscation. The advantages of a direct construction are somewhat simpler assumptions and the possibility of applying much more restricted versions of the SuP assumption (Chapter 13) to lift the q-query bound. The results in this chapter are based mainly on an unpublished manuscript [BM15a].

Chapter 9

Universal Computational Extractors

> "The lack of a proof of security for the instantiated scheme is [...] a consequence of an even more fundamental lack, namely that of a definition, of what it means for a family of functions to 'behave like a RO'."
>
> Bellare, Hoang, and Keelveedhi [BHK13:p]

Summary. In this chapter we introduce *Universal Computational Extractors* (UCE) a framework developed by Bellare, Hoang, and Keelveedhi [BHK13:p] to capture strong security of keyed hash functions. We present an impossibility result showing that some forms of UCEs cannot be constructed in the standard model if indistinguishability obfuscators exist and present new UCE notions that were designed to work around this impossibility result. The results in this chapter are based on [BFM14a, BM14c, Mit14, BFM15, BM15b]. The UCE framework itself was proposed by Bellare, Hoang, and Keelveedhi in [BHK13:p] and later updated in [BHK13:f, BHK13:1, BHK13:2].

Chapter content

9.1 Introduction

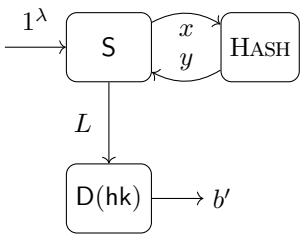
In Chapter 4 we discussed a variety of primitives and accompanying random oracle schemes and saw that schemes that are designed according to the random oracle methodology are often much simpler and more efficient than their standard model counterparts; that is, if a standard model candidate construction exists which often is not the case. For instance, for correlated-input security or for deterministic public-key encryption we do not know of standard-model constructions that meet the security definitions that we would like to achieve while in the random oracle model simple and efficient schemes exist. The enormous power of random oracles may, however, be deceiving and what may be worse, we do not know if and when this is the case. In the previous part, we presented constructions in the random oracle model that are uninstantiable and we have even seen a security

definition (MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$], cf. Chapter 7) which can easily be met in the random oracle model but, assuming indistinguishability obfuscation exists, cannot be constructed without random oracles.

In order to obtain schemes with provable security guarantees we could choose to ignore random oracles and try to construct schemes without the use of such idealized objects. As of July 2015, a Google Scholar search shows 336 papers that have the phrase "without random oracles" in the title. In many cases these schemes are less efficient and much more complex than their random oracle counterparts and in some cases it can also be argued that the resulting schemes are "less secure" [KM15]. (Less secure can, of course, only hold in case the random oracle scheme is secure, which we usually do not know, or at least, cannot prove.) Thus, ideally, we would like to stick to the random oracle methodology when designing schemes (as this often results in very clean and effective designs) but then be able to identify the properties used from the random oracle and choose an appropriate standard model hash function which *provably* achieves these properties. Naturally, this approach cannot be universal as from our uninstantiability results we know that there are properties which simply cannot be met in the standard setting. The hope, however, is that we can identify properties which suffice to instantiate random oracles for a large class of applications.

The search for *standard model definitions* for properties of random oracles was started by Canetti in 1997 with his seminal paper [Can97] in which he introduced a primitive called *oracle hashing* and, en passant, gave the first construction of a point-function obfuscator. Similarly to random oracles, oracle hashing schemes, which were later renamed to *perfectly (probabilistic) one-way hash functions* (POW) [CMR98], are supposed to "hide all partial information on their input". That is, a value $\mathsf{POW}(x)$ should not decrease any uncertainty about value x. A second property of random oracles, identified by Boldyreva et al. [BCFW09], is that random oracles are non-malleable: given a value $\mathsf{RO}(m)$ for some message m it is hard to find a value $\mathsf{RO}(m^*)$ for a related message m^*, for any reasonable notion of *related*. Boldyreva et al. introduce non-malleability for standard-model hash functions and show that it is necessary for an instantiation of one of the random oracles in an extension of the BR93 PKE scheme by Bellare and Rogaway [BR93] (see Section 4.3.3) which yields CCA-secure PKE encryption from any trapdoor function f in the random oracle model. For this a message m is encrypted as $(f(r), \mathsf{G}(r) \oplus m, \mathsf{H}(r\|m))$. Here r is the randomness and G and H are modeled as two random oracles. Similarly, Boldyreva and Fischlin [BF05] show that POWs can instantiate one of the random oracles in a variant of the PSS-E encryption scheme [CJNP02] and, under an extra assumption, also suffice for instantiating one of the random oracles in the Fujisaki–Okamoto transform [FO99]. As we have seen in the previous chapter the necessary extra assumption cannot hold in case indistinguishability obfuscation exists as we have seen that no standard model hash function can securely instantiate Fujisaki–Okamoto (see Section 8.4) and hence also POWs would not suffice. In hindsight, these attempts of modeling random oracle properties have had limited impact on tempting people to not use random-oracles but work in the standard model directly.

This picture may change with a recent work of Bellare, Hoang, and Keelveedhi [BHK13:p] who in 2013, twenty years after the formal introduction of the random oracle methodology, introduced the idea of *universal computational extractors*, UCEs for short. Universal computational extractors are defined as a framework to model strong properties of random oracles in the standard model with the explicit goal to be widely applicable. That is, UCEs come with the promise to instantiate random oracles in a wide range of applications and to, thus, also provide an insight into the capabilities that we require from random oracles in these constructions.



MAIN $\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)$

$b \leftarrow\$ \{0,1\}$
$\mathsf{hk} \leftarrow\$ \mathsf{H.KGen}(1^\lambda)$
$L \leftarrow\$ \mathsf{S}^{\text{HASH}}(1^\lambda)$
$b' \leftarrow\$ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$

HASH(x)

if $T[x] = \perp$ **then**
  if $b = 1$ **then**
    $T[x] \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$
  else $T[x] \leftarrow\$ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$

Figure 9.1: The schematic overview of the UCE game on the left and the corresponding pseudocode on the right.

9.1.1 Universal Computational Extractors

Universal computational extractors (UCEs) provide security definitions for keyed hash functions, that is, we consider a (deterministic) function H that takes two inputs, a key hk and a message m to produce an output y. UCEs are modeled along the idea that the output of a UCE-secure hash function should be indistinguishable from a random string. This should hold in the presence of hash key hk and, furthermore, even in the presence of leakage on the input as long as the leakage is sufficiently restricted. Originally, when UCEs were first presented by Bellare et al. [BHK13:p], sufficiently restricted meant that the leakage computationally hides the inputs. We note that this computational hiding restriction is identical to the computational hiding restriction for AIPOs and MB-AIPOs discussed in Chapters 6 and 7.[1]

More formally, UCEs are PRF-like assumptions where the PRF adversary is split into two parts: We consider a two-stage game $\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}$ where the first stage adversary called the source S interacts with an oracle HASH which either implements a standard-model keyed hash function H with a key hk chosen uniformly at random or which implements a random oracle. Source S can compute some leakage L from the interaction which then together with hash key hk (which is hidden from the source) is given to the second stage adversary called the distinguisher D. In case oracle HASH implements hash function H, the key hk given to D is the key used in the first stage. On the other hand, if HASH implements a random oracle then D is given a key sampled uniformly at random and, in particular, sampled independently of the first stage. The aim of distinguisher D is to distinguish whether source S was interacting with a random oracle, or with the standard-model hash function. We present the UCE game and schematic of the communication in Figure 9.1. We call a keyed standard-model hash function H UCE-secure if no pair of efficient adversaries (S, D) exists that wins in game $\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}$ with probability significantly better than guessing.

If no restriction is put on the leakage L, then the notion models a form of indistinguishability from a random oracle which is unachievable. For this, consider a source S that chooses a random input x, queries its oracle to obtain $y \leftarrow \text{HASH}(x)$ and then leaks $L \leftarrow (x, y)$. Distinguisher D, which additionally gets hash key hk as input can recompute $\mathsf{H}(\mathsf{hk}, x)$ and check if the result is identical to y. If so, then with overwhelming probability source S was interacting with the actual hash function.[2] One, thus, obtains specific UCE assumptions by restricting source (and possibly the distinguisher).

[1] We note that our notation for AIPO sampler classes, for example, the class $\mathcal{S}^{\text{cup}}_{\text{po}}$ is based on notation developed for the UCE framework by Bellare et al. in later revisions [BHK13:1].

[2] Even in case the output length of hash function H is just a single bit, distinguisher D still has (much) better than guessing success probability which can be further increased by considering multiple pairs (x_i, y_i).

A function H is called UCE secure with respect to the restriction if no adversary (S, D) adhering to the restriction exists that wins in the UCE game with better than guessing probability.

The UCE framework is very flexible and allows to model (weak) properties such as *pseudorandomness*—restrict source S to output a single bit and restrict distinguisher D to echo the single bit—as well as very strong (and unachievable) properties such as *indistinguishability from a random oracle*. A key question is, thus, to find notions that, on the one hand, are achievable but which also allow us to replace random oracles in applications. In their original work, Bellare, Hoang, and Keelveedhi (BHK; [BHK13:p]) present the UCE framework in terms of two assumptions called UCE1 and UCE2. We formally introduce UCE1 and UCE2 in Section 9.2 but here relay a general idea behind the restriction of UCE1 (the weaker of the two notions). In fact, we have already come across a similar restriction when discussing point-function obfuscation in the presence of auxiliary information (see Chapter 6). The restriction behind UCE1 is called *computationally unpredictable sources* and requires that the leakage output by a source computationally hides the source's queries when it interacts with a random oracle. The idea is that now the simple recomputation attack is no longer valid since for this the source must leak a query x and image y and, thus, is not unpredictable as a predictor can pick out x from the leakage. Also note that unpredictability is defined with respect to a random oracle. The rational behind this choice is that whether or not a source adheres to a restriction should be a property of the source only: that is, the source is either unpredictable or it is not.

BHK show that the two notions, UCE1 and UCE2, are sufficient to replace random oracles in a large number of applications and they specifically list *maximizing applicability* as a design goal. They write

> *"Our position is philosophically different from that of [34, 39]. These works aimed for security notions that they could achieve under standard assumptions. Expectedly, applicability was limited. We aim to maximize applicability and are willing to see our notion (UCE) as an assumption rather than something to achieve under other assumptions."*
>
> [BHK13:p]

Assuming that UCE1 (the weaker of the two notions) can be instantiated in the standard-model, then this allows us to instantiate random oracles in order to obtain universal hardcore functions, correlated-input secure hash functions, IND-secure deterministic public-key encryption, message-locked encryption and much more. In particular, UCE1 gives rise to the first standard model constructions for many different primitives that, before, were only known to exist in the random oracle model. The even stronger notion UCE2 was used primarily for obtaining adaptively secure garbling schemes [Yao82a, BHR12]. We provide an overview of applications in Table 9.1.

9.1.2 UCE Assumptions

Our work presented in this thesis is tightly intertwined with the work of BHK on UCEs. Indeed, the uninstantiability results that we presented in the previous chapters, chronologically, came after our first uninstantiability results for UCEs (which we will present shortly) and were in parts motivated by the question which forms of UCEs can be constructed in the standard model (this, in particular, motivated the research on Encrypt-with-Hash; Chapter 8). Thus, in order to understand why UCEs are what they are today we do need to understand their history.

Notion	Description	UCE
HC	UCE1 secure function families are hardcore for any one-way function on any unpredictable distribution for which the function is still one-way. We have introduced universal hardcore functions in Section 4.3 and we formally recall the result in Section 9.4.1.	UCE1
BR93	Being universal hardcore functions, UCE1 secure functions may instantiate the random oracle in the BR93 PKE scheme (see Section 4.3.3).	UCE1
CIH	Any UCE1 secure function is correlation-input secure in the strongest sense (see Section 4.2).	
STORE	Ristenpart, Shacham, and Shrimpton [RSS11] give a simple and efficient proof-of-storage protocol in the random oracle model. UCE1 secure functions can securely instantiate the random oracle.	UCE1
IMMU	UCE1-secure functions can be used to construct secure immunizers as countermeasure for backdoored PRGs [DGG$^+$15].	UCE1
CCA	UCE1-secure functions can be used to construct CCA secure public-key encryption [MH14b].	UCE1
MLE	Any UCE1-secure function can instantiate the random oracle in the convergent encryption transformation of Douceur et al. [DAB$^+$02] which was first formalized by Bellare et al. [BKR13]. For a brief introduction to MLE see Section 8.6.1.	UCE1
D-PKE	Any UCE1-secure function can securely instantiate the random oracle in the Encrypt-with-Hash transformation of Bellare et al. [BBO07] (also see Chapter 8). Furthermore, UCE1-secure functions can instantiate our Hybrid and Double Encrypt-with-Hash (HD-EwH) transformation from Section 8.5.1. We show the latter in Section 9.4.2.	UCE1
KDM	UCE1-secure functions can instantiate the random oracle in the BRS scheme of Black, Rogaway, and Shrimpton [BRS03] to obtain a standard-model symmetric encryption scheme that is secure in the presence of key-dependent messages.	mUCE1
RKA	UCE1-secure functions can be used to construct symmetric encryption schemes secure against related-key attacks.	mUCE1
PFOB	UCEs imply point-function obfuscation. We discuss this relationship in detail in Chapter 12.	mUCE1
GB	UCE2-secure functions can be used to obtain adaptively secure garbling schemes with short tokens.	UCE2
OAEP	Key-independent IND-CPA security of OAEP [BR95] can be obtained by assuming partial one-wayness (UCE1) or one-wayness (UCE2) of the underlying trapdoor function.	UCE1/2

Table 9.1: Applications of Universal Computational Extractors. If no additional reference is given then the application is given by Bellare, Hoang, and Keelveedhi in [BHK13:p, BHK13:f]. mUCE refers to a variant of UCEs that handles multiple hash keys. We introduce mUCEs in Section 9.2.1.

Bellare, Hoang, and Keelveedhi presented UCEs in 2013 around the same time that Garg et al. [GGH$^+$13b] presented their candidate construction for indistinguishability obfuscation. Soon after BHK introduced UCEs, we were able to show that certain forms of UCEs and, in particular, UCE1 and UCE2 are susceptible to obfuscation based attacks. We showed that, assuming indistinguishability obfuscation exists for all circuits, then UCE1 and UCE2 cannot be realized in the standard model [BFM13a]. This result lead to a flurry of new UCE notions and to a shift from understanding

UCEs as a simple (and mostly single) assumption that allows to replace random oracles in a large number of applications, towards considering UCE to be a framework of assumptions that may be tailored for a specific application.[3]

Having now many UCE notions and knowing that some notions cannot be achieved in the standard model raises the question:

What makes a good UCE assumption?

The view we take on this for this thesis is very simple:

A good UCE assumption can be validated down to standard-model assumptions.

In particular, we believe that an assumption which is supposed to be used as further validation that a scheme proven secure in the random oracle model is in fact secure, should come with further validation than that its achievable in the random oracle model.

We make the first steps in this direction in this thesis and show that most UCE notions fall into one of two categories. Either we can show that they are susceptible to obfuscation based attacks, or we can instantiate them with obfuscation. Both, for our positive and negative results we rely mostly on indistinguishability obfuscation. While this leaves open the possibility that the originally proposed UCE notions may be achievable in case indistinguishability obfuscation turns out not to exist after all, the evidence that we have so far points in the other direction: while we have several candidate constructions for indistinguishability obfuscation we do not have a candidate construction for UCE1 or UCE2 (other than in the random oracle model). For the notions that we cannot yet classify, further research is necessary. Meanwhile, our positive results suggest that UCE, after all, could be regarded as a single assumption with certain specializations. We show that one of the notions that emerged due to our initial attacks is almost as universal as the original notions **and** we can show that it exists in the standard model assuming indistinguishability obfuscation and certain forms of point obfuscation.

On referencing BHK. As explained, our negative results lead to updates of the UCE framework. We, thus, clarify the various references in order to properly present the history of the UCE framework. We reference three versions of the UCE framework together with the original proceedings publication that appeared at CRYPTO 2013 [BHK13:p]. We reference the corresponding full version by [BHK13:f]. After communicating a first attack on UCE1 [BFM13a], BHK updated their paper to introduce various new UCE notions [BHK13:1]. We published our initial attack and an extension at CRYPTO 2014 [BFM14a], which in turn lead to an update of the UCE paper which we refer to as [BHK13:2].

9.1.3 Outline

We start with a formal definition of the UCE framework and the original UCE notions (Section 9.2). We then show that the strongest UCE notion can be constructed in the random oracle model (Section 9.3). We present an extended result showing that HMAC is mUCE1 secure (an extension of UCE1 allowing the adversary to use multiple keys) assuming that the underlying compression function is a fixed-length random oracle. This solves an open problem from [BHK13:f]. In Section 9.4 we show

[3] We note that UCEs were formalized as a framework from day one.

how to work with UCEs presenting the proof by BHK [BHK13:f] that any UCE1-secure function is a universal hardcore function as well as showing that UCE1-secure functions are sufficient to instantiate our extended Hybrid and Double Encrypt-with-Hash transformation (HD-EwH) from Section 8.5.1. We then present our basic impossibility result showing that if indistinguishability obfuscators exist that then no standard-model function can be UCE1-secure (Section 9.5). Subsequently, in Section 9.6, we discuss new UCE notions that emerged as a result of our impossibility result and we close this chapter by showing an interesting equivalence between two of these notions 9.7.2.

Jumping ahead, in Chapter 10 we show how to further extend our impossibility result to cover additionally restricted UCE notions and then in Chapter 11 show how to construct certain UCE notions in the standard-model.

9.2 Defining Universal Computational Extractors

We begin with the formal definition of UCEs. Loosely speaking, UCEs are PRF-like assumptions that split a distinguisher (which should differentiate whether it operates relative to a UCE function or relative to a random oracle) into two parts: a first adversary S (called the *source*) gets oracle access to a keyed hash function or a random oracle but not the hash key, and a second adversary D (called the *distinguisher*) gets access to the hash key hk as well as to leakage produced by the source. The two algorithms together try to guess whether the source was given access to a keyed hash function or to a random oracle.

Concretely, the UCE notions are defined via a two-stage UCE game (we present the pseudocode in Figure 9.2 on the left). First, the source S is run with oracle access to Hash to output some leakage L. Subsequently, distinguisher D is run on the leakage L and hash key hk but without access to oracle Hash. Distinguisher D outputs a single bit b indicating whether oracle Hash implements a random oracle or hash function H with key hk.

Formal UCE definition. We denote hash function (families) by H. Let $\mathsf{H} = (\mathsf{H.KGen}, \mathsf{H.Eval}, \mathsf{H.kl}, \mathsf{H.il}, \mathsf{H.ol})$ be a hash-function family and let (S, D) be a pair of PPT algorithms. We define the UCE advantage of a pair (S, D) against H through

$$\mathsf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda) := 2 \cdot \Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1\right] - 1,$$

where game $\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)$ is shown in Figure 9.2 on the left.

Restricting sources to obtain concrete UCE assumptions. Without any restrictions the UCE game models that a keyed standard-model function is indistinguishable from a random oracle and hence it is not surprising that there exists a pair (S, D) of efficient algorithms that can win the UCE game. For example, say, source S makes a random query x to receive $y \leftarrow\!\!\$\ \textsc{Hash}(x)$ and then outputs the query answer pair (x, y) as leakage. As distinguisher D knows the hash key hk as well as the leakage (x, y), it can recompute the hash value and check whether $y = \mathsf{H}(\mathsf{hk}, x)$. In order to get a concrete UCE notion it is, thus, necessary to define a restriction on sources and distinguishers. Given this flexibility a key issue is to come up with *good UCE notions* that are not too strong (that is, in particular, they should be instantiable in the standard model) and, on the other hand, are not

$\underline{\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)}$ $\underline{\textsc{Hash}(x)}$

$b \leftarrow\!\$\, \{0,1\}$
$\mathsf{hk} \leftarrow\!\$\, \mathsf{H.KGen}(1^\lambda)$
$L \leftarrow\!\$\, \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$b' \leftarrow\!\$\, \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$

if $T[x] = \bot$ **then**
  if $b = 1$ **then**
    $T[x] \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$
  else $T[x] \leftarrow\!\$\, \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$

$\underline{\mathrm{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)}$ $\underline{\textsc{Hash}(x,i)}$

$(1^n, \mathsf{state}) \leftarrow\!\$\, \mathsf{S}(1^\lambda, \varepsilon)$
$b \leftarrow\!\$\, \{0,1\}$
for $i = 1, \ldots, n,$ **do**
  $\mathbf{hk}[i] \leftarrow\!\$\, \mathsf{H.KGen}(1^\lambda)$
$L \leftarrow\!\$\, \mathsf{S}^{\textsc{Hash}}(1^n, \mathsf{state})$
$b' \leftarrow\!\$\, \mathsf{D}(1^\lambda, \mathbf{hk}, L)$
return $(b = b')$

if $T[x,i] = \bot$ **then**
  if $b = 1$ **then**
    $T[x,i] \leftarrow \mathsf{H.Eval}(\mathbf{hk}[i], x)$
  else $T[x,i] \leftarrow\!\$\, \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x,i]$

Figure 9.2: The UCE security game on the left together with its multi-key variant mUCE on the right. In the UCE game source S has access to an oracle Hash, which returns real or ideal hash values, and leaks L to a distinguisher D. The latter additionally gets the hash key that was used in the previous stage (or a uniformly random key in case Hash implemented a random function) and outputs a bit b'. The multi-key variant mUCE on the right considers a scenario where source S can talk to multiple functions each either implementing an independent random function or the hash function with a uniformly random and independently chosen key.

too weak in order to be useful in applications. In order to capture restrictions for sources we define UCE security relative to a class of sources $\mathcal{S}$ and all PPT distinguishers.

Definition 9.1. *We say that a hash function* H *is* UCE-secure relative to source class $\mathcal{S}$ *denoted by* $\mathsf{H} \in \mathrm{UCE}[\mathcal{S}]$*, if for all* PPT *sources* $\mathsf{S} \in \mathcal{S}$ *and all* PPT *distinguishers* D *the advantage* $\mathsf{Adv}^{\mathrm{uce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda)$ *is negligible.*

Similarly, we could restrict distinguishers, but note that for all UCE notions that have been proposed so far, distinguishers were not restricted beyond the fact that they are assumed to be efficient, that is, PPT.

9.2.1 Multi-Key UCEs

Besides UCEs with a single hash key hk, BHK also define a multi-key version called mUCE to capture scenarios in which multiple hash keys are used. Multi-key UCE works analogously to plain UCEs with the exception that source S can decide with how many keys it wants to work and oracle Hash takes as input an index specifying the key [BHK13:p]. We present the pseudocode of the extension in Figure 9.2 on the right. Similarly to the single-key case, we need to consider specific restrictions on source (and distinguisher) to obtain an achievable notion. For restrictions in the single-key case one can usually construct an analogue for the multi-key case by requiring that the restriction holds for any key. For natural restrictions, the multi-key version then implies the single-key version although the inverse is an open research question.

For the remainder of this thesis we will, mostly, consider the single-key UCE variant and only sporadically refer to the multi-key variant when necessary.

9.2.2 The Original Assumptions: UCE1 and UCE2

As discussed in the introduction, many different concrete UCE notions have been proposed and we discuss the most prominent in detail in Section 9.6. We next discuss the two original UCE notions as defined by BHK in [BHK13:p]: UCE1 and UCE2 which were later renamed into $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}}]$ (UCEs with respect to *computationally unpredictable sources*) and $\mathrm{UCE}[\mathcal{S}^{\mathrm{crs}}]$ (UCEs with respect

$\underline{\mathrm{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)}$
$\mathsf{done} \leftarrow \mathbf{false}$
$Q \leftarrow \{\}$
$L \leftarrow\!\!\$\ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$\mathsf{done} \leftarrow \mathbf{true}$
$Q' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$
$\mathbf{return}\ \ (Q \cap Q' \neq \{\})$

$\underline{\textsc{Hash}(x)}$
$\mathbf{if}\ \mathsf{done} = \mathbf{false}\ \mathbf{then}$
  $Q \leftarrow Q \cup \{x\}$
$\mathbf{if}\ T[x] = \bot\ \mathbf{then}$
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$\mathbf{return}\ \ T[x]$

$\underline{\mathrm{Reset}^{\mathsf{R}}_{\mathsf{S}}(\lambda)}$
$Q \leftarrow \{\}$
$b \leftarrow\!\!\$\ \{0,1\}$
$L \leftarrow\!\!\$\ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$
$\mathbf{if}\ b = 0\ \mathbf{then}$
  $\mathbf{forall}\ x \in Q\ \mathbf{do}$
    $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$b' \leftarrow\!\!\$\ \mathsf{R}^{\textsc{Hash}}(1^\lambda, L)$
$\mathbf{return}\ \ (b = b')$

$\underline{\textsc{Hash}(x)}$
$Q \leftarrow Q \cup \{x\}$
$\mathbf{if}\ T[x] = \bot\ \mathbf{then}$
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$\mathbf{return}\ \ T[x]$

Figure 9.3: The prediction and reset-security game for UCE sources.

to *computationally reset-secure sources*). We will henceforth refer to UCE1 and UCE2 by their new names and formal source restrictions.

Formally, we denote by $\mathcal{S}^{\mathrm{cup}}$ the class of all computationally unpredictable sources and call a source S *computationally unpredictable* if the advantage of any PPT predictor P, defined by

$$\mathsf{Adv}^{\mathrm{pred}}_{\mathsf{S},\mathsf{P}}(\lambda) := \Pr\left[\mathrm{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda) = 1\right],$$

is negligible, where game $\mathrm{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ is shown in Figure 9.3 on the left.[4] Note that computational unpredictability is similarly defined as for point-function obfuscation (see Definition 6.5). For point obfuscation we required that the auxiliary information output by the sampler does not reveal the point address. For UCEs, on the other hand, we require that the leakage of a source does not reveal any entry from the set of its oracle queries. An important observation is that the Pred game is always in the random oracle setting, that is, no hash function family H is part of game Pred. Consequently, the restriction is well-defined in the sense that a source is either computationally unpredictable or not, but the property is solely a property of the source and not a property of the source in combination with a hash function.

UCE2, that is, UCEs with respect to *computationally reset-secure sources* are a strengthening of computational unpredictability in the sense that the class contains more sources: BHK show that any computationally unpredictable source is also a reset-secure source (Proposition 4.2 [BHK13:2]). While all queries by a computationally unpredictable source need to have computational min-entropy a reset-secure source may query known points as long as the leakage does not allow to distinguish whether the oracle was changed after the source finishes. This is captured by game $\mathrm{Reset}^{\mathsf{R}}_{\mathsf{S}}$ which similarly to game Pred is in the random oracle setting and where a source S plays against a reset adversary R. Here, the source is executed as usual and the leakage is then given to reset adversary R which also gets oracle access to Hash. Its goal is to distinguish whether it is connected to the same oracle Hash as the source, or whether the oracle was *reset* on the source's query points. We give the pseudocode in Figure 9.3 on the right.

Formally, we denote by $\mathcal{S}^{\mathrm{crs}}$ the class of all computationally unpredictable sources and call a

[4] BHK give also a simplified form of the unpredictability game where predictor Pred does not get access to oracle Hash and is allowed to output only a single guess. They call this *simple unpredictability* and show that a source S is computationally unpredictable if and only if it is simple computationally unpredictable [BHK13:p, Lemma1].

source S *computationally reset-secure* if the advantage of any PPT reset adversary R, defined by

$$\mathsf{Adv}^{\text{reset}}_{\mathsf{S},\mathsf{R}}(\lambda) := \Pr\left[\text{Reset}^{\mathsf{R}}_{\mathsf{S}}(\lambda) = 1\right],$$

is negligible.

Remark. BHK explain that the name "Universal Computational Extractors" was chosen in reference to UCE1, that is, in reference to computational unpredictability [BHK13:p]. As noted by BHK, the term *computational extractor* has been associated with primitives that extract (pseudo)-randomness from distributions that have (computational) min-entropy. UCEs, similarly extract randomness but from distributions that are restricted otherwise; restricted to be computationally unpredictable in case of UCE1. The term "universal" refers to the fact that they should be able to do so for *all* such restricted distributions. [BHK13:p].

Applicability of UCE1 and UCE2

UCE1 and UCE2 allow to instantiate random oracles in a wide range of interesting applications: deterministic public-key encryption (D-PKE), message-locked encryption (MLE), universal hardcore functions (HC), point-function obfuscation (MBPO), key dependent message security (KDM), related key security (RKA), correlation input secure hash functions (CIH) and more. In particular, the (weaker) notion of UCE[$\mathcal{S}^{\text{cup}}$] (aka. UCE1) alone allows to instantiate the random oracle in all previously mentioned applications. The notion UCE[$\mathcal{S}^{\text{cup}}$], thus, seems to capture one important property of random oracles which makes it applicable in a wide variety of applications. This was further strengthened by Matsuda and Hanaoka who show how to build CCA secure public-key encryption from UCE[$\mathcal{S}^{\text{cup}}$] secure functions [MH14b].

The notion of reset-security, aka. UCE[$\mathcal{S}^{\text{crs}}$], further strengthens UCE[$\mathcal{S}^{\text{cup}}$] by capturing scenarios in which low-entropy queries may occur as long as for these the corresponding images remain computationally hidden. BHK show that reset-security allows to extend a result on instantiating the random oracle in OAEP [BR95] for obtaining IND-CPA-KI secure PKE schemes[5] and, furthermore, can be used to obtain adaptively secure garbling schemes.

In the Section 9.4 we show how to work with UCEs and reprove BHK's result for deterministic public-key encryption but in a slightly different setting. In the next section we show how to construct a mUCE[$\mathcal{S}^{\text{crs}}$] secure function in an idealized setting.

9.3 Constructing UCEs in Idealized Models

BHK envisioned UCEs to form a layer between the random oracle model and actual constructions. That is, cryptographic schemes are no longer validated directly in the idealized model but instead are validated down to a UCE assumption: a standard model assumption. They referred to this approach as *layered cryptography* and compared it to *pairing-based* cryptography where often also standard-model assumptions are validated in the idealized generic-group model while actual constructions are then build on top of the standard model assumptions. As examples, Bellare et al. cite BDH [BF03],

[5] IND-CPA-KI refers to IND-CPA secure for key-independent messages. Note that usually, in the IND-CPA setting the adversary is given the public-key and may choose its messages depending on it.

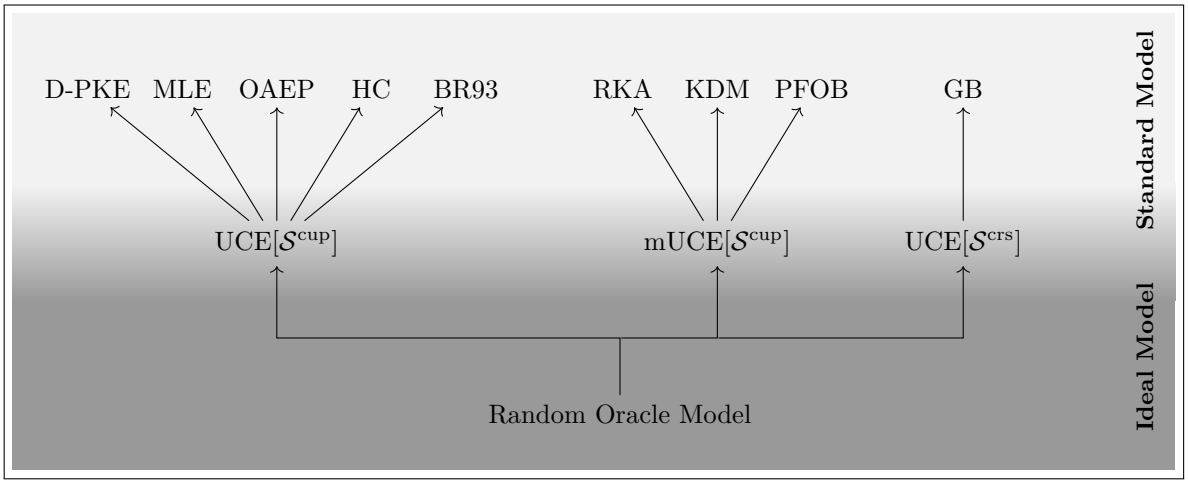


Figure 9.4: The layered cryptography approach of Bellare, Hoang, and Keelveedhi [BHK13:p]. Here, *base primitives* with standard-model security definitions (such as UCE notions) are validated in the random oracle model. *End goals* are the build on top of the base primitives and are, thus, reached solely in the standard model. For the layer of assumptions in between it is, however, not clear if these assumptions can be met in the standard-model or not.

DLIN [BBS04], SDH [BBS04], and more. We visualize the idea in Figure 9.4 where strong standard-model assumptions are validated in the random oracle model to then be used to reach *end goals* solely in the standard model.

Such a layered approach, with UCEs forming the gateway between the idealized random oracle model and the standard model offers various advantages over directly proving schemes secure in the random oracle model. On the one hand, UCEs have a clear definition which makes them a better cryptanalytic target. "Behaving like a random oracle" on the other hand is vague and not clearly graspable. Furthermore, having a layer in-between gives rise to better understanding what properties of the random oracle a particular scheme needs, as these properties are clearly defined. Finally, opting for assumptions that are validated in an ideal model, rather than being build on top of existing standard-model assumptions, may allow for a much broader applicability of the assumption as it can more easily capture a larger part of the ideal model. The downside is, of course, that such a layered approach and wide applicability may again fall prey to the inconsistencies of the idealized model which we hoped to eliminate with the standard-model assumption and proof in the first place.

Constructing UCEs in Idealized Models

Having shown that UCE[$\mathcal{S}^{\text{crs}}$] and UCE[$\mathcal{S}^{\text{cup}}$] is sufficient for many applications, what remains is to validate the assumptions in the random oracle model. BHK show that in the random oracle model the simple construction $\mathsf{H}^{\mathsf{RO}}(\mathsf{hk}, m) := \mathsf{RO}(\mathsf{hk} \| m)$ which prepends the hash key to the message and then evaluates the result on the random oracle achieves the strongest proposed UCE notion: mUCE[$\mathcal{S}^{\text{crs}}$].[6] We recall their theorem:

Theorem 9.1 ([BHK13:f], Theorem 7.1)**.** *Let* H *be a hash function family defined as follows: Key generation* $\mathsf{H}^{\mathsf{RO}}.\mathsf{KGen}$ *on input security parameter* 1^λ *returns a hash key* $\mathsf{hk} \leftarrow\!\!\$\, \{0,1\}^\lambda$ *uniformly at*

[6] Note that this simple construction is exactly the pseudorandom function that we constructed in the random oracle model in Section 3.3 when introducing the random oracle methodology.

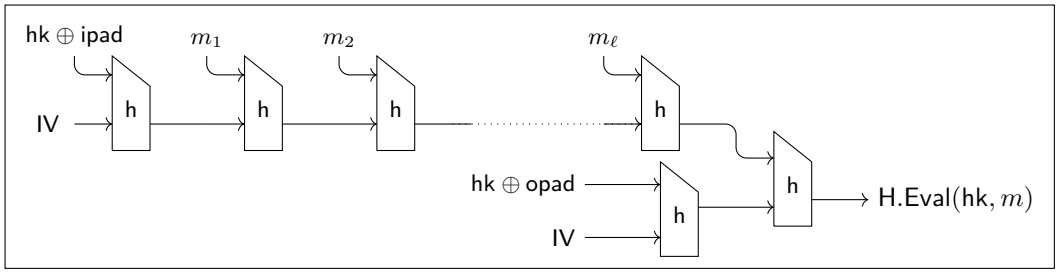


Figure 9.5: The HMAC construction evaluated on a message $m_1 \| \ldots \| m_\ell := m$ where $|m| := \ell \cdot \mu$ is a multiple of the block size μ. If the dashed boxes are exchanged for independent keys k_1 and k_2, we obtain the NMAC construction. Note that in this picture we are ignoring padding.

random. Evaluation relative to random oracle RO *is defined as* $\mathsf{H}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{hk}, m) := \mathsf{RO}(\mathsf{hk} \| m)$*. Then* $\mathsf{H}^{\mathsf{RO}} \in \text{mUCE}[\mathcal{S}^{\text{crs}}]$*, that is, function family* H *is* mUCE *secure with respect to all reset-secure sources* $\mathcal{S}^{\text{crs}}$*.*

Instead of prepending the key to the message one could similarly opt for the keyed random oracle model in which we could show the same result, that is, the keyed random oracle itself is $\text{mUCE}[\mathcal{S}^{\text{crs}}]$ secure. We do not repeat the proof and refer the interested reader to [BHK13:2].

Having constructed UCEs in the random oracle model again raises the question: *what to use in actual implementations in place of a UCE secure function?* BHK argue that the natural candidate, a block cipher such as AES, is not secure since given the key it is efficiently invertible. Instead they propose the use of keyed cryptographic hash functions (note that functions such as SHA256 are not keyed) and propose to use HMAC [BCK96, KBC97].

Given an unkeyed hash function H, one can construct HMAC on top of H as

$$\mathsf{HMAC}[\mathsf{H}](\mathsf{hk}, m) := \mathsf{H}\left((\mathsf{hk} \oplus \mathsf{opad}) \| \mathsf{H}\left((\mathsf{hk} \oplus \mathsf{ipad} \| M)\right)\right),$$

where opad and ipad are constants. Assuming that the underlying hash function is constructed from a compression function $\mathsf{h} : \{0,1\}^\mu \times \{0,1\}^\ell \to \{0,1\}^\ell$ according to the Merkle-Damgård principle [Mer90, Dam90]—examples are, MD5, SHA1, or SHA256—we have that HMAC is an iterated construction. That is, HMAC is constructed by splitting a message m into blocks of length μ and then processing the message block by block by iterating compression function h as depicted in Figure 9.5. Here, IV is a fixed value depending on the underlying hash function H. Further note that hash key hk is only used in the two "outer" h evaluations once xored with a fixed value ipad and once with fixed value opad (with $\mathsf{ipad} \neq \mathsf{opad}$).

A note for the impatient reader. The following construction of UCEs in the fixed-input length random oracle model is irrelevant for the remainder of this thesis and can, thus, be safely skipped.

Constructing UCEs in the Ideal Compression Function Model

Being highly structured, HMAC should *behave* very differently from a truly random function. But, can we detect this, that is, does the iterative structure lead to exploitable weaknesses when used

Example 9.1: Length Extension Attack on SHA1

Hash functions strictly following the Merkle-Damgård design—prominent examples are MD5, SHA1, and SHA512—are subject to length extension attacks. Let hash function H be a Merkle-Damgård based function [Mer90, Dam90] with compression function h. Then, for a message $M := M' \| m$ where $|M'|$ is a multiple of the block length and $|m|$ has the length of a single block, we can compute $\mathsf{H}(M)$ by computing $\mathsf{h}(m, \mathsf{H}(M'))$. In other words, given a hash value for some message M allows an adversary to compute hash values for messages that have prefix M; a property that is not shared by an ideal hash function.

instead of a truly random function? This question was first asked by Coron et al. [CDMP05] who studied iterated hash constructions such as HMAC under the premise that the iterated function h is ideal, that is, a fixed-input length random oracle, but that otherwise the structure of the function is observed. Indeed, an adversary against a scheme employing an iterated function can be assumed to know the structure and, if possible, to exploit it. We give an example of how the structure of hash functions can be exploited by an adversary as Example 9.1.

Indifferentiability. The *indifferentiability framework* of Maurer, Renner, and Holenstein [MRH04] is the prevalent notion of equivalence between ideal primitives and is also used by Coron et al. [CDMP05] in their study of iterated hash functions. Assume that we would like to argue that a construction G^{π} with oracle access to an ideal primitive π can be used in place of an ideal primitive Π. In our case π would be a fixed-length random oracle, construction G is HMAC and we would like to use G^{π} instead of a full-fledged random function Π without any input restrictions.

Now, demanding a distinguisher D to distinguish oracle access to (G^{π}, π) from oracle access to Π is of little sense; already the oracle structure reveals in which world the distinguisher lives in. Thus, additionally to the oracle Π, the distinguisher gets access to a simulator Sim which tries to emulate π's behavior consistently with Π. The distinguisher then tries to distinguish the pair of oracles (G^{π}, π) from the pair of oracles $(\Pi, \mathsf{Sim}^{\Pi})$ and we say that G^{π} is indifferentiable from Π if for any distinguisher such a simulator Sim exists. Formally, we define indifferentiability as follows:

Definition 9.2. *A Turing machine G with black-box access to an ideal primitive π is* indifferentiable[7] *from an ideal primitive Π if for any* PPT *distinguisher D there exists a simulator Sim^{Π} such that:*

$$\left| \Pr\left[\mathsf{D}^{G^{\pi},\pi}(1^{\lambda}) = 1\right] - \Pr\left[\mathsf{D}^{\Pi,\mathsf{Sim}^{\Pi}}(1^{\lambda}) = 1\right]. \right| \leq \mathsf{negl}(\lambda)$$

Applied to hash function constructions in general and HMAC in particular, we can, thus, study whether a distinguisher D can distinguish between getting oracle access to $(\mathsf{HMAC}^{\mathsf{h}}, \mathsf{h})$ and getting oracle access to $(\mathsf{RO}, \mathsf{Sim}^{\mathsf{RO}})$. Here, RO is a random oracle, simulator Sim an efficient algorithm, and $\mathsf{h} : \{0,1\}^{\mu} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ denotes a fixed-input length random oracle (the little halo marking h as an idealized primitive).

Coron et al. [CDMP05] show that for HMAC (and various other iterated hash constructions) there exists an efficient simulator Sim such that no efficient distinguisher D can distinguish the two

[7] Sometimes this notion is referred to as *weak indifferentiability*. In this context *strong indifferentiability* considers a reversed order of quantifiers, that is, there exists a simulator for any distinguisher.

worlds, that is, for any PPT distinguisher D we have that

$$\left|\Pr\left[\mathsf{D}^{\mathsf{HMAC}^{\mathsf{h}},\mathsf{h}}(1^\lambda)=1\right]-\Pr\left[\mathsf{D}^{\mathsf{RO},\mathsf{Sim}^{\mathsf{RO}}}(1^\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\,.$$

In other words, HMAC is indifferentiable from a random oracle. This is where the composition theorem of Maurer, Renner, and Holenstein [MRH04] can now be applied to lift a proof in the random oracle setting to a proof in the ideal compression function setting. Assume that there exists an adversary $\mathcal{A}$ for some scheme that exploits access to the underlying fixed-input length random oracle h. Then, we can construct an adversary $\mathcal{A}'$ in the random oracle setting by combining adversary $\mathcal{A}$ with indifferentiability simulator Sim. That is, we set

$$\mathcal{A}'^{\mathsf{RO}} := \mathcal{A}^{\mathsf{RO},\mathsf{Sim}^{\mathsf{RO}}}.$$

If adversary $\mathcal{A}'$ is not similarly successful then this yields a distinguisher against the indifferentiability of the hash function.

Indifferentiability in multi-stage settings. By the above argument, Theorem 9.1 together with the indifferentiability of HMAC should imply that HMAC is mUCE[$\mathcal{S}^{\text{crs}}$] secure in the ideal compression function model. This is, however, not the case. As noted by Ristenpart, Shacham, and Shrimpton [RSS11] the composition theorem for indifferentiable functions does not hold in general, but only in single-stage games where a global adversary can be assumed (resp. multiple adversaries that share arbitrary state). (See Section 2.3.3 for an introduction to the distinction between single-stage and multi-stage games.) The UCE security game with its adversary (S,D), on the other hand, is a multi-stage game since the adversary is split into source S and distinguisher D and the state shared between the two stages (leakage L) is restricted. The reason why the composition theorem fails in this setting is that the simulator needs to see all h-queries and, thus, if the adversary is split over several stages without sharing full state, the simulation of the compression function might fail. A possible remedy would be to find stateless and (pseudo)-deterministic simulators [RSS11, DGHM13, BBM13] but it was shown [DGHM13, BBM13] that for domain-extending constructions, such as HMAC, such simulators cannot exist.

Not all is lost. We can show that under certain conditions one can also work with indifferentiability in multi-stage settings [Mit14]. In [Mit14] we give an extended composition theorem that applies to a class of multi-stage games and allows us to show that, indeed, HMAC is mUCE[$\mathcal{S}^{\text{crs}}$]-secure thereby answering an open problem stated by BHK in [BHK13:p]. In the following, we sketch a direct proof of this statement without the overhead needed for the more general result in [Mit14].

Theorem 9.2. *Let* $\mathsf{h}:\{0,1\}^\mu\times\{0,1\}^\ell\to\{0,1\}^\ell$ *be a fixed length random oracle. Then* $\mathsf{HMAC}^{\mathsf{h}}$ *is* mUCE[$\mathcal{S}^{\text{crs}}$] *secure.*

Let us give some intuition first. We know that if UCE was a single-stage game, then the indifferentiability of HMAC would already be sufficient for the above result. As UCE is not single-stage we, thus, need a different strategy to simulate the h-queries across both stages (that is for source S and distinguisher D) without sharing any state. For this, we will exploit two observations. First we note that source S has oracle access to Hash which may implement $\mathsf{HMAC}^{\mathsf{h}}$. Thus, potentially problematic queries by S to h are those that are identical to queries to h that occur during a Hash

evaluation. However, as source S does not get access to hash key hk it can be shown that such queries are extremely unlikely. Consider the structure of HMAC given in Figure 9.5. The outermost $\mathcal{H}$ queries contain hash key hk and, thus, without guessing key hk source S will not be able to make any of the two outermost queries. Then, however, all the inner queries take as input a completely random string hidden from the source. In summary, we can simulate function $\mathcal{H}$ for source S with a random function chosen independent from the function used by HMAC within oracle HASH. The second observation is that for the second and last stage (distinguisher D) we can use the stateful simulator by Coron et al. [CDMP05] if we can ensure that it answers consistently on queries that were also made by source S. For this note that by the above argument, source S does not benefit at all from access to $\mathcal{H}$ as we can simulate it by lazy sampling. However, source S and distinguisher D may use access to a shared random function to get an edge and we, thus, need to ensure that the view of D is consistent, that is, queries that were already asked by source S are answered identically. For this we will use a derandomization technique due to Bennet and Gill [BG81] which uses the random oracle as source of "pseudorandom" coins.[8] The idea behind the derandomization is that an algorithm running in time t cannot distinguish between getting real random coins or coins generated by evaluating the random oracle on a message of length $t+1$. We next provide a proof sketch and refer to [Mit14] for a formal treatment.

Proof sketch of Theorem 9.2. We will proceed in three steps. We start with the $\mathrm{mUCE}[\mathcal{S}^{\mathrm{crs}}]$ game with hidden bit b fixed to 1, that is, oracle HASH answers using $\mathsf{HMAC}^{\mathcal{H}}$. Additionally, all parties (i.e., source and distinguisher) get access to ideal functionality $\mathcal{H}$. We denote this setup as Game_1 which we depict in Figure 9.6 on the left. Note that we provide each party (source S, distinguisher D, and construction HMAC) with their own oracle-interface all of which, however, work on the same underlying table T_{h}. That is, the interface to $\mathcal{H}$ given to S, D and HMAC are denoted by $\mathcal{H}_{\mathsf{S}}$, $\mathcal{H}_{\mathsf{D}}$, and $\mathcal{H}_{\mathsf{H}}$, respectively. By construction we, thus, have that

$$\Pr[\mathsf{Game}_1(\lambda)=1] = \Pr\left[\mathrm{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{HMAC}^{\mathsf{h}}}(\lambda)=1 \;\middle|\; b=1\right].$$

We next ensure that source S with its direct queries to $\mathcal{H}_{\mathsf{S}}$ does not make any query that collides with a query to $\mathcal{H}_{\mathsf{H}}$ by an HMAC evaluation triggered through a HASH query. For this, all the source's $\mathcal{H}_{\mathsf{S}}$ queries are recorded in set Q_{S} and similarly all queries to $\mathcal{H}_{\mathsf{H}}$ are recorded in a set Q_{H}. After the source is finished it is checked if a collision occurred (line 6). If so, we abort the game, by which we mean that we stop and the game outputs 1 (i.e., the adversary wins). This change is captured in Game_2 (Figure 9.6).

By the fundamental lemma of the game-playing technique [BR06] we have that games Game_1 and Game_2 are identical unless the abort occurs. For source S to make a colliding query there are two cases to consider: either the source guesses one of the keys (which would enable it to make one of the outer HMAC queries; $\mathsf{hk}\oplus\mathsf{ipad}$ or $\mathsf{hk}\oplus\mathsf{opad}$), or it "accidentally" hits an inner query by guessing one of the random ℓ-bit values returned by an $\mathcal{H}_{\mathsf{H}}$ call. As keys are of length μ and there are n keys and for each key there are two *outer* queries we thus have that

$$|\Pr[\mathsf{Game}_2(\lambda)=1] - \Pr[\mathsf{Game}_1(\lambda)=1]| \leq \frac{2\cdot n\cdot q^{\mathcal{H}}_{\mathsf{S}}}{2^{\mu}} + \frac{q^{\mathsf{h}}_{\mathsf{S}}\cdot q^{\mathsf{h}}_{\mathsf{HMAC}}}{2^{\ell}},$$

[8] Bennet and Gill show that relative to a random oracle the complexity classes BPP and P are equivalent [BG81].

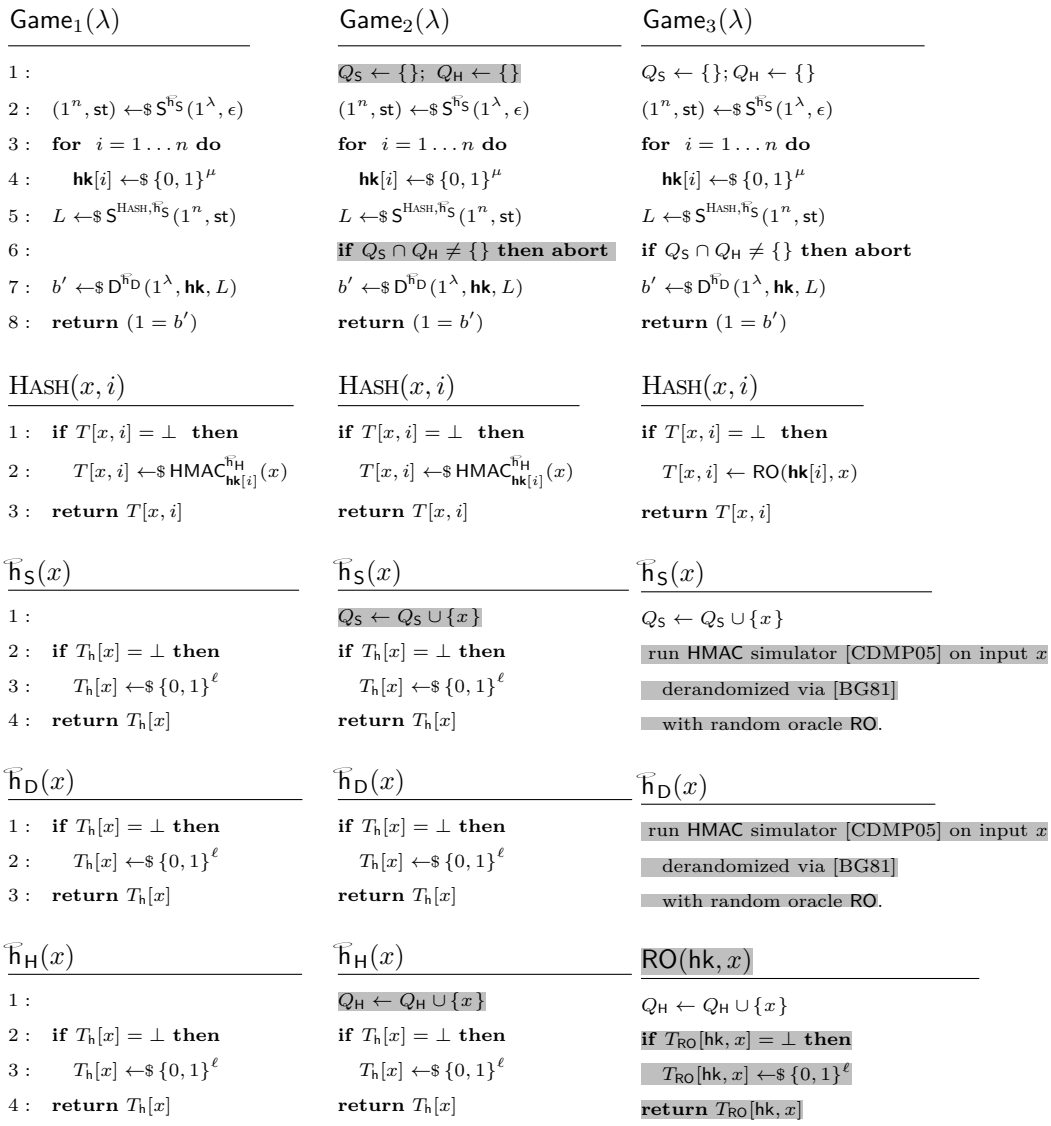


Figure 9.6: Security games in proof of Theorem 9.2.

where $q^{\tilde{h}}_{\mathsf{S}}$ denotes an upper bound on the number of $\tilde{h}_{\mathsf{S}}$-queries by the source and $q^{\tilde{h}}_{\mathsf{HMAC}}$ is an upper bound on the number of $\tilde{h}_{\mathsf{H}}$-queries during HMAC-evaluations triggered by HASH-oracle calls.

In game Game_3 (Figure 9.6 on the right) we make the jump to the random oracle setting. For a full formal analysis, of how this game hop can be achieved we refer to [Mit14]. In Game_3, instead of using HMAC the HASH-oracle now uses the keyed random oracle. The $\tilde{h}$-oracles expected by source and distinguisher are now implemented using the simulator which similarly gets oracle access to the random oracle. For this we set $\mathsf{Sim}^{\mathsf{RO}}$ to be the indifferentiability simulator from [CDMP05] and derandomize it relative to the random oracle via a technique from [BG81]. The derandomization depends on the maximum runtime t of the source and the distinguisher.

The derandomization ensures that the two distinct instantiations of the simulator (one for the source and one for the distinguisher) are consistent on common $\tilde{h}$-queries between source and dis-

tinguisher. For this we note that for the generation of random values by the simulator, the chosen value only depends on the randomness of the simulator and the query itself, but not on accumulated state. That is, the simulator either responds with a random value or with a deterministically generated value [CDMP05]. Thus, for derandomization it is sufficient to "deterministically generate the randomness" for the generation of such random values. For this, we set the random coins of the simulator on a query (m, x) to be

$$\mathsf{RO}(0^{t+1} \| m \| x)$$

that is, the random oracle evaluated on point $0^{t+1} \| m \| x$. (If the simulator needs more random coins than can be obtained by a single random oracle call, we run the random oracle in counter mode.) Since source and distinguisher have a runtime bound of t they cannot query the random oracle on such a value and, thus, the distribution of such deterministically generated random coins is identical to that of the original simulator.

What is left to show is that games Game_2 and Game_3 are indistinguishable. For this note that simulator Sim is the indifferentiability simulator for HMAC which answers with random values unless the query belongs to an actual HMAC execution [CDMP05].[9] What we mean by this is that the simulator can check whether a query corresponds to a valid HMAC execution by storing previous queries and checking if there is a path from the current query to an initial query, that is, a query of the form $(\mathsf{hk} \oplus \mathsf{ipad}, \mathsf{IV})$ (cf. Figure 9.5). In order for the simulator to check whether a query belongs to an HMAC execution it, thus, needs to be initialized with the correct key (or keys $\mathbf{hk}$) in our case. The distinguisher knowing the keys can initialize its simulator with them. For the source, we note that we have discarded collisions between $\mathsf{HMAC}^{\widetilde{\mathsf{h}}}$ queries and $\widetilde{\mathsf{h}}_{\mathsf{S}}$-queries by source S in the second game hop. Thus, $\widetilde{\mathsf{h}}_{\mathsf{S}}$-queries by source S will never be valid HMAC-queries and, hence, it is sufficient for the simulator when working with source S to not check queries but to always return random values. It follows that the difference between Game_2 and Game_3 is negligible, that is,

$$|\Pr[\mathsf{Game}_3(\lambda) = 1] - \Pr[\mathsf{Game}_2(\lambda) = 1]| \leq \mathsf{negl}(\lambda)\,.$$

Finally, Game_3 is equivalent to the mUCE-security game with hidden bit b set to 1 and for scheme $\mathsf{H}^{\mathsf{RO}}.\mathsf{Eval}(\mathsf{hk}, m) := \mathsf{RO}(\mathsf{hk}, m)$. The remainder of the proof follows with Theorem 9.1. □

9.4 Using UCEs

In this section we want to show how to work with UCEs in general and with UCE[$\mathcal{S}^{\text{cup}}$] in particular. We present two examples:

1. We reprove a result by BHK [BHK13:f], namely that any UCE[$\mathcal{S}^{\text{cup}}$] secure function is a universal hardcore function (see Definition 4.2).

2. We use UCE[$\mathcal{S}^{\text{cup}}$] to show that our Hybrid and Double Encrypt-with-Hash transformation (HD-EwH) from Section 8.5.1 is IND-secure in the random oracle model.

[9] We note that Coron et al. [CDMP05] not only consider the ideal compression function setting but go one step further and consider that the compression function can be constructed via the Davies-Meyer construction [Win83]. A careful argument, thus, allows to further strengthen the result here to capture also HMAC based on ideal ciphers.

9.4.1 Universal Hardcore Functions from UCE

If f is a one-way function and H is hardcore for f then with H one can extract random bits from a preimage x even in the presence of $\mathsf{f}(x)$. If H is hardcore for any (keyed) one-way function then we say that H is a *universal hardcore function*. We formally introduce hardcore functions in Section 4.3.

BHK establish that any UCE[$\mathcal{S}^{\text{cup}}$] secure function is a universal hardcore function [BHK13:f]. We recall their result:

Theorem 9.3 ([BHK13:f], Theorem 5.1)**.** *If* $\mathsf{H} \in \text{UCE}[\mathcal{S}^{\text{cup}}]$ *then* H *is a universal hardcore function (i.e., a hardcore function for any one-way function* f*).*

Proof. Let f be a one-way function. We show that for any adversary $\mathcal{A}$ against the hardcore property of H for f, there exists a PPT source S and PPT distinguisher D such that

$$\mathsf{Adv}^{\text{hc}}_{\mathsf{f},\mathsf{H},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{uce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda). \tag{9.1}$$

As by assumption $\mathsf{Adv}^{\text{uce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda)$ is negligible this proves the claim.

We construct pair (S,D) to imitate the hardcore game. That is, we consider the following algorithms (we give the hardcore game for reference on the right):

$\underline{\mathsf{S}^{\text{HASH}}(1^\lambda)}$

$\mathsf{fk} \leftarrow\!\$\ \mathsf{f.KGen}(1^\lambda)$
$x \leftarrow\!\$\ \{0,1\}^{\mathsf{f.il}(\lambda)}$
$y \leftarrow \mathsf{f.Eval}(\mathsf{fk}, x)$
$r \leftarrow\!\$\ \text{HASH}(x)$
$L \leftarrow (\mathsf{fk}, y, r)$
return L

$\underline{\mathsf{D}(1^\lambda, \mathsf{hk}, L)}$

$(\mathsf{fk}, y, r) \leftarrow L$
$b' \leftarrow\!\$\ \mathcal{A}(1^\lambda, \mathsf{fk}, \mathsf{hk}, y, r)$
return b'

$\underline{\text{HC}^{\mathcal{A}}_{\mathsf{f},\mathsf{hc}}(\lambda)}$

1 : $d \leftarrow\!\$\ \{0,1\}$
2 : $\mathsf{fk} \leftarrow\!\$\ \mathsf{f.KGen}(1^\lambda);\ \mathsf{hk} \leftarrow\!\$\ \mathsf{hc.KGen}(1^\lambda)$
3 : $x \leftarrow\!\$\ \{0,1\}^{\mathsf{f.il}(\lambda)}$
4 : $y \leftarrow \mathsf{f.Eval}(\mathsf{fk}, x)$
5 : $r_0 \leftarrow\!\$\ \{0,1\}^{\mathsf{hc.ol}(\lambda)}$
6 : $r_1 \leftarrow \mathsf{hc.Eval}(\mathsf{hk}, x)$
7 : $d' \leftarrow\!\$\ \mathcal{A}(1^\lambda, \mathsf{fk}, \mathsf{hk}, y, r_d)$
8 : **return** $(d = d')$

Let b denote the hidden bit in the UCE game. If $b = 1$ then oracle HASH implements function H with a uniformly random key. Otherwise, if $b = 0$ then oracle HASH implements a random oracle and value r is set to a uniformly random string. Thus, the pair (S,D) within the UCE game perfectly simulate the hardcore game for adversary $\mathcal{A}$ which establishes the equivalence in Equation (9.1). What remains to show is that source S is unpredictable.

Let P be a predictor, then we construct an inverter $\mathcal{B}$ against the one-way function f as follows. On input an image y and a one-way function key fk, inverter $\mathcal{B}$ samples a uniformly random string $r \leftarrow\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$. It then runs predictor P to obtain a set Q. Finally, inverter $\mathcal{B}$ picks a random value from the set, that is, $x' \leftarrow\!\$\ Q$ which it returns.

We observe that $\mathcal{B}$ perfectly simulates the prediction game $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}$ for predictor P and thus $\mathcal{B}$ is successful with the same probability as P. For this note that game Pred is always in the random oracle setting and the predictor expects r to be a random string. We, thus, have that

$$\mathsf{Adv}^{\text{pred}}_{\mathsf{S},\text{Pred}}(\lambda) \leq |Q| \cdot \mathsf{Adv}^{\text{owf}}_{\mathsf{f},\mathcal{B}}(\lambda)$$

where the factor $|Q|$ stems from inverter $\mathcal{B}$ having to guess which value in Q is the correct value.[10] As by assumption f is one-way, the right-hand side of the equation is negligible, thus, showing that $\mathsf{S} \in \mathcal{S}^{\text{cup}}$, that is, source S is computationally unpredictable. □

9.4.2 Deterministic Public-Key Encryption from UCE

We next show that our Hybrid and Double Encrypt-with-Hash transformation (HD-EwH) from Section 8.5.1 is IND-secure assuming that the random oracle is instantiated with a UCE[$\mathcal{S}^{\text{cup}}$] secure hash function and that the PKE scheme is one-way. The latter may be surprising at first, but we note that the transformation may also be regarded as an extension of the BR93 scheme (see Section 4.3.3) for which, similarly, a trapdoor function is sufficient.

Let us recall the Hybrid and Double Encrypt-with-Hash transformation (HD-EwH) in the keyed random oracle model. For a randomized public-key encryption scheme PKE we define $\mathsf{HD\text{-}EwH}^{\mathsf{RO}}[\mathsf{PKE}]$ as follows. Key generation creates $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$ as well as four keys $\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{gk}_1, \mathsf{gk}_2$ each chosen uniformly at random in $\{0,1\}^{\mathsf{kl}(\lambda)}$. It returns $(\mathsf{sk}, (\mathsf{hk}_1, \mathsf{hk}_2, \mathsf{gk}_1, \mathsf{gk}_2, \mathsf{pk}))$. An encryption of a message m consists of the following three components

$$\mathsf{PKE.Enc}\left(\mathsf{pk}, \mathsf{RO}(\mathsf{hk}_1, \mathsf{pk}\|m); \mathsf{RO}(\mathsf{gk}_1, \mathsf{pk}\|m)\right), \mathsf{PKE.Enc}\left(\mathsf{pk}, \mathsf{RO}(\mathsf{hk}_2, \mathsf{pk}\|m); \mathsf{RO}(\mathsf{gk}_2, \mathsf{pk}\|m)\right),$$
$$m \oplus \mathsf{RO}^2(\mathsf{hk}_1, \mathsf{pk}\|m) \oplus \mathsf{RO}^2(\mathsf{hk}_2, \mathsf{pk}\|m)\ ,$$

where $\mathsf{RO}^2(\mathsf{hk}, x) := \mathsf{RO}(\mathsf{hk}, \mathsf{RO}(\mathsf{hk}, x))$. The decryption algorithm decrypts the asymmetric components of the ciphertext to get $\mathsf{RO}(\mathsf{hk}_1, \mathsf{pk}\|m)$ and $\mathsf{RO}(\mathsf{hk}_2, \mathsf{pk}\|m)$, hashes them under keys hk_1 and hk_2 respectively and xors them to compute the symmetric mask, which it uses to recover the message m.

As the transformation makes use of multiple keys, we need to work in the multiple-key UCE setting, that is, we will show that if RO is implemented by a mUCE[$\mathcal{S}^{\text{cup}}$] secure function H and PKE is one-way secure then $\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}]$ is IND secure. The unpredictability requirement for mUCE[$\mathcal{S}^{\text{cup}}$] is extended to hold for all queries independent of the key. That is, the predictor needs to predict just a single query for any of the keys.

Proposition 9.4. *If* $\mathsf{H} \in$ mUCE[$\mathcal{S}^{\text{cup}}$] *secure and* PKE *is a public-key encryption scheme that is one-way secure and the probability of correctly guessing a public-key as generated by* PKE.KGen *on uniformly random coins is negligible, then* $\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}]$ *is IND secure.*

A simple corollary is that HD-EwH is IND secure in the keyed random oracle model by noting that a keyed random oracle is mUCE[$\mathcal{S}^{\text{cup}}$] secure (thus, proving Proposition 8.5). The latter is obtained by combining the above with Theorem 9.1. Note, however, that the proof yields even more. IND is a multi-stage security game so indifferentiability is not immediately applicable (see Section 9.3). We provide a means to work with indifferentiability in multi-stage settings in [Mit14] but note that a proof via UCEs makes this superfluous as our study of HMAC directly yields that HMAC with an ideal compression function provides a secure instantiation (Theorem 9.2).

In their original paper, Bellare et al. [BHK13:f] list deterministic public-key encryption as one of the main applications of UCEs and they prove that UCE[$\mathcal{S}^{\text{cup}}$] secure functions are sufficient to instantiate the random oracle in the Encrypt-with-Hash transformation (see Chapter 8). We

[10] We note that with access to key fk inverter $\mathcal{B}$ can actually test which of the values in Q is the correct value.

note that the following proof of security for our HD-EwH transformation is inspired by [BHK13:f, Theorem 5.3].

Proof of Proposition 9.4. Assuming the existence of an IND adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the IND security of $\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}]$ we construct a UCE adversary (S, D) as follows. Source S will ask to play with 4 keys. The remainder of the pseudocode of source S and distinguisher D is presented next:

$\mathsf{S}^{\textsc{Hash}}(1^\lambda)$

1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
2 : $d \leftarrow \{0,1\}$
3 : $\mathbf{m}_0, \mathbf{m}_1 \leftarrow\!\$\ \mathcal{A}_1(1^\lambda)$
4 : **for** $i = 1, \ldots |\mathbf{m}_0|$ **do**
5 : $\quad \tau_1 \leftarrow\!\$\ \textsc{Hash}(1, \mathsf{pk} \| m_d[i])$
6 : $\quad \tau_2 \leftarrow\!\$\ \textsc{Hash}(2, \mathsf{pk} \| m_d[i])$
7 : $\quad r_1 \leftarrow\!\$\ \textsc{Hash}(3, \mathsf{pk} \| m_d[i])$
8 : $\quad r_2 \leftarrow\!\$\ \textsc{Hash}(4, \mathsf{pk} \| m_d[i])$
9 : $\quad c_1 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \tau_1; r_1)$
10 : $\quad c_2 \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \tau_2; r_2)$
11 : $\quad c_3 \leftarrow\!\$\ \mathbf{m}_d[i] \oplus \textsc{Hash}(1, \tau_1) \oplus \textsc{Hash}(2, \tau_2)$
12 : $\quad \mathbf{c}[i] \leftarrow (c_1, c_2, c_3)$
13 : $L \leftarrow (\mathsf{pk}, \mathbf{c}, d)$
14 : **return** L

$\mathsf{D}(1^\lambda, \mathbf{hk}, L)$

$(\mathsf{pk}, \mathbf{c}, d) \leftarrow L$
$d' \leftarrow\!\$\ \mathcal{A}_2(1^\lambda, (\mathsf{pk}, \mathbf{hk}), \mathbf{c})$
if $d = d'$ **then**
$\quad$ **return** 1
return 0

The intuition is easily explained. The source runs the first stage of the adversary $\mathcal{A}_1$ to obtain two message vectors and then simulates the creation of ciphertexts (lines 4 to 12). For this it generates a key pair $(\mathsf{pk}, \mathsf{sk})$ and flips a bit d. It then uses public key pk to encrypt message vector $\mathbf{m}_d$ where it uses its oracle HASH for any calls to the hash function. Source S leaks the public key pk, ciphertext vector $\mathbf{c}$ as well as bit d. Distinguisher D gets leakage $(\mathsf{pk}, \mathbf{c}, d)$ and additionally four hash keys, vector $\mathbf{hk}$. It runs the second stage of the IND adversary $\mathcal{A}_2$ on input $(1^\lambda, (\mathsf{pk}, \mathbf{hk}), \mathbf{c})$ to receive a bit d'. If d' and d are the same then it outputs 1 and otherwise it outputs 0.

Let us call b the hidden bit in the UCE game. Then, if $b = 1$, that is, if the HASH oracle implements the actual hash function H we have that source and distinguisher perfectly simulate the IND game for adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. Thus, we have

$$\begin{aligned} \Pr[d = d' \mid b = 1] &= \Pr\left[\mathrm{IND}^{\mathcal{A}_1,\mathcal{A}_2}_{\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}]}(\lambda) = 1\right] \\ &= \frac{1}{2} + \frac{1}{2} \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}],\mathcal{A}}(\lambda) \end{aligned}$$

which we can reorganize as:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}],\mathcal{A}}(\lambda) \leq 2 \cdot \Pr[d = d' \mid b = 1] - 1 \tag{9.2}$$

Further note that we can rewrite the advantage in the UCE setting as

$$\begin{aligned}\mathsf{Adv}^{\mathrm{muce}}_{\mathsf{S},\mathsf{D}}(\lambda) &= 2 \cdot \Pr\left[\mathrm{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1\right] - 1. \\ &= \Pr\left[\mathrm{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1 \;\middle|\; b = 1\right] + \Pr\left[\mathrm{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1 \;\middle|\; b = 0\right] - 1 \\ &= \Pr[d = d' \mid b = 1] - \Pr[d = d' \mid b = 0] \qquad (9.3)\end{aligned}$$

Combining Equations (9.2) and (9.3) and rearranging yields and upper bound for the advantage of adversary $\mathcal{A}$ against the IND security:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}],\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathrm{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda) + 2 \cdot \Pr[d = d' \mid b = 0] - 1$$

What is left is to show that the final term, that is, $2 \cdot \Pr[d = d' \mid b = 0] - 1$ is negligible and that the presented source is computationally unpredictable, that is, source S is in $\mathcal{S}^{\mathrm{cup}}$. Let us first bound probability $\Pr[d = d' \mid b = 0]$.

In case bit b is fixed to 0, oracle Hash implements a random oracle. Thus, unless the adversary guess public-key pk, we can think of values τ_1, τ_2, r_1, r_2 as uniformly random (also note that by assumption all messages are distinct) and hence ciphertexts c_1 and c_2 are ciphertexts for uniformly random values (with uniformly random coins). Furthermore, ciphertext c_3 generated in line 11 can be thought of as a one-time pad since both $\textsc{Hash}(1, \tau_1)$ and $\textsc{Hash}(2, \tau_2)$ are uniformly random values. It follows that the leakage L consists of encryptions of random strings as well as random strings and, in particular, leakage L is independent of any of the messages. We, thus, have that

$$\Pr[d = d' \mid b = 0] \leq \tfrac{1}{2} + \nu(\lambda) \cdot \mathsf{Adv}^{\mathrm{guess-pk}}_{\mathsf{PKE},\mathcal{B}}(\lambda).$$

Here ν is a polynomial bounding the number of plaintexts output by adversary $\mathcal{A}_1$ and $\mathsf{Adv}^{\mathrm{guess-pk}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$ denotes the advantage of an adversary guessing the public key as generated by $\mathsf{PKE.KGen}$ when run on uniformly random coins. It follows that the IND-advantage of any adversary is directly upper bounded by the mUCE advantage as:

$$\mathsf{Adv}^{\mathrm{ind}}_{\mathsf{HD\text{-}EwH}^{\mathsf{H}}[\mathsf{PKE}],\mathcal{A}}(\lambda) \leq 2 \cdot \mathsf{Adv}^{\mathrm{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda) + 2\nu(\lambda) \cdot \mathsf{Adv}^{\mathrm{guess-pk}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$$

To complete the proof we need to argue that source S is computationally unpredictable, that is, source $\mathsf{S} \in \mathcal{S}^{\mathrm{cup}}$. For this note that the unpredictability game Pred is always relative to a random oracle. By assumption adversary $\mathcal{A}$ generates message vectors such that each message has super-logarithmic min-entropy meaning that queries occurring from lines 5 to line 8 cannot be guessed by any (even an unbounded predictor). The only information about message m is contained in the third ciphertext component c_3. As the prediction game is in the random oracle world, c_3, computed as

$$c_3 \leftarrow\!\$\; m \oplus \textsc{Hash}(1, \tau_1) \oplus \textsc{Hash}(2, \tau_2),$$

information theoretically hides m unless both random oracle values for $\textsc{Hash}(1, \tau_1)$ and $\textsc{Hash}(2, \tau_2)$ are known. It remains to argue that τ_1 and τ_2 cannot be recovered by any efficient predictor P even though the leakage information theoretically contains them (observe that c_1 is an encryption of τ_1 and c_2 is an encryption of τ_2). Noting, once more, that, unless adversary $\mathcal{A}_1$ guesses key pk,

values τ_1 and τ_2 as well as r_1 and r_2 are uniformly random values we obtain that a predictor P that successfully recovers either τ_1 or τ_2 can be turned into an efficient inverter against the public-key encryption scheme PKE. It follows that for any predictor P

$$\Pr\left[\mathrm{Pred}_{\mathsf{S}}^{\mathsf{P}}(1^\lambda) = 1\right] \leq \frac{q(\lambda)}{2^{\mathsf{H.il}(\lambda)}} + \mathsf{Adv}^{\mathrm{inv}}_{\mathsf{PKE},\mathcal{A}}(\lambda) + (q(\lambda) + \nu(\lambda)) \cdot \mathsf{Adv}^{\mathrm{guess-pk}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$$

where $q(\lambda)$ denotes an upper bound on the number of guesses by P and $\nu(\lambda)$ denotes the number of plaintexts as output by adversary $\mathcal{A}$. As both P and $\mathcal{A}_1$ can try to guess the public key this accounts for the last summand. Adversary $\mathcal{A}$ is the induced inversion algorithm that simulates the leakage of source S for P using its challenge as either c_1 or c_2. Consequently, source S is computationally unpredictable (note that an unbounded predictor can break the one-wayness of the PKE scheme) and, thus, $\mathsf{S} \in \mathcal{S}^{\mathrm{cup}}$ which concludes the proof. □

In the above proof we require the predictor to be efficient as otherwise it can invert the PKE components of the ciphertext and learn HASH queries. As noted by Bellare and Hoang [BH15] we can use lossy trapdoor functions [PW08, FGK+13] to strengthen some results and make sources unpredictable even against unbounded predictors. (We note that they do not consider the $\mathsf{HD\text{-}EwH}$ but their techniques similarly apply here.) For this we require that the PKE scheme is not one-way (i.e., a trapdoor function) but that it has the security of a lossy trapdoor function. A lossy trapdoor function has the property that there exist two key generation algorithms, one for the generation of honest keys and one for the generation of lossy keys. While the no computationally bounded distinguisher can distinguish between an honest evaluation key and a lossy evaluation key, lossy evaluation keys have the property that they do result in an injective function. In other words, if our above scheme would be instantiated with a lossy trapdoor function (LTDF) and a lossy key one could no longer decrypt. We can use this fact to change our UCE security proof in one point to obtain a stronger result. If we let our source generate lossy trapdoor keys instead of honest trapdoor keys then the analysis of the UCE security remains identical but for an additional loss of a distinguisher against the security of the LTDF. What we have gained, however, is that the analysis of unpredictability of the source can now handle even unbounded predictors as values τ_1 and τ_2 are no longer information theoretically contained in the leakage due to the lossyness. The consequence of this is that we can obtain a security proof also for a weaker notion of UCE security, namely UCE[$\mathcal{S}^{\mathrm{sup}}$], UCE security against statistically unpredictable sources (which we introduce shortly). Following the techniques of Bellare and Hoang [BH15] we, thus, obtain

Proposition 9.5 (informal [BH15])**.** *If lossy trapdoor functions and* $\mathsf{H} \in \mathrm{mUCE}[\mathcal{S}^{\mathrm{sup}}]$*-secure functions exist, then IND-secure deterministic public-key encryption exists.*

As mentioned, proofs using UCEs can be much cleaner and easier to follow than proofs in the random oracle model. The above is a good example (a corresponding random oracle proof (for a slightly weaker statement) can be found in [BFM15]). Similarly, the proof BHK give for showing that UCE[$\mathcal{S}^{\mathrm{cup}}$] is sufficient to constructing deterministic public-key encryption [BHK13:f, Theorem 5.3] is simpler than the original proof given in the random oracle model [BBO07]. Thus, UCEs and, in particular, UCE[$\mathcal{S}^{\mathrm{cup}}$] provide an interesting alternative to proofs in the random oracle model: as seen, proofs via UCEs can be simpler and cleaner and furthermore are stronger than proofs in the random oracle model. For the latter note that we have shown that HMAC is mUCE[$\mathcal{S}^{\mathrm{cup}}$] secure

which gives us a security proof in the ideal compression-function model for free which, at least for multi-stage games, may be hard to obtain otherwise [RSS11, DGHM13, BBM13, Mit14].

9.5 UCE1 and Indistinguishability Obfuscation cannot Co-Exist

As mentioned, UCE[$\mathcal{S}^{\text{cup}}$] seems to capture a universal property of random oracles which makes it applicable in a wide variety of applications. Of particular interest to us is the result that UCE[$\mathcal{S}^{\text{cup}}$] is sufficient to obtain deterministic public-key encryption. BHK show this via instantiating the random oracle in the Encrypt-with-Hash transform that we have encountered in Chapter 8 and in the previous section we have seen that it is also sufficient to instantiate the random oracle in our HD-EwH transformation (Section 8.5.1).

Combined with Theorem 8.1 (resp. with the uninstantiability result for HD-EwH transformation from Section 8.5.1), which states that no standard model hash function can instantiate Encrypt-with-Hash if indistinguishability obfuscation exists, we directly get the following impossibility result for UCEs:

Theorem 9.6. *If indistinguishability obfuscation exists for all circuits in* P/poly*, then no standard model hash function can be* UCE[$\mathcal{S}^{\text{cup}}$] *secure. This holds even for sources making only a single query.*

An immediate corollary is that also UCE[$\mathcal{S}^{\text{crs}}$] and indistinguishability obfuscation are mutually exclusive as any computationally unpredictable source is also reset secure. Further note that for this negative result, the weaker version of Theorem 8.1 is sufficient as we only need to argue that for any potential UCE function H there exists a PKE scheme such that $\mathsf{EwH}[\mathsf{H}, \mathsf{PKE}]$ is not IND secure.

9.5.1 Indistinguishability Obfuscation vs. UCE1 Without Detours

Attacking UCEs indirectly via our previous uninstantiability result doesn't quite capture what exactly the role of obfuscation is in the attack. We can, however, give a much simpler and direct attack: The source picks a random point x, and queries it to oracle Hash to obtain value y. It then prepares an iO of the Boolean circuit $(\mathsf{H}(\cdot, x) = y)$, and leaks it to the distinguisher as leakage L. The distinguisher, that gets as additional input hash key hk, now plugs hk into this obfuscated circuit and returns whatever the circuit outputs. It is easy to see that the distinguisher recovers the challenge bit correctly with an overwhelming probability. What is less clear, however, is whether or not the source is unpredictable. Recall that the unpredictability game operates with respect to a random oracle. Let us now assume, for simplicity, that $|\mathsf{hk}| < |y|/2$ (we will not need to rely on this assumption in our full attack). For any x, there are at most $2^{|\mathsf{hk}|}$ possible values for $\mathsf{H}(\mathsf{hk}, x)$, and a random y would be one of them with probability at most $2^{|\mathsf{hk}|}/2^{|y|} < 2^{-|y|/2}$, which is negligible. Consequently, the obfuscated circuit implements the constant *zero* function with overwhelming probability. This allows us to apply the security of the obfuscator to conclude the attack: the obfuscated circuit does not leak any more information about x than the constant zero function would. Since x was chosen randomly it follows that it remains hidden from the view of any efficient predictor. Let us next make this intuition formal.

Proof of Theorem 9.6. Let H be a UCE[$\mathcal{S}^{\text{cup}}$]-secure hash function family and let us assume for now that $\mathsf{H.ol}(\lambda) \geq 2 \cdot \mathsf{H.kl}(\lambda)$. That is, we assume that the output length of the hash function is at least twice the size of a hash key. (We will be dropping this condition shortly.) Define a source S which generates a random value $x \leftarrow\!\!\$\, \{0,1\}^{\mathsf{H.il}(\lambda)}$ and computes $y \leftarrow\!\!\$\, \textsc{Hash}(x)$. It then constructs the Boolean circuit

$$\frac{C[x,y](\mathsf{hk})}{y' \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)}$$
$$\textbf{return } (y = y')$$

constructs an indistinguishability obfuscation $\overline{C} \leftarrow\!\!\$\, \mathsf{iO}(C[x,y])$ and sets $L \leftarrow \overline{C}$. Distinguisher D recovers circuit $\overline{C}$ from the leakage L and computes $b' \leftarrow \overline{C}(\mathsf{hk})$ using the given hash key hk which it returns. By construction $\overline{C}(\mathsf{hk}) = C[x,y](\mathsf{hk})$ and, thus, adversary (S, D) has advantage $1 - 2^{-\mathsf{H.ol}(\lambda)}$: When the source is run with oracle access to $\mathsf{H.Eval}(1^\lambda, \mathsf{hk}, \cdot)$ the circuit always returns 1. When S interacts with a random oracle, y coincides with $\mathsf{H.Eval}(1^\lambda, \mathsf{hk}, x)$ with probability $2^{-\mathsf{H.ol}(\lambda)}$. It remains to show that the adapted source S is unpredictable.

Unpredictability. Let P be a predictor in the $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ game. We bound the success probability of P based on the security of the indistinguishability obfuscator iO via a sequence of two games as follows.

$\mathsf{Game}_1(\lambda)$**:** The first game is identical to the computational unpredictability game $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ (see Figure 9.2 on page 186).

iO

$\mathsf{Game}_2(\lambda)$**:** is similar to the previous game except that S now leaks $\mathsf{iO}(\mathsf{Z}_{|C[x,y]|}(\cdot))$, where $\mathsf{Z}_{|C[x,y]|}(\cdot)$ denotes the constant zero circuit padded to the length of Boolean circuit $C[x,y]$.

We consider the distinguishing advantage of an adversary between the two games and then show that in Game_2 the prediction advantage is negligible.

$\mathsf{Game}_1(\lambda)$ **to** $\mathsf{Game}_2(\lambda)$**.** We bound the change in P's advantage from Game_1 to Game_2 down to the security of the indistinguishability obfuscator. For this note that the sources in games Game_1 and Game_2 induce an equivalence sampler in the indistinguishability obfuscation game. We construct sampler Sam to output the circuits $C[x,y]$, for a uniformly random $x \in \{0,1\}^{\mathsf{H.il}(\lambda)}$ and a uniformly random y in $\{0,1\}^{\mathsf{H.ol}(\lambda)}$, as well as the constant zero circuit $\mathsf{Z}_{|C[x,y]|}$. As auxiliary information the sampler leaks pair (x,y). As distinguisher, we construct an algorithm D that on input an obfuscated circuit $\overline{C}$ runs predictor P on input $\overline{C}$ and which simulates the random oracle for P by answering queries $x' \neq x$ with a uniformly random value and a query $x' = x$ with value y. When predictor P stops and outputs set Q' we check whether $x \in Q'$ and if so output 1 and otherwise output 0.

If $\overline{C}$ is an obfuscation of circuit $C[x,y]$, then we perfectly simulate Game_1 and if it is an obfuscation of the constant zero circuit $\mathsf{Z}_{|C[x,y]|}$, then we perfectly simulate Game_2. It, thus, remains to show that sampler Sam is an equivalence circuit sampler. For this we first note that the two circuits output by Sam always have the same size. We next argue that circuit $C[x,y]$ is 0 on all inputs with overwhelming probability, more precisely,

$$\Pr_{x,y}[\exists \mathsf{hk} : C[x,y](\mathsf{hk}) \neq 0] \leq \frac{1}{2^{\mathsf{H.ol}(\lambda)/2}}.$$

The inequality holds from the union bound and the fact that $\mathsf{H.ol}(\lambda) \geq 2 \cdot \mathsf{H.kl}(\lambda)$. For this note there are at most $2^{\mathsf{H.kl}(\lambda)}$ possible values for $\mathsf{H.Eval}(\mathsf{hk}, x)$. A random y would be one of the image values with probability at most $2^{\mathsf{H.kl}(\lambda)} 2^{-\mathsf{H.ol}(\lambda)}$, which by assumption is at most $2^{-\mathsf{H.ol}(\lambda)/2}$. In summary, we have that for any PPT predictor Pred the probability of predicting x in Game_2 is almost as high as in Game_1. That is,

$$|\Pr[\mathsf{Game}_2(\lambda) = 1] - \Pr[\mathsf{Game}_1(\lambda) = 1]| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathsf{Sam},\mathsf{D}}(\lambda) + \frac{1}{2^{\mathsf{H.ol}(\lambda)/2}}.$$

Finally, in game $\mathsf{Game}_2(\lambda)$ the leakage contains no information on value x and hence the probability of guessing x is upper bounded by $q_{\mathsf{S}} \cdot |Q(\lambda)| / 2^{\mathsf{H.il}(\lambda)}$, where q_{S} denotes the number of queries by source S and $Q(\lambda)$ denotes the set output by predictor P. It follows that the source is computationally unpredictable.

Dropping the length requirement. It remains to argue how we can drop the requirement on the size of hash keys. For this note that we can choose a t such that $t \geq 2 \cdot \lceil \mathsf{H.kl}(\lambda)/\mathsf{H.ol}(\lambda) \rceil$ and let the source leak an obfuscation of the circuit $(\mathsf{H.Eval}(1^\lambda, \cdot, x_1) = y_1 \wedge \cdots \wedge \mathsf{H.Eval}(1^\lambda, \cdot, x_t) = y_t)$. That is, instead of a single pair x, y the source uses multiple pairs (x_i, y_i) which it all encodes into the circuit together. □

9.5.2 On the Number of Source Queries

In the above proof, we relied on the source being able to make multiple queries to its hash oracle. On the other hand, if we take the detour via the uninstantiability result for Encrypt-with-Hash, we see that making multiple queries is not necessary, and that the result can be strengthened to rule out even sources that are restricted to making a single query. The reason is that for the uninstantiability result for EwH we combine the hash function with a pseudorandom generator.[11] To adapt our above result we would change source S to leak an obfuscation of circuit

$$\frac{C[x, s \leftarrow \mathsf{PRG}(y)](\mathsf{hk})}{s' \leftarrow \mathsf{PRG}\left(\mathsf{H.Eval}(\mathsf{hk}, x)\right)}$$
$$\textbf{return } (s = s')$$

The circuit has value s, computed as $\mathsf{PRG}(y)$, hard-coded instead of y directly. Note that this requires that the output length of hash function H is super-logarithmic in the security parameter as otherwise we cannot use the security of the pseudorandom generator.

9.6 UCE Assumptions

In their original work [BHK13:p], BHK introduce two concrete UCE notions: UCE with respect to computationally unpredictable sources (UCE[$\mathcal{S}^{\text{cup}}$]) and UCE with respect to computationally reset-secure sources (UCE[$\mathcal{S}^{\text{crs}}$]). As seen in the previous section, both notions are mutually exclusive with

[11] We note that the possibility of strengthening our above simplified result via the application of a pseudorandom generator was pointed out by Bellare, Hoang, and Keelveedhi [BFM13a].

the existence of indistinguishability obfuscation for all circuits and subsequently new UCE notions have been proposed to work around this impossibility result. In this section we discuss the most prominent proposals.

9.6.1 Statistically Secure Sources

Our iO attack immediately gives rise to the following question: can UCE[$\mathcal{S}^{\text{cup}}$] and UCE[$\mathcal{S}^{\text{crs}}$] be somehow patched so that they avoid the attack while maintaining (part of) their wide applicability? In our attack on UCEs our source leaked an indistinguishability obfuscation of a breaker circuit

$$\frac{C[x,y](\mathsf{hk})}{y' \leftarrow \mathsf{H.Eval}(\mathsf{hk},x)}$$
$$\textbf{return } (y = y')$$

where x was chosen uniformly at random and y was computed as $y \leftarrow\!\!\$\, \textsc{Hash}(x)$.

As we have seen in Section 5.5.2 (Theorem 5.7) the security guarantee of the indistinguishability obfuscator is only computational (unless the polynomial hierarchy collapses). Consequently, the attack can be directly ruled out by demanding the source to be *statistically* unpredictable, i.e., by letting a potential predictor run in unbounded time (but still impose polynomial query complexity). In that way, any leaked indistinguishability obfuscation is as good as leaking the contained secrets in the clear which seems to rule out the usefulness of using iO. More formally, we say a source S is *statistically unpredictable* if the advantage of any (possibly unbounded) predictor P with polynomial query complexity in the $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ game shown in Figure 9.3 (page 187; left) is negligible. Statistical reset security can be defined analogously, where we let the reset distinguisher run in unbounded time and only place a polynomial bound on the number of its queries. We denote statistically unpredictable sources by $\mathcal{S}^{\text{sup}}$ and statistically reset-secure sources by $\mathcal{S}^{\text{srs}}$; this yields the UCE notions UCE[$\mathcal{S}^{\text{sup}}$] (UCEs with respect to statistically unpredictable sources) and UCE[$\mathcal{S}^{\text{srs}}$] (UCEs with respect to statistically reset-secure sources).

The above definition, in turn, leads to the following question: Are there any application scenarios which only rely on this weaker property? Recall that the unpredictability game is always defined with respect to a random oracle, and hence statistical unpredictability may be (non-trivially) achievable. Indeed, consider a source which samples a random point x, queries it to its oracle, and leaks the result to the distinguisher. It is easy to see that this source is statistically unpredictable as a random oracle is one-way against unbounded adversaries if restricted to make only polynomially many oracle queries. Indeed, many of the cryptosystems considered by BHK admit security proofs with sources that essentially take this simple form [BHK13:p, BHK13:f]. These include the applications of key-dependent message secure and related-key secure symmetric encryption schemes, correlated-input secure hash functions, point obfuscation, the proof storage scheme and the garbling schemes.

Remark. BHK in a revision of their paper [BHK13:1] independently suggested statistical notions of unpredictability and reset-security and recast the proofs of the above schemes in terms of the new statistical UCE notions. We refer to [BHK13:1] for details on the applications that can be salvaged with statistical unpredictability UCE[$\mathcal{S}^{\text{sup}}$] (resp. statistical reset-security UCE[$\mathcal{S}^{\text{srs}}$]).

Salvaging remaining applications. For the hardcore predicate, BR93 encryption, EwH, MLE and OAEP application scenarios discussed in [BHK13:p, BHK13:f], the leakage contains auxiliary information related to a query x that only computationally hides x (e.g., it might contain a one-way image $f(x)$, or an encryption of x). Consequently, an unbounded predictor might well be able to guess the point x, and in these cases our statistical patch is no longer useful. Despite this, we observe that UCE-secure hash functions with regard to statistical unpredictability are hardcore for highly non-injective one-way functions. (The proof is essentially equivalent to that in [BHK13:f] and relies on the fact that any (even an unbounded) predictor cannot recover the *exact* query if the preimage space is super-polynomially large.)

We note that for message-locked encryption (MLE; see also Section 8.6.1) BHK later showed how this can be achieved down to UCE[$\mathcal{S}^{\text{sup}}$], that is, down to a statistically unpredictable source [BHK13:2]. In the computational unpredictable scenario, BHK showed that the convergent encryption transformation of [DAB+02, BKR13] can be instantiated with any sufficiently strong symmetric encryption scheme. The source used in the proof, similarly to the D-PKE case needs to leak ciphertexts of HASH oracle queries. Thus, depending on the symmetric encryption scheme used the source may become predictable, if the predictor is allowed to run in unbounded time. For a statistically secure symmetric encryption scheme (i.e., a one-time pad), on the other hand, the same proof still works while the source can be shown to be statistically unpredictable. Thus, when restricting the symmetric encryption scheme to be a one-time pad, one can show that a UCE[$\mathcal{S}^{\text{sup}}$] secure function can instantiate the random oracle in the convergent encryption transformation [BHK13:2]. Note that this does not contradict our uninstantiability result (Theorem 8.6) for the convergent encryption scheme as there we need to control the symmetric encryption scheme. That is, there we show that there exists a symmetric encryption scheme for which the transform does not work.

Finally, in a recent and exciting work, Bellare and Hoang [BH15] show how to also obtain deterministic public-key encryption from UCEs with respect to statistically unpredictable sources and lossy trapdoor functions [PW08, FGK+13]. In essence, if a source queries the result of a lossy trapdoor function computation to its oracle then the query can remain hidden given that the trapdoor function is operated in lossy mode. As the UCE game only needs to be secure against computationally bounded distinguishers sources can exploit this fact as long as the inversion key is not needed (which is the case in the security definition of a deterministic public-key encryption scheme). This trick can be used, for example, with the HD-EwH transformation (see also Proposition 9.5 and its preceding paragraph) and Bellare and Hoang present additional D-PKE schemes based on UCE[$\mathcal{S}^{\text{sup}}$] and lossy trapdoor functions.

Additional Restrictions. Various different restrictions have been proposed in order to salvage the remaining applications. We present an overview of the applications and newly proposed UCE notions in Table 9.2 and in the next sections provide an overview over the various restrictions.

9.6.2 Split and Bounded Parallel Sources

We discussed that statistical versions of unpredictability and reset-security allow to salvage a large number but not all of the original applications presented by BHK. In order to salvage the remaining applications, BHK introduced two additional restrictions [BHK13:1] which were meant to be used in combination with either unpredictable or reset-secure sources: in particular they were designed

Notion	Original UCE	Restricted UCE	Construction/Contention
HC	UCE1	UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]	Construction based on iO and point-function obfuscation presented in Chapter 11.
BR93	UCE1	UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]	
CIH	UCE1	UCE[$\mathcal{S}^{\text{s-sup}}$]	Construction based on iO and point-function obfuscation achieving q-query security presented in Chapter 11. Full security can be obtained with SuPA assumption (Chapter 13).
STORE	UCE1	UCE[$\mathcal{S}^{\text{s-sup}}$]	
IMMU	UCE1	UCE[$\mathcal{S}^{\text{s-sup}}$]	
MLE	UCE1	UCE[$\mathcal{S}^{\text{s-sup}}$]	
D-PKE	UCE1	UCE[$\mathcal{S}^{\text{s-sup}}$]	
MLE	UCE1	UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}$]	Mutually exclusive with indistinguishability obfuscation (Chapter 10).
D-PKE (EwH)	UCE1	UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}$]	
		UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$]	For any hash function H there exist schemes PKE such that $\mathsf{H} \notin$ UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$]. Also see Chapter 8.
KDM	mUCE1	mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{1-query}}$]	Construction based on indistinguishability obfuscation and point-function obfuscation presented in Chapter 11.
RKA	mUCE1	mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{1-query}}$]	
PFOB	mUCE1	mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{1-query}}$]	
GB	UCE2	UCE[$\mathcal{S}^{\text{srs}}$]	Existence is still open.
OAEP	UCE1/2	UCE[$\mathcal{S}^{\text{crs}} \cap \mathcal{S}_{\mathsf{TF},\ell_1,\ell_2,\ell_3}$]	

Table 9.2: The table lists new UCE notions for the various applications that were proposed to salvage UCEs in response to our impossibility result (Theorem 9.6). On the right we present whether or not standard model constructions for the restricted UCE notions exist or whether further impossibility results for the restricted notions can be shown.

to be used in combination with the *computational* variants. The idea, in both cases, is to restrict the source structurally. That is, the source needs to be able to compute its leakage in a specific way which should be such that the iO-attack can no longer be mounted. The first observation made by BHK is that in order to run the iO attack a source needs to see both a query and the corresponding answer for a HASH oracle query. This observation lead to the definition of so-called *split sources* which restrict sources to be composed of two parts S_0 and S_1 which both contribute to leakage L but such that S_0 defines a list of HASH queries (but does not see the answers) and S_1 gets to see the oracle answers (but not the list of queries). Note that the source presented in our iO attack needs to see both x and y in order to construct an obfuscation of circuit $C[x, y]$ and by restricting sources to be *split* it seems that such an obfuscation cannot be leaked.

The second observation made by BHK is that obfuscation is an expensive operation in terms of computation power. On the other hand, if one looks at the sources used within proofs, then these appear to be much more efficient. As example take the source from the proof of Proposition 9.4. The source obtains a vector of messages from the underlying adversary and then the remaining step is to encrypt each of the messages. In particular, the second part of the source can be parallelized, that is, each message can be encrypted in parallel. For such sources, BHK defined a restriction called *bounded parallel sources* which consist of a first stage which may not access the HASH oracle, but which is otherwise unrestricted, and a second stage which is restricted computationally and which needs to be executed in parallel. Again, considering the source in our attack there is no stage that can be

Prl SOURCE $\mathsf{S}^{\text{HASH}}(1^\lambda)$

$(L_0, \mathbf{L}') \leftarrow\!\$\ \mathsf{S}_0(1^\lambda)$
for $i = 1, \ldots, \left|\mathbf{L}'\right|$ **do** $\mathbf{L}[i] \leftarrow\!\$\ \mathsf{S}_1^{\text{HASH}}(1^\lambda, \mathbf{L}'[i])$
$L \leftarrow (L_0, \mathbf{L})$
return L

Splt SOURCE $\mathsf{S}^{\text{HASH}}(1^\lambda)$

$(L_0, \mathbf{x}) \leftarrow\!\$\ \mathsf{S}_0(1^\lambda)$
for $i = 1, \ldots, |\mathbf{x}|$ **do** $\mathbf{y}[i] \leftarrow\!\$\ \text{HASH}(\mathbf{x}[i])$
$L_1 \leftarrow\!\$\ \mathsf{S}_1(1^\lambda, \mathbf{y}); L \leftarrow (L_0, L_1)$
return L

Figure 9.7: The parallel source $\mathsf{S} = \text{Prl}[S_0, S_1]$ on the left and the split source $\mathsf{S} = \text{Splt}[S_0, S_1]$ on the right as defined in the updated version of [BHK13:1]. In both cases the source consists of two parts S_0 and S_1 that jointly generate leakage L. For split sources neither part gets direct oracle access to HASH. For parallel sources additional restrictions on the runtime and the number of queries of S_1, and the length of leakage L_0 are imposed. Note that the invocations of S_1 are parallelizable and independent of one another.

parallelized, and indeed, the stage that requires access to oracle HASH seems to be very inefficient, since creating an indistinguishability obfuscation is an expensive operation, at least, intuitively much more so than, say, encrypting a message. We next discuss both restrictions in greater detail.

Split Sources

A source S is called split source, denoted by $\mathsf{S} \in \mathcal{S}^{\text{splt}}$ if it can be decomposed into two algorithms S_0 and S_1 such that neither part gets direct access to oracle HASH. We give the pseudocode of split sources in Figure 9.7 on the right. In a first step algorithm S_0 outputs a leakage string L_0 together with a vector $\mathbf{x}$. Then, each of the entries in $\mathbf{x}$ is queried to HASH and the results stored in vector $\mathbf{y}$. Finally, the second algorithm S_1 is run on vector $\mathbf{y}$ to produce the second part of the leakage L_1.

BHK show that with split sources in combination with *computationally* unpredictable sources one can recover universal hardcore functions, that is, BHK show that any UCE$[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}]$ is a universal hardcore function [BHK13:1]. For this note, that the source in Theorem 9.3 is *split*.

There is one intricacy with split sources that we discovered when discussing strong unpredictability (a notion that we introduce shortly) with Mihir Bellare [BM14b]. S_0 outputs a list $\mathbf{x}$ that is then used to generate the input to S_1 as

$$\textbf{for } i = 1, \ldots, |\mathbf{x}| \quad \textbf{do } \mathbf{y}[i] \leftarrow\!\$\ \text{HASH}(\mathbf{x}[i])$$

As $\mathbf{x}$ is a *list* it allows for duplicate values. This, however, means that S_0 can communicate arbitrary values to S_1 by encoding them using duplicates (e.g., using the two identical values to encode a 0 and two different values to encode a 1). The result is that the iO attack still applies to split sources. A simple solution to recover split sources is to disallow duplicates in vector $\mathbf{x}$. We note that when considering only single-query split sources, which suffice for the only known application (universal hardcore functions) then both formulations are equivalent.

Remark. Building on our negative result for MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] (Theorem 7.3) Bellare, Stepanovs, and Tessaro have recently shown that UCE[$\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}}$] is mutually exclusive with indistinguishability obfuscation [BST15]. For this, they construct an MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] secure function from a UCE[$\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}}$] function which together with Theorem 7.3 yields the result. Note, that this does not rule out hash functions that are UCE[$\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}}$]-secure against sources that make only a single query, which are sufficient for the application of universal hardcore functions (cf. Theorem 9.3). We discuss their result in greater detail in Section 10.3.

Bounded Parallel Sources

A second restricted source class that BHK introduce in order to recover the deterministic public-key encryption (D-PKE via EwH), message-locked encryption (MLE), and OAEP applications is that of *bounded parallel sources* [BHK13:1]. In parallel sources the source splits into two parts S_0 and S_1 as follows. The first part of the source S_0 does not get oracle access to HASH and outputs some preliminary leakage L_0 and a vector $\mathbf{L}'$ of arbitrary bit strings. For each entry in $\mathbf{L}'$ an independent instance of the second part of the source S_1 is run. This can be done in parallel as the several invocations do not share any coins or state. Instance i of S_1 is given $\mathbf{L}'[i]$ as input which then produces leakage $\mathbf{L}[i]$. As opposed to S_0, the second part S_1 of parallel sources has oracle access to HASH. The final leakage of the source $\mathsf{S} := \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ is set to be $L := (L_0, \mathbf{L})$. The details of a parallel source $\mathsf{S} = \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ are given in Figure 9.7 on the left.

Without any further restrictions, parallel sources are as powerful as regular sources: simply ignore S_0 and let a single S_1 generate the entire leakage. Thus, in order to circumvent the iO attack, further restrictions are necessary. To this end, BHK restrict the resources of S_0 and S_1 via polynomials τ, σ, and q as follows: (1) the running time (circuit size) of each invocation of S_1 is at most $\tau(\cdot)$; (2) each invocation of S_1 makes at most $q(\cdot)$ oracle queries; and (3) the length of initial leakage L_0 output by S_0 is at most $\sigma(\cdot)$. BHK then consider the class $\mathcal{S}^{\text{prl}}_{\tau,\sigma,q}$ consisting of all parallel sources satisfying these bounds, and define UCE for computationally unpredictable, bounded parallel sources by considering UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}$].

When appropriately choosing τ, σ, and q, BHK show that one can recover the remaining applications: deterministic public-key encryption (D-PKE), message-locked encryption (MLE), and OAEP. For this, value τ must be chosen in accordance to the construction that one wants to prove secure. Let us consider the Encrypt-with-Hash transformation. BHK show the following result

Theorem 9.7 ([BHK13:1], Theorem 5.3)**.** *Let* PKE *be an IND-CPA secure randomized encryption scheme and let* H *be a hash function with appropriate input and output lengths. Let*

$$\tau(\lambda) := \mathcal{O}(\mathsf{time}_{\mathsf{PKE.Enc}}(\lambda) + \mathsf{PKE.il}(\lambda) + \mathsf{PKE.rl}(\lambda) + \mathsf{PKE.kl}(\lambda))\,.$$

Let further $\sigma(\lambda) := \mathcal{O}(\mathsf{PKE.kl}(\lambda))$ *and let* $q = 1$*. If* $\mathsf{H} \in \text{UCE}[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}]$ *then* $\mathsf{EwH}[\mathsf{H}, \mathsf{PKE}]$ *is IND secure.*

An important difference to the previous version of their theorem is that it is now no longer necessary for a hash function H to instantiate the random oracle in Encrypt-with-Hash for all public-key encryption schemes PKE, but only for schemes within bounds σ and τ. We may ask how this goes together with our uninstantiability result for Encrypt-with-Hash? In Theorem 8.1 we show that

for any polynomial p there exists a PKE scheme such that no hash function of length at most p can securely instantiate the random oracle in EwH for PKE. The reason is that the PKE scheme that we construct in Theorem 8.1 uses indistinguishability obfuscation as part of its encryption algorithm. Thus, an encryption operation is "as slow" as an obfuscation operation and, consequently, no bound τ exists that will prevent our initial iO attack for that particular PKE scheme. In other words, there can be no hash function H with the bounds as required in Theorem 9.7 for that particular PKE scheme. However, there could be PKE schemes for which suitable bounds exist.

Jumping ahead, we note that we can extend our basic iO attack to also cover bounded parallel sources. That is, we show that our attack can be adapted to fit the restrictions of bounded parallel sources for bounds $q \neq 0$, $\sigma \geq 0$ and $\tau \in \Omega(\lambda)$. This rules out any practical application of bounded parallel sources. We present the extended attack in detail in Chapter 10.

9.6.3 Bounded Queries, Runtime or Leakage

Bounded parallel sources restrict the runtime of parts of the source and in addition restrict the number of queries of each parallelizable part. Instead of applying such restrictions only on parts of the source, we may restrict the entire source. For example, we can consider sources that make only a single query. We call such sources single-query sources and denote the corresponding source class by $\mathcal{S}^{\text{1-query}}$. We also consider a relaxed notion to allow for polynomially bounded number of queries for some polynomial $\mathsf{q} := \mathsf{q}(\lambda)$. We call the corresponding sources q-query sources and denote their source class by $\mathcal{S}^{\mathsf{q}\text{-query}}$. We note that such query restrictions were also proposed by BHK in [BHK13:1].

Similarly to restricting the number of queries a source can make, we can restrict the source's runtime. Again the difference to bounded parallel sources is that the restriction does not only apply to a specific part of the source, but to the entire source. Such computationally bounded sources are introduced by Matsuda and Hanaoka who show how to use UCEs to construct CCA-secure encryption [MH14b].

Finally, we note that one way around our iO attack is to restrict the size of the leakage that source S may generate. If this is less than the size of circuit $C[x, y]$ (or rather less than an obfuscation of the circuit) then our attack no longer works.

9.6.4 Fine-Tuned Sources

We already mentioned that our iO attack can be extended to cover also bounded parallel sources. We published this result at Crypto 2014 [BFM14a] which in turn lead to an update of the UCE framework description by BHK [BHK13:2]. With their update, BHK removed bounded parallel sources and introduced new UCE notions to recover the applications that previously depended on bounded parallel sources. As already mentioned BHK recover message-locked encryption by fixing the symmetric encryption scheme in the convergent encryption transformation which allows them to switch to a statistically unpredictable source. For D-PKE and OAEP, on the other hand, BHK introduce new UCE notions that we refer to as *fine-tuned sources*. The new UCE notions introduce structural restrictions, similarly to split and bounded parallel sources. These restrictions are, however, much more detailed: basically BHK took the sources from the original proofs for UCE1 [BHK13:f] and parameterized the source with the adversary that the source needs to run and the scheme the source needs to evaluate, yielding a source class. Let us exemplify this for the case of D-PKE.

As discussed, BHK show that UCE[$\mathcal{S}^{\text{cup}}$] is sufficient to instantiate the random oracle in the Encrypt-with-Hash transformation for any IND-CPA secure public-key encryption scheme PKE [BHK13:f, Theorem 5.3]. In the proof of the theorem they give a reduction from an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ breaking IND security to a pair (S, D) breaking UCE[$\mathcal{S}^{\text{cup}}$] security. The source S that they construct is:

$\underline{\mathsf{S}^{\text{HASH}}(1^\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
$d \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}_1(1^\lambda)$
for $i = 1, \ldots, |\mathbf{m}_0|$ **do**
  $r \leftarrow\!\!\$\ \text{HASH}(\mathsf{pk} \| \mathbf{m}_d[i])$
  $\mathbf{c}[i] \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathbf{m}_d[i]; r)$
$L \leftarrow (\mathsf{pk}, d, \mathbf{c})$
return L

The UCE notion that BHK introduce in their updated paper [BHK13:2] to salvage D-PKE is called UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$]. That is, they consider *computationally unpredictable sources* which are also in class $\mathcal{S}_{\mathsf{PKE}}$, which is defined as follows:

$$\mathcal{S}_{\mathsf{PKE}} := \{\mathsf{S}_{\mathsf{PKE},\mathcal{A}} : \mathcal{A} \text{ is } \mathsf{PKE}\text{-valid}\}$$

A PPT algorithm $\mathcal{A}$ is called PKE-valid if on input the security parameter 1^λ it outputs two message vectors $(\mathbf{m}_0, \mathbf{m}_1)$ such that $|\mathbf{m}_0| = |\mathbf{m}_1|$ and $|\mathbf{m}_b[i]| = \mathsf{PKE.il}(\lambda) - \mathsf{PKE.pkl}(\lambda)$ for $b \in \{0,1\}$ and $i \in [|\mathbf{m}_0|]$, where $\mathsf{PKE.pkl}$ denotes the length function for public keys.

$\underline{\mathsf{S}^{\text{HASH}}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)}$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{PKE.KGen}(1^\lambda)$
$d \leftarrow\!\!\$\ \{0,1\}$
$(\mathbf{m}_0, \mathbf{m}_1) \leftarrow\!\!\$\ \mathcal{A}(1^\lambda)$
for $i = 1, \ldots, |\mathbf{m}_0|$ **do**
  $r \leftarrow\!\!\$\ \text{HASH}(\mathsf{pk} \| \mathbf{m}_d[i])$
  $\mathbf{c}[i] \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathbf{m}_d[i]; r)$
$L \leftarrow (\mathsf{pk}, d, \mathbf{c})$
return L

Similarly to bounded parallel sources, source class $\mathcal{S}_{\mathsf{PKE}}$ is not a single class of sources but a parameterized class, that is, each public-key encryption scheme PKE induces a corresponding source class $\mathcal{S}_{\mathsf{PKE}}$. Our negative result for Encrypt-with-Hash (Theorem 8.1) tells us that, if indistinguishability obfuscation exists, then there are public-key encryption schemes such that no hash function H can be UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{\mathsf{PKE}}$] secure. On the other hand, for a specific PKE scheme, say, RSA it might be that a hash function exists that is UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}_{RSA}$] secure.

OAEP. Similarly, for their OAEP application BHK construct a fine-tuned source class that is parameterized by a choice of trapdoor functions and various length functions. We here do not present the corresponding UCE notion but rather refer the interested reader to [BHK13:2].

9.6.5 Strongly Unpredictable Sources

Strong unpredictability is a strengthening of the unpredictability notion that we introduce in [BM14c] which requires that the source's queries remain unpredictable even if the predictor gets to see a set

$\underline{\text{stPred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)}$	$\underline{\text{HASH}(x)}$
$X^*, Y^* \leftarrow \{\}$	$X^* \leftarrow X^* \cup \{x\}$
$L \leftarrow\!\!\$\ \mathsf{S}^{\text{HASH}}(1^\lambda)$	$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$x' \leftarrow\!\!\$\ \mathsf{P}^{\text{HASH}}(1^\lambda, L, Y^*)$	$Y^* \leftarrow Y^* \cup \{y\}$
return $(x' \in X^*)$	**return** y

Figure 9.8: The strong unpredictability game where the predictor, in addition to the leakage is also given the result of the HASH queries.

containing all of the HASH oracle's answers. Formally, the restriction on sources captured by strong unpredictability is defined via the security game stPred given in Figure 9.8.

Similarly to plain unpredictability one can further differentiate between PPT predictors and unbounded predictors which gives rise to the UCE notions UCE[$\mathcal{S}^{\text{s-cup}}$] (UCEs for computationally strong-unpredictable sources) and its statistical counter-part UCE[$\mathcal{S}^{\text{s-sup}}$] (UCEs for statistically strong-unpredictable sources).

Split Sources vs. Strongly Unpredictable Sources

One can prove that split sources are a (strict) subclass of strongly unpredictable sources, that is, $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}} \subseteq \mathcal{S}^{\text{s-cup}}$ (and similarly in the statistical case $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}} \subseteq \mathcal{S}^{\text{s-sup}}$). For this note that the leakage L_0 of the first algorithm of a split source is independent of any oracle answers. Similarly, if the oracle is implemented by a random oracle (which is the case in the unpredictability experiment) then the leakage L_1 of the second algorithm is independent of any actual oracle query. The inclusion is strict. Consider, for example, a source that queries oracle HASH on x to receive y to then output $PRF_x(y)$ that is the image of a pseudorandom function at point y under key x. For the case of statistical unpredictability consider the source that outputs $x \oplus y$. Both distributions cannot be simulated by a split source. This yields the following lemma:

Lemma 9.8. *The class of split sources is a strict subclass of strongly unpredictable sources:*

$$\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{cup}} \subsetneq \mathcal{S}^{\text{s-cup}} \qquad \textit{and} \qquad \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}} \subsetneq \mathcal{S}^{\text{s-sup}}$$

We formally prove Lemma 9.8 for the statistical case. The computational case follows analogously.

Proof. We have already seen that there are sources in $\mathcal{S}^{\text{s-sup}}$ that are not in $\mathcal{S}^{\text{splt}}$. It, thus, remains to show that any source in $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}}$ is also a strongly, statistically unpredictable source. We assume that we are always in the random oracle setting, that is, HASH is implemented by a random oracle. Note that this is without loss of generality since membership in $\mathcal{S}^{\text{s-sup}}$ is defined via the strong unpredictability game stPred (Figure 9.8) which is always in the random oracle setting.

Let $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1)$ be a source in $\mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{sup}}$ and assume there exists a predictor P in the strong unpredictability game. We construct P' in the plain unpredictability game. Predictor P' gets as input leakage $L = (L_0, L_1)$. It guesses the number of queries q that were made by source S_0 and constructs a vector $\mathbf{y}'$ consisting of q random values of length $\mathsf{H.ol}(\lambda)$. It then runs source S_1 on input $\mathbf{y}'$ to receive leakage L_1'. Finally, it runs predictor P on input $((L_0, L_1'), Y^*)$ where Y^* is a set

containing the values in $\mathbf{y}'$ (i.e., a random permutation of the values with duplicates removed). It outputs whatever P outputs.

Analysis. As the unpredictability game (as well as the strong unpredictability game) is in the random oracle setting, the simulation of the input for S_1 is perfect in case P' guesses the correct number of queries. In this case leakage (L_0, L_1) and (L_0, L_1') are distributed identically and hence

$$\mathsf{Adv}^{\mathrm{pred}}_{\mathsf{S},\mathsf{P}'}(\lambda) = \frac{1}{\mathsf{max}_q} \cdot \mathsf{Adv}^{\mathrm{stpred}}_{\mathsf{S},\mathsf{P}}(\lambda),$$

where max_q is an upper bound on the maximum number of queries of source S_0. □

9.7 From Strong Unpredictability to (Plain) Unpredictability

Strongly unpredictable sources are not only related to split sources but there is also a surprising relationship to *plain* unpredictable sources. Namely, for some cases, the class of unpredictable sources is equivalent to the class of strongly unpredictable sources. This (partial) equivalence is highly surprising as on first sight it is not clear how to simulate the set of oracle answers for a predictor P. The trick is to consider UCEs that have a very short output length as this restricts potential oracle answers. In particular if we consider a UCE function which has only a single output bit, then there are exactly four possibilities for the set of oracle answers:

$$\Big\{\{\},\{0\},\{1\},\{0,1\}\Big\}.$$

For this note that the predictor in the strong unpredictability game stPred is not given the *list* of oracle answers, but the *set*. In order to formally state this result we first define the notion of bitwise UCEs, that is, UCEs that only output a single bit.

9.7.1 Bitwise UCEs

We next make a UCE notion explicit which was implicitly already considered by BHK in their original formulation of UCEs [BHK13:p], namely, UCEs that output only a single bit, i.e., for which $\mathsf{ol}(\lambda) = 1$. We extend the UCE notation to denote the output length as subscript: we write UCE_1 to denote functions that have a single bit output and write UCE for functions which have arbitrary (polynomial) output length.

BHK show that for unpredictable UCEs (both for computational unpredictability and statistical unpredictability) we can extend the output length of the function by running it in counter mode [BHK13:2]. That is, given a function H with output length $\mathsf{H.ol}$ we can construct a function $\overline{\mathsf{H}}$ with output length ℓ as

$\overline{\mathsf{H}}(\mathsf{hk}, x)$

for $i = 1, \ldots, \lceil \ell / \mathsf{H.ol}(\lambda) \rceil$ **do**
  $z_i \leftarrow \langle i \rangle_{\log \lceil \ell / \mathsf{H.ol}(\lambda) \rceil} \| x$
  $h_i \leftarrow \mathsf{H.Eval}(\mathsf{hk}, z_i)$
$h \leftarrow h_1 \| \cdots \| h_l$
return $h[1, \ell]$

Theorem 9.9 (Theorem 4.6 [BHK13:2])**.** *For any function* $\mathsf{H} \in \mathrm{UCE}_{\mathsf{H.ol}}[\mathcal{S}^{\mathrm{cup}}]$ *we can construct* $\overline{\mathsf{H}} \in \mathrm{UCE}_{\ell}[\mathcal{S}^{\mathrm{cup}}]$ *for an arbitrary function* ℓ *that is upper bounded by a polynomial. The same holds for statistical unpredictability, i.e.,* $\mathrm{UCE}[\mathcal{S}^{\mathrm{sup}}]$*.*

When we consider query-restricted sources such as $\mathcal{S}^{1\text{-query}}$ or $\mathcal{S}^{\mathsf{q}\text{-query}}$ then one needs to be careful with the above theorem. In case we consider sources making q-queries then we get that

$$\mathsf{H} \in \mathrm{UCE}_{\mathsf{H.ol}}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}] \implies \overline{\mathsf{H}} \in \mathrm{UCE}_{\ell}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathsf{p}\text{-query}}]$$

where p depends on q and the extension factor, i.e.,

$$\mathsf{p}(\lambda) = \left\lfloor \frac{\mathsf{q}(\lambda)}{\ell / \mathsf{H.ol}(\lambda)} \right\rfloor .$$

In particular this means that when we want to move from a constant output length to a polynomial output length this is not possible if we consider sources which are restricted to making a constant number of queries. Furthermore, we cannot argue that $\mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{1\text{-query}}]$ implies that $\overline{\mathsf{H}} \in \mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{1\text{-query}}]$ and, indeed, it seems unlikely that this implication holds.

9.7.2 From Strong Unpredictability to (Plain) Unpredictability

In the following section we show that for bit-wise UCEs, i.e., UCE_1 the strong unpredictability and (plain) unpredictability restrictions are equivalent. This holds both for the statistical case and for the computational case.

Theorem 9.10. *For any function* H *with* $\mathsf{H.ol}(\lambda) = 1$ *it holds that*

1. H *is UCE secure against computationally unpredictable sources* $\mathcal{S}^{\mathrm{cup}}$ *if and only if it is UCE secure against computationally strong unpredictable sources* $\mathcal{S}^{\text{s-cup}}$*:*

$$\mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\mathrm{cup}}] \iff \mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{s-cup}}]$$

2. H *is UCE secure against* statistically *unpredictable sources* $\mathcal{S}^{\mathrm{sup}}$ *if and only if it is UCE secure against statistically strong unpredictable sources* $\mathcal{S}^{\text{s-sup}}$*:*

$$\mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\mathrm{sup}}] \iff \mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{s-sup}}]$$

The two statements above hold also in the case the number of queries a source can make is restricted to some polynomial q*, that is*

$$\mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}] \iff \mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$$
$$\mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}] \iff \mathsf{H} \in \mathrm{UCE}_1[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$$

Proof. In order to prove the above theorem recall that the difference between unpredictability and strong unpredictability resides solely in the information the predictor is given. The following description captures both forms of unpredictability, the boxed statements showing the additional information given to the predictor in the strong unpredictability game:

$\mathrm{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ $\boxed{\mathrm{stPred}^{\mathsf{P}_*}_{\mathsf{S}}(\lambda)}$	$\textsc{Hash}(x)$
$X^* \leftarrow \{\}; \boxed{Y^* \leftarrow \{\}}$	$X^* \leftarrow X^* \cup \{x\}$
$L \leftarrow\!\!\$\ \mathsf{S}^{\textsc{Hash}}(1^\lambda)$	$y \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$x' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}_{\boxed{*}}(1^\lambda, L\boxed{, Y^*})$	$\boxed{Y^* \leftarrow Y^* \cup \{y\}}$
return $(x' \in X^*)$	**return** y

In the strong unpredictability game predictor P_* gets a set containing the answers from the HASH oracle. It is important to emphasize that this is indeed a set and not a list. Hence, when we consider bit-UCEs Y^* can only take one of 4 forms:

$$Y^* \in \left\{ \{\}, \{0\}, \{1\}, \{0,1\} \right\}.$$

As the correct Y^* can be guessed with probability $\frac{1}{4}$ we get that from any predictor P_* in game stPred we can construct a predictor P in game Pred such that

$$\mathsf{Adv}^{\text{stpred}}_{\mathsf{S},\mathsf{P}_*}(\lambda) = \frac{1}{4} \cdot \mathsf{Adv}^{\text{pred}}_{\mathsf{S},\mathsf{P}}(\lambda).$$

Similarly, any *good* predictor P in game Pred immediately yields a *good* predictor in game stPred which concludes the proof of Theorem 9.10. □

Combining Theorems 9.9 and Theorem 9.10 we obtain several interesting corollaries. Given that we can extend the length of a UCE function by sacrificing on the number of queries that a source can make we get:

Corollary 9.11. *It holds that*

$$\mathrm{UCE}_1[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}] \implies \mathrm{UCE}[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\mathsf{p}\text{-query}}]$$
$$\mathrm{UCE}_1[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}] \implies \mathrm{UCE}[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{p}\text{-query}}]$$

The implications are to be read as: if there exists H *that is* $\mathrm{UCE}_1[X]$ *secure, then there exists* $\overline{\mathsf{H}}$ *that is* $\mathrm{UCE}[Y]$ *secure.*

We have seen earlier that indistinguishability obfuscation and $\mathrm{UCE}[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{1\text{-query}}]$ are mutually exclusive (Theorem 9.6). Combining this with Corollary 9.11 yields that also $\mathrm{UCE}_1[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$

and in particular $\mathrm{UCE}[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{q-query}}]$ are mutually exclusive with indistinguishability obfuscation.

Corollary 9.12. *If indistinguishability obfuscation exists, then* $\mathrm{UCE}_1[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{q-query}}]$ *and thus, in particular,* $\mathrm{UCE}[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{q-query}}]$ *security cannot be achieved in the standard model.*

Chapter 10

UCEs for Computationally Unpredictable Sources

> "at this point our sense is that the existence of iO is more likely than the existence of families that are UCE[$\mathcal{S}^{\text{cup}}$]-secure or UCE[$\mathcal{S}^{\text{crs}}$]-secure, and that the latter assumptions should not be used. More broadly, the BFM attack [45] indicates that one must be careful in making any UCE-related assumptions."
>
> Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi [BHK13:1]

Summary. In this chapter we present an extension to our basic attack on UCE1 (UCE[$\mathcal{S}^{\text{cup}}$]) which uses randomized encodings to also show that UCEs restricted to bounded parallel sources cannot exist if indistinguishability obfuscation does. This result is based on a work published at CRYPTO 2014 [BFM14a]. We then recall a recent result by Bellare, Stepanovs, and Tessaro [BST15] who show that split sources can be used to construct MB-AIPO[$\mathcal{S}^{\text{cup}}$] and hence our negative result also carries over to UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$]. We present a direct form of their result to show that the existence of indistinguishability obfuscation implies the non-existence of UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$]-secure functions.

Chapter content

10.1 Introduction

In the previous chapter we introduced the UCE framework and discussed a variety of concrete UCE notions. We showed that UCEs with respect to *computationally* unpredictable sources cannot exist if indistinguishability obfuscation does (see Theorem 9.6, Section 9.5) and have seen that this negative result also carries over to UCEs with respect to strongly computationally unpredictable sources (UCE[$\mathcal{S}^{\text{s-cup}}$]) for sources that make polynomially many queries.

As discussed in Section 9.6.2, in an update to their UCE framework and as a response to our initial iO attack [BFM13a] Bellare, Hoang, and Keelveedhi [BHK13:1] (BHK) propose new UCE assumptions, based on computational unpredictability but restricted to so-called *bounded parallel sources* and *split sources*. These allow to recover the applications of deterministic public-key

encryption (D-PKEs), message-locked encryption (MLEs), as well as OAEP. In both cases sources are structurally restricted to consist of two parts as shown in Figure 10.1. While for split sources neither part of the source gets direct access to the HASH oracle, bounded parallel sources can access the HASH oracle in their second parallelized phase but, there, are restricted to be *very efficient*.

Extending the iO attack. In Section 10.2 we show that restricting computationally unpredictable sources to be also *bounded parallel* still falls prey to a similar but more complex version of the iO attack that we described in Section 9.5. The idea is to also split the iO attack into two stages consisting of a high-complexity first stage and a parallelizable second stage. To this end, we use the *randomized encodings* paradigm of Applebaum, Ishai, and Kushilevitz [AIK04] to bring down the complexity of the second stage of the attack. If f is a function, then the randomized encoding $\hat{f}(x;r)$ of $f(x)$ is simply an encoding of value $f(x)$ such that a decoder Dec can retrieve the original value $f(x)$ from it, i.e., $\mathsf{Dec}(\hat{f}(x;r)) = f(x)$. In addition, a randomized encoding specifies an efficient simulator Sim such that for all x the distributions $\hat{f}(x;r)$ over uniformly chosen r and $\mathsf{Sim}(f(x))$ are computationally indistinguishable. These properties combined allow us to show that we can adapt our original attack such that the source does not leak the obfuscated circuit but rather a randomized encoding of it. This alone, however, is still not enough for an attack with the restrictions of bounded parallel sources. To finalize the attack, we utilize a special form of so-called *decomposable* randomized encodings [IKOS08]. Such encodings have the property that each output bit of $\hat{f}(x;r)$ depends on at most a *single* bit of x (but possibly on the entire string r). The randomized encoding of Applebaum, Ishai, and Kushilevitz [AIK06] is decomposable and supports all functions in $\mathsf{P}/poly$. We show how to use such an encoding scheme to split the computation of the encoding into two phases: a complex first preprocessing phase which does not depend on the actual input and a very simple second stage which can be parallelized and where each parallel instance essentially only has to drop one of two bits. We show that this second stage (which will correspond to S_1) can be implemented by a constant-depth circuit consisting only of very few gates. This application of decomposable randomized encodings could be of interest also in other scenarios where efficiently computing an encoding is important and preprocessing is possible.

Very recently, Bellare, Stepanovs, and Tessaro (BST) showed how to use split source UCEs to construct multi-bit output point obfuscators [BST15]. In more detail, they show how to build an MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] from a UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$]-secure function.[1] Together with our negative result for MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] (see Theorem 7.3) this result implies that also indistinguishability obfuscation and UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] are mutually exclusive. In Section 10.3 we present a direct proof for this result, that is, we give the UCE adversary that can be extracted from the combination of the two proofs and show that it, indeed, succeeds in attacking UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] assuming indistinguishability obfuscation exists.

10.2 Bounded Parallel Sources

Let us recall the restriction behind bounded parallel sources as introduced by BHK in [BHK13:1]. In parallel sources the source splits into two parts S_0 and S_1 as follows. The first part of the source

[1] We note that BST show that the query restriction can be even stricter, that is, they can restrict the source to only make super-logarithmically many oracle queries. Furthermore, we note that they renamed MB-AIPO into *key-message leakage resilient symmetric encryption*.

Prl SOURCE $\mathsf{S}^{\text{HASH}}(1^\lambda)$

$(L_0, \mathbf{L}') \leftarrow\!\$\ \mathsf{S}_0(1^\lambda)$
for $i = 1, \ldots, \left|\mathbf{L}'\right|$ **do** $\mathbf{L}[i] \leftarrow\!\$\ \mathsf{S}_1^{\text{HASH}}(1^\lambda, \mathbf{L}'[i])$
$L \leftarrow (L_0, \mathbf{L})$
return L

Splt SOURCE $\mathsf{S}^{\text{HASH}}(1^\lambda)$

$(L_0, \mathbf{x}) \leftarrow\!\$\ \mathsf{S}_0(1^\lambda)$
for $i = 1, \ldots, |\mathbf{x}|$ **do** $\mathbf{y}[i] \leftarrow\!\$\ \text{HASH}(\mathbf{x}[i])$
$L_1 \leftarrow\!\$\ \mathsf{S}_1(1^\lambda, \mathbf{y}); L \leftarrow (L_0, L_1)$
return L

Figure 10.1: The parallel source $\mathsf{S} = \text{Prl}[S_0, S_1]$ on the left and the split source $\mathsf{S} = \text{Splt}[S_0, S_1]$ on the right as defined in the updated version of [BHK13:1]. In both cases the source consists of two parts S_0 and S_1 that jointly generate leakage L. For split sources neither part gets direct oracle access to HASH. For parallel sources additional restrictions on the runtime and the number of queries of S_1, as well as on the length of leakage L_0 are imposed. Note that the invocations of S_1 are parallelizable and independent of one another.

S_0 does not get oracle access to HASH, and simply outputs some preliminary leakage L_0 and a vector $\mathbf{L}'$ of arbitrary bit strings. For each entry in $\mathbf{L}'$ an independent instance of the second part of the source S_1 is run. This can be done in parallel as the several invocations do not share any coins or state. Instance i of S_1 is given $\mathbf{L}'[i]$ as input and produces leakage $\mathbf{L}[i]$. As opposed to S_0, the second part S_1 of parallel sources has oracle access to HASH. The final leakage of the source $\mathsf{S} := \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ is set to be $L := (L_0, \mathbf{L})$. The details of a parallel source $\mathsf{S} = \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ are given in Figure 10.1 on the left.

Without any further restrictions, parallel sources are as powerful as regular sources: simply ignore S_0 and let a single S_1 generate the entire leakage. Thus, in order to circumvent the iO attack, further restrictions are necessary. To this end, BHK restrict the resources of S_0 and S_1 via polynomials τ, σ, and q as follows: (1) the running time (circuit size) of each invocation of S_1 is at most $\tau(\cdot)$; (2) each invocation of S_1 makes at most $q(\cdot)$ oracle queries; and (3) the length of initial leakage L_0 output by S_0 is at most $\sigma(\cdot)$. BHK then consider the class $\mathcal{S}^{\text{prl}}_{\tau,\sigma,q}$ consisting of all parallel sources satisfying these bounds, and define UCE for computationally unpredictable, bounded parallel sources by considering $\text{UCE}[\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{prl}}_{\tau,\sigma,q}]$.

For their results on D-PKE and MLE schemes, the parameters τ, σ, and q need to be fine-tuned according to the underlying encryption scheme. More precisely, BHK set q to 1 (each instance of S_1 makes a single hash query), σ to the size of a key-pair (0 in the case of MLEs), and τ to the runtime of the encryption operation plus the input and key sizes of the encryption scheme. It is easily seen that our basic attack (see Section 9.5) does not fall into this class as long as the computation to construct the obfuscated circuit takes longer than what is granted by τ.

Choosing parameters for bounded parallel sources. We start by observing that BHK's bound on the initial leakage L_0 output by the first part of the source S_0 seems to be unnecessary. Indeed, there are no restrictions on the size of the combined leakage $L = (L_0, \mathbf{L})$ nor on the length of the vector $\mathbf{L}'$ in the definition. This in turn allows a source to easily bypass the leakage bound by routing L_0 through S_1. To see this, note that the source S_0 can simply split L_0 into several smaller packets and place them into the components of vector $\mathbf{L}'$. Various instantiations of source S_1 now simply recover these packets and leak them via their own leakage.

A second observation is that in choosing the parameters for bounded parallel sources, one has to strike a delicate balance between the complexity of obfuscating a hash function and the cost of encryption (resp. the application in question). Indeed, suppose that a bounded parallel source

assumption with parameters as above is used to prove an MLE scheme secure in the standard model. Now if the complexity of the encryption scheme is high—our schemes from Sections 5.5.5 and 8.2 as well as the scheme of Sahai and Waters [SW14] are, for example, build on top of obfuscation—then the assumption can be broken by the iO attack as described in the previous section. Similarly, if one could reduce the complexity of obfuscating the hash function, an attack would become feasible. However, considering the current state of research, obfuscation is a very costly operation and thus, intuitively, computing the obfuscation of a hash function should be harder than encrypting a message. Interestingly, it is the *parallel* complexity of obfuscating a hash function (after a possibly complex preprocessing phase) that matters for the attack, and we can show that the latter can lie in a complexity class which is dramatically below that of computing the obfuscation of the hash function. More precisely, we show how to combine our iO attack with the *randomized encodings* of Applebaum, Ishai, and Kushilevitz [AIK04] to split the attack into two stages such that the second stage is highly parallelizable, indeed, we show that it is in NC^0. Before describing our attack, let us recall the notion of randomized encodings.

10.2.1 Randomized Encodings

Randomized encodings allow to substantially reduce the complexity of computing a function f by instead computing an *encoding* of it. This technique was first introduced by Ishai and Kushilevitz [IK00, IK02] in the context of multi-party computation and has since found many applications [AIK04, AIK06, IKOS08, GIS+10, AIKW13, App14]. The formalization of randomized encodings that we use here is due to Applebaum, Ishai, and Kushilevitz [AIK04] and is adapted to the setting of perfect correctness and computational privacy. Informally, we say that $\hat{f}(x;r)$ is a randomized encoding of some function value $f(x)$ if: 1) given $\hat{f}(x;r)$ one can efficiently recover function value $f(x)$, and 2) given $f(x)$, one can efficiently sample from the distribution $\hat{f}(x;r)$ induced by uniformly choosing r. The randomized encodings that we employ work on circuits as description for functions, that is, we construct a randomized encoding for a circuit C_f implementing function f. To stay consistent with the notation used in the literature for randomized encodings we identify by f both the function and the corresponding circuit description.

More precisely, a randomized encoding scheme RE consists of three efficient algorithms (Enc, Dec, Sim) as follows: The probabilistic encoding algorithm Enc on input the security parameter 1^λ, a description of a circuit computing function $f_\lambda : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ (of size polynomial in λ) and an input $x \in \{0,1\}^\lambda$ outputs an encoding $\langle f_\lambda[x] \rangle \leftarrow\!\!\$\ \mathsf{RE.Enc}(1^\lambda, f_\lambda, x)$ such that $\langle f_\lambda[x] \rangle \in \{0,1\}^{\mathsf{RE.ol}(\lambda)}$. As mentioned we simplify notation and identify by f_λ both the circuit and its description. The deterministic decoder algorithm Dec on input the security parameter 1^λ and an encoding $\langle f_\lambda[x] \rangle \in \{0,1\}^{\mathsf{RE.ol}(\lambda)}$ outputs an image point $y \in \{0,1\}^{\ell(\lambda)}$. Finally, the probabilistic simulation algorithm Sim on input 1^λ and an image point $y \in \{0,1\}^{\ell(\lambda)}$ outputs an encoding $z \in \{0,1\}^{\mathsf{RE.ol}(\lambda)}$.

What we require from a randomized encoding is that it is *perfectly correct* and *computationally private*. The first property asks that it does not matter whether we evaluate function f_λ directly or via the randomized encoding: we always get the same result. For the second property we require that no computationally bounded algorithm can distinguish between seeing values generated by simulator Sim or honest encodings. We formalized these properties next:

Definition 10.1. *Let* $\mathsf{RE} := (\mathsf{Enc}, \mathsf{Dec}, \mathsf{Sim})$ *be a randomized encoding scheme. We say* RE *is* perfectly correct *for a circuit class* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *if it satisfies the following property:*

- CORRECTNESS. *For any* $\lambda \in \mathbb{N}$*, any* $f_\lambda \in \mathcal{F}_\lambda$ *and any input* $x \in \{0,1\}^\lambda$ *we have that*

$$\Pr\left[\mathsf{RE.Dec}\left(1^\lambda, \mathsf{RE.Enc}(1^\lambda, f_\lambda, x)\right) = f_\lambda(x)\right] = 1,$$

where the probability is over the random coins for RE.Enc.

We say that scheme RE *is a computationally private randomized encoding for a circuit class* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *if it satisfies:*

- PRIVACY. *For any pair* $(\mathsf{Sam}, \mathsf{D})$ *of* PPT *algorithms, any* $f_\lambda \in \mathcal{F}_\lambda$ *the following distinguishing advantage in game* $\mathsf{RE\text{-}PRIV}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{RE},f_\lambda}$ *(on the right) is negligible:*

$$\mathsf{Adv}^{\mathrm{re}}_{\mathsf{RE},f_\lambda,\mathsf{Sam},\mathsf{D}}(\lambda) := 2 \cdot \Pr\left[\mathsf{RE\text{-}PRIV}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{RE},f_\lambda}(\lambda)\right] - 1$$

$\underline{\mathsf{RE\text{-}PRIV}^{\mathsf{Sam},\mathsf{D}}_{\mathsf{RE},f_\lambda}(\lambda)}$

$b \leftarrow\!\$\ \{0,1\}$

$(x, \mathsf{state}) \leftarrow\!\$\ \mathsf{Sam}(1^\lambda)$

$\langle f_\lambda[x]\rangle_0 \leftarrow\!\$\ \mathsf{Sim}(1^\lambda, f_\lambda(x))$

$\langle f_\lambda[x]\rangle_1 \leftarrow\!\$\ \mathsf{RE.Enc}(1^\lambda, f_\lambda, x)$

$b' \leftarrow\!\$\ \mathsf{D}(1^\lambda, \langle f_\lambda[x]\rangle_b, \mathsf{state})$

return $b = b'$

To keep our notation consistent with the previous literature on randomized encodings, for a given circuit f_λ, we will refer to the the mapping $\mathsf{RE.Enc}(1^\lambda, f_\lambda, \cdot; \cdot)$ by $\hat{f}_\lambda : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{RE.rl}(\lambda)} \to \{0,1\}^{\mathsf{RE.ol}(\lambda)}$, where $\{0,1\}^{\mathsf{RE.rl}(\lambda)}$ is the randomness space of the randomized encoding. That is, we write

$$\hat{f}_\lambda(x; r)$$

to identify encoding $\langle f_\lambda[x]\rangle$ generated as $\mathsf{RE.Enc}(1^\lambda, f_\lambda, x; r)$.

A randomized encoding can be trivially achieved by setting $\hat{f}(x; r) := f(x)$. However one is usually interested in an encoding, $\hat{f}$, which is in a low complexity class (typically NC^0), although f itself might be only computable only in a higher class. Applebaum et al. [AIK04, AIK06], utilizing garbled circuits, construct such encodings for several standard cryptographic primitives, such as one-way functions and pseudo-random generators, in NC^0. In our case, the complexity of the encoding is not crucial and, indeed, we only require it to be computable in polynomial time. Rather, for our application, we are concerned with the *input locality* of the computation of $\hat{f}$. More precisely, the number of input bits of x which affect each output bit of the encoding $\hat{f}(x; r)$ should be small. We will return to the topic of locality in Section 10.2.3.

10.2.2 Composing Obfuscation with Randomized Encodings

To ease readability, we present our attack in two stages. First, we show that our iO attack can be composed with any randomized encoding scheme in a way which neither affects the adversary's advantage nor the unpredictability of its implicit source. Then, in Section 10.2.3, we use a special type of randomized encoding scheme known as a *decomposable randomized encoding* [IKOS08] to split and parallelize the adversary's source in order to meet the (minimal) bounds of $q(\lambda) = 1$, $\sigma(\lambda) = 0$, and $\tau(\lambda) \in \Omega(\lambda)$. Consequently, our attack will rule out bounded parallel sources for these

parameters. Since the bounds that our attacks achieves are very stringent and an encryption scheme has to at least run in time $\mathcal{O}(\lambda)$ (and make a single HASH query), assuming indistinguishability obfuscation, it is unlikely that bounded parallel sources can be used to instantiate random oracles in any meaningful application scenario. This is captured by the following theorem.

Theorem 10.1. *If indistinguishability obfuscation and one-way functions exist, then* $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{prl}}_{\tau,\sigma,q}]$ *security cannot be achieved in the standard model for* $q \neq 0$*, any* $\sigma \geq 0$*, and* $\tau \in \Omega(\lambda)$*.*

Recall our basic iO attack on $\mathrm{UCE}[\mathcal{S}^{\mathrm{cup}}]$-secure functions from Section 9.5. There, source S queried oracle HASH on a uniformly random x to receive value y and then constructed circuit $C[x, y]$ as follows:

$$\begin{array}{l} \underline{C[x,y](\mathsf{hk})} \\ y' \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x) \\ \textbf{return } (y = y') \end{array}$$

Source S then created an iO of circuit $C[x, y]$ which it output as leakage. To complete the attack, all that is left for the distinguisher is to run the leaked circuit on the provided key. In order for source S to be computationally unpredictable we assumed that $\mathsf{H.ol}(\lambda) \geq 2 \cdot \mathsf{H.kl}(\lambda)$. As explained, this is without loss of generality, as we can either let the source output multiple pairs (x_i, y_i) or compose the hash function with a pseudorandom generator. That is, we consider

$$\begin{array}{l} \underline{C[x,s](\mathsf{hk})} \\ s' \leftarrow \mathsf{PRG}(\mathsf{H.Eval}(\mathsf{hk}, x)) \\ \textbf{return } (s = s') \end{array}$$

where now instead of value y the circuit has value $s \leftarrow \mathsf{PRG}(y)$ hard-coded, where PRG is a pseudorandom generator with sufficient stretch (say, PRG doubles its input length).

In the following, we adapt the attack such that the source is a bounded parallel source, that is, we consider $\mathsf{S} \in \mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{prl}}_{\tau,\sigma,q}$ for minimal values of τ, σ and q. The basic idea is that, instead of leaking an obfuscation of $C[x, s]$ we leak a randomized encoding of a circuit computing function

$$f : (x, y, r_{\mathrm{io}}) \mapsto \mathsf{iO}(C[x, \mathsf{PRG}(y)]; r_{\mathrm{io}}).$$

That is, function f takes as input a pair (x, y) and random coins r_{io} and computes an obfuscation of circuit $C[x, s]$ with random coins r_{io} and with $s \leftarrow \mathsf{PRG}(y)$. In full detail we can describe f as

$$\begin{array}{l} \underline{f(x, y, r_{\mathrm{io}})} \\ s \leftarrow \mathsf{PRG}(y) \\ \overline{C} \leftarrow \mathsf{iO}(C[x, s]; r_{\mathrm{io}}) \\ \textbf{return } \overline{C} \end{array}$$

Note that f is deterministic and the *random coins* used for the indistinguishability obfuscator are provided as "normal input".

We now construct a source S which instead of directly leaking an obfuscation of $C[x, s]$ leaks a randomized encoding of f. That is, source S chooses a random value x which it queries to oracle

HASH to obtain $y \leftarrow\!\!\$\ \text{HASH}(x)$. It additionally chooses two sets of random coins r_{io} and r_{Enc}. The first set of coins r_{io} is used as input to f as random coins for the obfuscation operation and the second set of coins r_{Enc} is used to compute the randomized encoding $\hat{f}$. Finally, it computes and leaks the randomized encoding of $f(x, y, r_{\text{io}})$, that is,

$$L \leftarrow \hat{f}(x, y, r_{\text{io}}; r_{\mathsf{Enc}}) \ .$$

By the perfect correctness of the randomized encoding we have that $\hat{f}(x, y, r_{\text{io}}; r_{\mathsf{Enc}})$ is an encoding of the obfuscated circuit $C[x, s]$. Hence, using the decoder of the randomized encoding scheme we can recover $\mathsf{iO}(C[x, s]; r_{\text{io}})$, that is, we can recover a circuit that is functionally equivalent to circuit $C[x, s]$ and which is of polynomial size.

To complete the description of the adversary, we define distinguisher D to operate as follows: Distinguisher D gets as input a hash key hk and an encoding $\hat{f}(x, y, r_{\text{io}}; r_{\mathsf{Enc}})$. It uses the decoder Dec of the randomized encoding scheme to recover

$$f(x, y, r_{\text{io}}) \leftarrow \mathsf{RE.Dec}(1^\lambda, \hat{f}(x, y, r_{\text{io}}; r_{\mathsf{Enc}})) \ .$$

It then interprets the result as a circuit (which is functionally equivalent to $C[x, s]$), runs it on hk, and returns whatever the circuit outputs.

There are four tasks left, in order to complete the proof of Theorem 10.1. We need to show that 1) the advantage of (S, D) in the UCE game is sufficiently good; 2) source S is computationally unpredictable; 3) source S is a bounded parallel source with restrictions as in Theorem 10.1; and 4) we can construct a suitable randomized encoding scheme. We prove all properties in turn.

Advantage. We start by showing that adversary (S, D) has overwhelming advantage in the UCE game for any standard-model hash function H.

Claim 10.2. *Let* H *be a hash function family. Let* PRG *be a secure pseudorandom generator with stretch at least* 2*. Let* RE *be a randomized encoding that is perfectly correct. Then, the pair* (S, D) *as defined above has overwhelming advantage in game* $\text{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}$.

Proof. Distinguisher D as defined above uses the decoder of the randomized encoding to obtain value $f(x, y, r_{\text{io}})$. Due to the definition of f and the perfect correctness of the encoding scheme this value is equal to $\mathsf{iO}(C[x, s]; r_{\text{io}})$. A similar analysis as in Theorem 9.6 shows that in case y was computed as $\mathsf{H.Eval}(\mathsf{hk}, x)$, the obfuscated circuit, and hence D, always return 1. In case that y was drawn at random, it is highly unlikely, down to the security of the pseudorandom generator, that

$$\mathsf{PRG}(\mathsf{H.Eval}(1^\lambda, \mathsf{hk}, x)) = \mathsf{PRG}(y),$$

and, thus, D returns 0 with overwhelming probability. For this note that otherwise we can construct an adversary against the PRG security which on input a value s chooses a random value x and random key hk, computes $\mathsf{PRG}(\mathsf{H.Eval}(1^\lambda, \mathsf{hk}, x))$ and compares the result to s. In case s was generated as $\mathsf{PRG}(y)$ for a uniformly random y, we perfectly simulate the above setup. In case s is uniformly random, with probability $1 - 2^{-\lambda}$ there is no value y such that $\mathsf{PRG}(y) = s$ because we have chosen PRG to double its input length.

$\underline{\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)}$

$\mathsf{done} \leftarrow \textbf{false}$
$Q \leftarrow \{\}$
$L \leftarrow\!\!\$\ \mathsf{S}^{\text{HASH}}(1^\lambda)$
$\mathsf{done} \leftarrow \textbf{true}$
$Q' \leftarrow\!\!\$\ \mathsf{P}^{\text{HASH}}(1^\lambda, L)$
return $(Q \cap Q' \neq \{\})$

$\underline{\text{HASH}(x)}$

if $\mathsf{done} = \textbf{false}$ **then**
  $Q \leftarrow Q \cup \{x\}$
if $T[x] = \perp$ **then**
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$

Figure 10.2: We here repeat the unpredictability game $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}$ for UCEs. Unpredictability for UCEs is defined in Section 9.2.2.

It follows that distinguisher D together with source S have overwhelming advantage in the UCE game. □

Unpredictability. We next show that source S is computationally unpredictable, that is, $\mathsf{S} \in \mathcal{S}^{\text{cup}}$. For this we use the computational privacy of the randomized encoding, the security of the PRG as well as the security of the indistinguishability obfuscator.

Claim 10.3. *Let* PRG *be a secure pseudorandom generator with stretch at least* 2*. Let* RE *be a randomized encoding that is computationally private. Then, source* S *as defined above is computationally unpredictable, that is,* $\mathsf{S} \in \mathcal{S}^{\text{cup}}$*.*

Proof. We need to show that query x is computationally hidden given leakage L from source S when oracle HASH implements a random oracle. As in Theorem 9.6, we observe that the circuit $C[x, s]$ is the constant zero circuit with overwhelming probability if s is chosen uniformly at random,

Let P be a predictor that succeeds with non-negligible probability in the $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ game (see Figure 10.2). We bound the success probability of P down to the security of the pseudorandom generator, the security of the indistinguishability obfuscator iO and the privacy of the randomized encoding scheme via a sequence of 4 games as follows (we give the pseudocode in Figure 10.3).

$\mathsf{Game}_1(\lambda)$**:** The first game is identical to the computational unpredictability game $\text{Pred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)$ (see Figure 10.2).

RE

$\mathsf{Game}_2(\lambda)$**:** is equivalent to the previous game except that the source S now computes the leakage by first computing the circuit $\mathsf{iO}(C[x, s])$ as an obfuscation of $C[x, s]$ and then running the randomized encoding simulator Sim on input the obfuscated circuit, i.e., leakage L is computed as $L \leftarrow\!\!\$\ \mathsf{RE.Sim}(\mathsf{iO}(C[x, s](\cdot)))$.

PRG

$\mathsf{Game}_3(\lambda)$**:** is similar to the previous game except that now value s is sampled uniformly at random in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$.

iO

$\mathsf{Game}_4(\lambda)$**:** is similar to the previous game except that source S now computes $\mathsf{iO}(\mathsf{Z}_{|C[x,s]|})$, that is, it computes an obfuscation of the constant zero circuit padded to length $|C[x, s]|$, instead of an obfuscation of $C[x, s]$.

In the following we bound the difference between each consecutive game and show that the distinguishing advantage is negligible. In Game_4 we note that predictor P only has negligible advantage since its input does no longer contain any information on the single query x.

$\mathsf{Game}_1^{\mathsf{P}}(\lambda)$	$\xrightarrow{\text{RE}}$	$\mathsf{Game}_2^{\mathsf{P}}(\lambda)$	$\xrightarrow{\text{PRG}}$	$\mathsf{Game}_3^{\mathsf{P}}(\lambda)$	$\xrightarrow{\text{iO}}$	$\mathsf{Game}_4^{\mathsf{P}}(\lambda)$
1 : $x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.il}(\lambda)}$		$x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.il}(\lambda)}$		$x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.il}(\lambda)}$		$x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.il}(\lambda)}$
2 : $y \leftarrow\!\!\$\ \textsc{Hash}(x)$		$y \leftarrow\!\!\$\ \textsc{Hash}(x)$				
3 :		$s \leftarrow \mathsf{PRG}(y)$		$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$		$s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$
4 : $r_{\text{io}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{iO.rl}(\lambda)}$		$r_{\text{io}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{iO.rl}(\lambda)}$		$r_{\text{io}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{iO.rl}(\lambda)}$		$r_{\text{io}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{iO.rl}(\lambda)}$
5 : $r_{\mathsf{Enc}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{RE.rl}(\lambda)}$						
6 :		$\overline{C} \leftarrow \mathsf{iO}(C[x,s]; r_{\text{io}})$		$\overline{C} \leftarrow \mathsf{iO}(C[x,s]; r_{\text{io}})$		$\overline{C} \leftarrow \mathsf{iO}(\mathsf{Z}_{\lvert C[x,s]\rvert}; r_{\text{io}})$
7 : $L \leftarrow \hat{f}(x, y, r_{\text{io}}; r_{\mathsf{Enc}})$		$L \leftarrow\!\!\$\ \mathsf{RE.Sim}(\overline{C})$		$L \leftarrow\!\!\$\ \mathsf{RE.Sim}(\overline{C})$		$L \leftarrow\!\!\$\ \mathsf{RE.Sim}(\overline{C})$
8 : $Q' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$		$Q' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$		$Q' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$		$Q' \leftarrow\!\!\$\ \mathsf{P}^{\textsc{Hash}}(1^\lambda, L)$
9 : **return** $(x \in Q')$		**return** $(x \in Q')$		**return** $(x \in Q')$		**return** $(x \in Q')$

$\textsc{Hash}(x)$

if $T[x] = \perp$ **then**
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x]$

$f(x, y, r_{\text{io}})$

$s \leftarrow \mathsf{PRG}(y)$
$\overline{C} \leftarrow \mathsf{iO}(C[x,s]; r_{\text{io}})$
return $\overline{C}$

$C[x,s](\mathsf{hk})$

$y' \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$
$s' \leftarrow \mathsf{PRG}(y')$
return $(s = s')$

Figure 10.3: The pseudocode for proof of Claim 10.3

$\mathsf{Game}_1(\lambda)$ to $\mathsf{Game}_2(\lambda)$. We bound the difference between P's advantage in Game_1 and in Game_2 using the privacy of the RE scheme. For this note that the difference in computing leakage L resembles exactly the difference in game RE-PRIV (see Definition 10.1). We construct an adversary $(\mathsf{Sam}, \mathsf{D})$ against the randomized encoding as follows: Sampler Sam samples a uniformly random point $x \in \{0,1\}^{\mathsf{H.il}(\lambda)}$, a uniformly random point $y \in \{0,1\}^{\mathsf{H.ol}(\lambda)}$ and random coins $r_{\text{io}} \in \{0,1\}^{\mathsf{iO.rl}(\lambda)}$. It outputs (x, y, r_{io}) and stores x in state. As function we consider function f. Distinguisher D gets as input an encoding $\langle f[x, y, r_{\text{io}}]\rangle$ which is either an honestly generated or a simulated encoding. It runs predictor P on input the security parameter and encoding $\langle f[x, y, r_{\text{io}}]\rangle$ to obtain a set Q' and it outputs 1 if $x \in Q'$.

By construction, adversary $(\mathsf{Sam}, \mathsf{D})$ perfectly simulates Game_1 in case encoding $\langle f[x, y, r_{\text{io}}]\rangle$ is generated honestly as $\hat{f}(x, y, r_{\text{io}})$ (with uniformly random coins) and perfectly simulates Game_2 in case the encoding is simulated. It follows:

$$\left|\Pr\left[\mathsf{Game}_2^{\mathsf{P}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathsf{P}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{re}}_{\mathsf{RE},f,\mathsf{Sam},\mathsf{D}}(\lambda).$$

$\mathsf{Game}_2(\lambda)$ to $\mathsf{Game}_3(\lambda)$. In Game_3 value s is no longer chosen by applying the pseudorandom generator to value y but by simply choosing value s uniformly at random in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$. The analysis for this game hop is analogous to, for example, the analysis of the first game hop in the proof of Theorem 8.1 (page 158) or the first game hop in the proof of Lemma 7.4 (page 145). The distinguishing advantage is upper bounded by the advantage of an adversary against the security of the PRG. We, thus, have:

$$\left|\Pr\left[\mathsf{Game}_3^{\mathsf{P}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_2^{\mathsf{P}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda)$$

where adversary $\mathcal{A}$ simulates Game_2 using its input s instead of the s computed in line 3.

$\mathsf{Game}_3(\lambda)$ to $\mathsf{Game}_4(\lambda)$. The games are identical but for the obfuscated circuit. That is, in Game_3 circuit $C[x,s]$ is obfuscated while in Game_4 the constant zero circuit $\mathsf{Z}_{|C[x,s]|}$ is obfuscated. The analysis of this step is identical to the analysis of the last step in the proof for the basic iO attack (Theorem 9.6 page 9.6). That is, a distinguisher induces a distinguisher against the indistinguishability obfuscator.

$$\left|\Pr\left[\mathsf{Game}_4^{\mathsf{P}}(\lambda)=1\right]-\Pr\left[\mathsf{Game}_3^{\mathsf{P}}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{Sam},\mathsf{D}}(\lambda)+\frac{1}{2^{\lambda}}.$$

For the factor $2^{-\lambda}$ note that, by assumption, PRG is length doubling and, thus, a uniformly random $s \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ is with probability $1-2^{-\lambda}$ not in the image of PRG.

Prediction advantage in Game_4. Finally, in $\mathsf{Game}_4(\lambda)$, the probability that P guesses the single query x is upper bounded by $|Q(\lambda)|\,/2^{\mathsf{H.il}(\lambda)}$, where $|Q(\lambda)|$ denotes the number of guesses the predictor outputs. For this note that x is chosen uniformly at random and the leakage L that the predictor receives is independent of x; the predictor gets an obfuscation of the constant zero circuit. Putting the above distances together, we get that

$$\Pr\left[\mathrm{Pred}_{\mathsf{S}}^{\mathsf{P}}(\lambda)=1\right] \leq \mathsf{Adv}^{\mathrm{re}}_{\mathsf{RE},f,\mathsf{Sam},\mathsf{D}}(\lambda)+\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda)+\mathsf{Adv}^{\mathrm{io}}_{\mathsf{Sam},\mathsf{D}}(\lambda)+\frac{1}{2^{\lambda}}+\frac{|Q(\lambda)|}{2^{\mathsf{H.il}(\lambda)}}\ .$$

We conclude that, P's advantage is negligible, and the source S is computationally unpredictable. □

Thus far we have shown properties 1) and 2), namely that the advantage of (S,D) in the UCE game is sufficiently good and that source S is computationally unpredictable. In the following section we show the two remaining properties, namely, that source S is a bounded parallel source with restrictions as in Theorem 10.1; and that we can construct a suitable randomized encoding scheme.

10.2.3 Splitting-Up and Parallelizing Source S Using Decomposable REs

Our analysis so far holds for any randomized encoding scheme. We next show that if a *decomposable* scheme, is used to instantiate the attack, that we can recast the source above as a bounded parallel source. Let us begin with the definition of decomposable randomized encodings.

Definition 10.2 (Decomposable randomized encodings)**.** *A randomized encoding scheme* RE *is called* decomposable randomized encoding *(DRE) for a circuit class* $\{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *if for all* $\lambda\in\mathbb{N}$ *and all* $f_\lambda\in\mathcal{F}_\lambda$ *and any choice of* x *and* r *we have that the encoding* $\hat{f}(x;r)$ *depends on at most a single bit of* x *(but possibly on arbitrarily many bits of* r*). Furthermore, there exists an efficient algorithm* idx *that on input a circuit* f_λ *and an index* $i\in[\mathsf{DRE.ol}(\lambda)]$ *outputs an index* $j\in[\lambda]\cup\{0\}$ *such that if* $j>0$ *the output bit at position* i *depends only on input* $x[j]$*.*

In other words, in a decomposable randomized encoding (DRE) scheme every output bit of the encoding $\hat{f}(x;r)$ depends on at most a single bit of x (but possibly on arbitrarily many bits of r). We can think of a decomposable randomized encoding scheme DRE to consist of four efficient algorithms $(\mathsf{idx},\mathsf{Enc},\mathsf{Dec},\mathsf{Sim})$ as follows. Algorithm idx on input a circuit f and an index $i\in[\mathsf{DRE.ol}(\lambda)]$ outputs an index $j\in[\lambda]\cup\{0\}$. The decomposable encoding algorithm Enc operates based on a local encoding algorithm $\overline{\mathsf{Enc}}$ as follows. On input a circuit f, a point x, and random coins r_{Enc}, for each $i\in[\mathsf{DRE.ol}(\lambda)]$ compute $\langle f[x]\rangle_i \leftarrow \overline{\mathsf{Enc}}(f,i,x[\mathsf{idx}(f,i)];r_{\mathsf{Enc}})$, where we define $x[0]:=\perp$, and return $\langle f[x]\rangle \leftarrow (\langle f[x]\rangle_1,\ldots,\langle f[x]\rangle_{\mathsf{DRE.ol}(\lambda)})$. Algorithms Dec and Sim play the same roles as those in a

conventional randomized encoding scheme. As before, we denote $\overline{\mathsf{Enc}}(f, i, b; r_{\mathsf{Enc}})$ by $\hat{f}_i(b; r_{\mathsf{Enc}})$. Thus, we may write

$$\hat{f}(x; r_{\mathsf{Enc}}) = \hat{f}_1(x[\mathsf{idx}(1)]; r_{\mathsf{Enc}}) \| \hat{f}_2(x[\mathsf{idx}(2)]; r_{\mathsf{Enc}}) \| \cdots \| \hat{f}_s(x[\mathsf{idx}(\mathsf{dre.ol}(\lambda))]; r_{\mathsf{Enc}}) \ .$$

As Ishai et al. [IKOS08] point out, several constructions of randomized encodings are decomposable. For example, the construction based on Yao's garbled circuits [Yao86] due to Applebaum et al [AIK06] is a decomposable, perfectly correct, and computationally private randomized encoding for any efficiently computable function. Their construction relies only on the existence of secure pseudorandom generators. We capture this as

Fact 10.4 ([AIK06] Construction 4.7)**.** *If one-way functions exist then a decomposable, perfectly correct, and computationally private randomized encoding exists for any efficiently computable function.*

Applebaum et al. [AIK06] show that Yao's garbled circuits [Yao86] can be used to construct decomposable randomized encodings. They give the following intuition behind their construction which, in particular, also emphasizes the fact that the construction yields a *decomposable* randomized encoding:

Consider a circuit f and an input x. To obtain an encoding $\langle f[x] \rangle$ we first assign to every wire in f two keys: a 0-key and a 1-key representing the two possible values for this wire. We then randomly paint one key of each pair black and one white. Thus, given a white key does not reveal whether this key represents a 0 or a 1 as the coloring was done randomly. In a next step we encode each gate (AND and OR) of the circuit. Note that each gate has two input wires and one output wire and, thus, we have two associated key pairs for the inputs and one associated key pair for the output of the gate. The encoding of a gate consists of four boxes each locked with two keys: a white-white box (locked with the two white keys corresponding to the two input wires), a white-black box (locked with the white key of the left input and the black key of the right input), a black-white box (locked with the black key of the left input and the white key of the right input) and a black-black box (locked with both black keys). Inside each box we put one of the two output keys corresponding to the function represented by the gate: if the box is locked with keys representing bits a and b then we put the key representing bit $g(a, b)$ into the box where g represents the gate's function (either AND or OR). Note that each gate can be encoded in parallel. If for a gate one is given two keys corresponding to one of the two input wires one is, thus, able to open exactly one of the four boxes and obtain exactly one key for the output wire. Furthermore, just holding the keys does not reveal any information on the semantics of the keys.

Via the above method we can encode a function f. To complete the encoding for an input x note that value x induces one key for each input wire which corresponds to the value of x for this wire. These are called the active keys. The encoding $\langle f[x] \rangle$ contains the active keys corresponding to x for all the input wires of f and this is the only point where the encoding depends on input x. Given the active keys for the input wires and the encodings of each gate one can now "compute value $f(x)$" in a top-down fashion by opening the encoded boxes one after the other, until the final box is opened. As noted before, this process does not leak any information about the actual function value as all that is learned is whether or not the function evaluates to a black or a white key at the

very last level. Thus, to make this information useful the encoding $\langle f[x] \rangle$ additionally contains the information whether the 1-key of the output wire is black or white.

Parallelizing the source with decomposable encodings. Next we show that we can compute decomposable randomized encodings in a special way consisting of two phases where the first phase is not given the entire input (this can easily be extended such that the first phase does not depend on the input at all), and the second phase can be computed in parallel by constant-depth circuits of low gate count. We will use this result to adapt our above attack to bounded parallel sources by constructing a source $\mathsf{S} = \mathrm{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ that computes the very same encoding that our previously described source did. Looking ahead, we note that the decomposability of the encoding is convenient because every bit of the encoding depends on only a single bit of the actual input and one can easily precompute both possibilities (that is, the encodings when the input bit is 0 and when it is 1) as long as one knows the randomness for the encoding. Once the actual input is known all one has to do to compute a proper encoding is to drop one of the two bits, an operation which can be easily parallelized.[2]

Claim 10.5. *Let* RE *be a* decomposable *randomized encoding. Then, we can compute source* S *with the restriction of a bounded parallel source with* $\tau \in \Omega(\lambda)$*,* $q = 1$*, and* $\sigma = 0$*. That is:* $\mathsf{S} \in \mathcal{S}^{\mathrm{prl}}_{\Omega(\lambda),0,1}$.

Proof. We will rewrite source S as $\mathrm{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ as follows where we give the pseudocode of algorithms S_0 and S_1 in Figure 10.4 and next give a textual description.

Algorithm S_0. Algorithm S_0 of the source begins by generating the randomness for the randomized encoder and the obfuscator (line 2). It then picks a random x in the domain of the hash function and sets an auxiliary variable aux to $x \| 0^{\mathsf{H.ol}(\lambda)} \| r_{\mathrm{io}}$ (line 4). We will use this variable to access the specific input bits that the randomized encoder needs. Note that we have not yet specified a proper value for y at this point, but fixed it to the arbitrary string $0^{\mathsf{H.ol}(\lambda)}$. Next, algorithm S_0 generates a leakage value for every output bit of the encoding, i.e., for every $i \in [\mathsf{DRE.ol}(\lambda)]$ (where $\mathsf{DRE.ol}(\lambda)$ describes the size of the encoding). Here, we distinguish two cases. If the i-th output bit of the encoding depends on an input bit corresponding to y, then the if branch in line 6 is executed. Otherwise, the else branch in line 11 is chosen. In the first case, we first compute the index j of y that the encoding depends upon (line 7). As at this point y has not yet been chosen, S_0 computes the output bits for the two possible values of $y[j]$; that is, it stores the output bit for $y[j] = 0$ as b_0 and the output bit for $y[j] = 1$ as b_1. It then sets the leakage at position i to $1 \| b_0 \| b_1 \| j \| x$ (line 10). In the second case (i.e., when the output bit of the encoding does not depend on a bit of y), algorithm S_0 simply computes the output bit (it knows the value of the corresponding input bit), stores it in b, and sets the leakage at position i to $0 \| b$ (line 13). At the end of the for loop $\mathbf{L}'$ contains a single value for every output bit of the randomized encoding. Algorithm S_0 returns $(\varepsilon, \mathbf{L}')$. That is, it does not directly contribute to the leakage and hence $\sigma = 0$.

Algorithm S_1. An independent instance of algorithm S_1 (see Figure 10.4 on the right) is run for every entry in $\mathbf{L}'$. On input $\mathbf{L}'[i]$ algorithm S_1 checks if its input is of the form $0 \| b$. If so, it simply

[2] The (decomposable) randomized encoding of [AIK06] is in NC^0 assuming that pseudorandom generators exist in NC^0. Thus, using this randomized encoding scheme, the pre-computation part could, potentially, also be parallelized. For our application this is, however, not necessary.

ALGO. $\mathsf{S}_0(1^\lambda)$

1 : $\mathbf{L}' \leftarrow [\,]$
2 : $r_{\mathsf{Enc}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{DRE.rl}(\lambda)};\ r_{\mathrm{io}} \leftarrow\!\!\$\ \{0,1\}^{\mathsf{iO.rl}(\lambda)}$
3 : $x \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.il}(\lambda)}$
4 : $\mathsf{aux} \leftarrow x \| 0^{\mathsf{H.ol}(\lambda)} \| r_{\mathrm{io}}$
5 : **for** $i = 1 \ldots s$ **do**
6 : **if** $\mathsf{H.il}(\lambda) < \mathsf{idx}(i) \le \mathsf{H.il}(\lambda) + \mathsf{H.ol}(\lambda)$ **then**
7 : $j \leftarrow \mathsf{idx}(i) - \mathsf{H.il}(\lambda)$
8 : $b_0 \leftarrow \hat{f}_i(0; r_{\mathsf{Enc}})$
9 : $b_1 \leftarrow \hat{f}_i(1; r_{\mathsf{Enc}})$
10 : $\mathbf{L}'[i] \leftarrow 1 \| b_0 \| b_1 \| j \| x$
11 : **else**
12 : $b \leftarrow \hat{f}_i(\mathsf{aux}[\mathsf{idx}(i)]; r_{\mathsf{Enc}})$
13 : $\mathbf{L}'[i] \leftarrow 0 \| b$
14 : **return** $(\varepsilon, \mathbf{L}')$

ALGO. $\mathsf{S}_1^{\text{HASH}}(\mathbf{L}'[i])$

1 : $c \leftarrow \text{MostSigBit}(\mathbf{L}'[i])$
2 : **if** $c = 0$ **then**
3 : parse $\mathbf{L}'[i]$ as $0 \| b$
4 : $L \leftarrow b$
5 : **else**
6 : parse $\mathbf{L}'[i]$ as $1 \| b_0 \| b_1 \| j \| x$
7 : $y \leftarrow\!\!\$\ \text{HASH}(x)$
8 : **if** $y[j] = 0$ **then**
9 : $L \leftarrow b_0$
10 : **else**
11 : $L \leftarrow b_1$
12 : **return** L

Figure 10.4: Pseudocode of the parallel source $\mathsf{S} = \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$.

sets $L \leftarrow b$ and returns L (line 2). Else, it parses $\mathbf{L}'[i]$ as $1 \| b_0 \| b_1 \| j \| x$ and computes $y \leftarrow\!\!\$\ \text{HASH}(x)$. (Note that S_1 has access to oracle HASH.) It then sets $L \leftarrow b_0$ if $y[j] = 0$ and $L \leftarrow b_1$ otherwise. Finally, it outputs L.

By construction, the parallel source $\mathsf{S} := \text{Prl}[\mathsf{S}_0, \mathsf{S}_1]$ computes the same randomized encoding $\hat{f}(x, y, r_{\mathrm{io}}; r_{\mathsf{Enc}})$ that our previous source did. Consequently, by setting the distinguisher D to be identical to that given in the previous attack, we obtain a successful bounded parallel attack. Furthermore, (each instance of) algorithm S_1 makes a single call to oracle HASH and can be implemented by a constant-depth circuit. Finally, and consistently with our observation that L_0 can be routed via the second stage, algorithm S_0 always returns $L_0 = \varepsilon$. □

Putting the claims together, we obtain Theorem 10.1.

Proof of Theorem 10.1. Follows with Claims 10.2, 10.3, 10.5, and Fact 10.4. □

10.2.4 Specifying the Exact Size of S_1

We can specify the size of a circuit that computes S_1, thereby giving a precise lower bound on τ. For this we encode index j (the bit position of y the encoding depends on) by a bitmap. That is, we encode it by a bit string y_{idx} of length $|y_{\mathsf{idx}}| = \mathsf{H.ol}(\lambda)$ that contains a single 1 at the j-th position:

$$y_{\mathsf{idx}} := 0^{j-1} \| 1 \| 0^{\mathsf{H.ol}(\lambda) - j} \ .$$

Furthermore, we let the output of S_1 to be a bit string of length $1 + \mathsf{H.ol}(\lambda)$ rather than a single bit. This string will contain at most a single 1 and hence, in order to recover the correct output bit, the distinguisher must simply compute the logical OR of all the bits in the string.

Consider the following input to S_1

$$d\|b_0\|b_1\|y_{\mathsf{idx}}[1]\|\ldots\|y_{\mathsf{idx}}[\mathsf{H.ol}(\lambda)]\|x[1]\|\ldots\|x[\mathsf{H.ol}(\lambda)]\ ,$$

where we assume that in case it does not depend on y (see line 13 of source part S_0), it is padded with zeros. The circuit computes values $(y[1],\ldots,y[\mathsf{H.ol}(\lambda)]) \leftarrow\!\!\$\ \textsc{Hash}(x)$, and outputs

$$\begin{gathered}\neg d \wedge b_0\ ,\\ (d \wedge y_{\mathsf{idx}}[1]) \wedge \Big((\neg y[1] \wedge b_0) \vee (y[1] \wedge b_1)\Big)\ ,\\ \vdots\\ (d \wedge y_{\mathsf{idx}}[\mathsf{H.ol}(\lambda)]) \wedge \Big((\neg y[\mathsf{H.ol}(\lambda)] \wedge b_0) \vee (y[\mathsf{H.ol}(\lambda)] \wedge b_1)\Big)\ .\end{gathered}$$

The first bit computed above corresponds to line 4 of algorithm S_1 (see Figure 10.4). The remaining bits correspond to the computation of the else clause (line 5). Each of these bits corresponds to testing whether the t-th bit of y needs to be considered (that is, if $j = t$; see line 7 of algorithm S_0) and whether to output b_0 or b_1 if this is the case. Note that, by construction, at most a single bit of the output is set to 1 and to recover the output of our original algorithm S_1 all that the distinguisher needs to do is to compute the logical OR of the bits of the string. Furthermore, we note that delegating the computation of the final OR to the distinguisher does not leak any information, since, given the result of the OR operation, the distinguisher can reconstruct the bit string: If the resulting bit is 0, then the bit string was the all-zero string. Else, the distinguisher only has to evaluate the index function idx of the randomized encoding to reconstruct the position of the single 1 within the bit string.

By counting the operations above, we get that S_1 can be implemented with a single oracle gate, $\mathsf{H.ol}(\lambda)$ many OR gates, $\mathsf{H.ol}(\lambda)+1$ many NOT gates, and $1+4\cdot\mathsf{H.ol}(\lambda)$ AND gates where all the AND and OR gates have fan-in 2. Furthermore, the depth of the circuit is constant and, thus, our implementation of S_1 is in NC^0.

10.3 Split-Source UCEs

A second structural restriction introduced by Bellare, Hoang, and Keelveedhi in [BHK13:1] are split sources (see Section 9.6.2). A split source S is composed of two algorithms S_0 and S_1 where neither algorithm gets access to the Hash oracle. Algorithm S_0 outputs L_0 together with a vector of distinct points $\mathbf{x}$. For each entry of $\mathbf{x}$, the corresponding hash value is computed yielding vector $\mathbf{y}$ of hash values. Algorithm S_1 is then run on $\mathbf{y}$ to produce leakage L_1. The leakage of the split source $\mathsf{S} := \mathrm{Splt}[\mathsf{S}_0,\mathsf{S}_1]$ is set to $L := (L_0, L_1)$. The pseudocode for split sources is given in Figure 10.1 on the right. We say that a source S is in class $\mathcal{S}^{\mathrm{splt}}$ if there exists PPT algorithms S_0 and S_1 such that $\mathsf{S} = \mathrm{Splt}[\mathsf{S}_0,\mathsf{S}_1]$.

Similarly to bounded parallel sources, split sources were designed to be used in combination with computational indistinguishability to work around our iO attack. As for the attack a source needs to know both, value x and answer y, in order to construct an obfuscation of circuit $C[x,y]$ it seems that this might thwart of obfuscation based attacks, even if the leakage computationally

contains queries. Furthermore, BHK show that a such restricted UCE, that is, UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$], is a universal hardcore function (also see Section 9.4.1).

In a very recent paper [BST15], Bellare, Stepanovs, and Tessaro (BST) show that such UCEs cannot exist if indistinguishably obfuscation exists. In more detail, they show that UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}} \cap \mathcal{S}^{\text{q-query}}$] is mutually exclusive with iO. To this end, they show how such UCEs can be used to to construct an MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$]—a multi-bit output point-function obfuscator secure in the presence of computationally hard-to-invert leakage (cf. Definition 6.9 on page 126). Using our negative result for this form of point obfuscation (Theorem 7.3, page 143) one can conclude that UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$] and indistinguishability obfuscation are mutually exclusive.

Remark. In [BST15], Bellare, Stepanovs, and Tessaro recast MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] as a symmetric encryption scheme which allows leakage both on the key and the message (they call their primitive key-message leakage-resilient symmetric-encryption). We note that the two notions are equivalent [CKVW10]. A small difference, introduced by Bellare et al. is that they consider schemes that are not perfectly correct, that is, they require correctness only for random messages and with non-negligible probability. This translates into multi-bit output point obfuscators which may output functions where the point message is not correctly recovered on input the correct point address. As argued by BST [BST15] and discussed in Section 7.2.2 our negative result for MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] carries over also to this more relaxed obfuscation notion.

In the following we present a direct proof showing that UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$] and indistinguishability obfuscation are mutually exclusive. For this, we extracted a UCE adversary, that is, a source and distinguisher, from the BST construction of an MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$] scheme [BST15], and our corresponding negative result (Theorem 7.3, page 143).

Theorem 10.6 ([BST15], Theorem 5.2)**.** *If indistinguishability obfuscation and one-way functions exist, then* UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\text{splt}}$] *security cannot be achieved in the standard model.*

Remember that, in order to show that indistinguishability obfuscation and MB-AIPO with computationally hard-to-invert auxiliary information are mutually exclusive (see Section 7.2), we used a circuit similar to the following

$C[\mathsf{secret}, s](\mathsf{aux})$

$m \leftarrow \mathsf{recoverMessage}(\mathsf{secret}, \mathsf{aux})$
if $\mathsf{PRG}(m) = s$ **then**
  return 1
return 0

The circuit had a point s hard-coded which was simply the image under a PRG of a uniformly random message, that is, $s \leftarrow \mathsf{PRG}(m)$ for $m \leftarrow\!\!\$\, \{0,1\}^\lambda$. In addition the circuit had a secret hard-coded. For the MB-AIPO case this was the point address x. The circuit was designed to take an input aux—in the MB-AIPO case this was a point-function obfuscation—which it uses in combination with secret to recover message m. It then checks if the recover operation succeeded by comparing $\mathsf{PRG}(m)$ to the hard-coded value s.

The idea to attack split sources is to use the above blueprint and leak a circuit that can be computed by a split source. For this, the crux is to come up with a secret and a $\mathsf{recoverMessage}$

algorithm which permits to run the distinguishing operation. What BST exploit is that we can encode a message m relative to oracle HASH bit by bit by querying HASH on value $m[i] \| \langle i \rangle_{\lceil \log(\lambda) \rceil}$ to receive value $\mathbf{y}[i]$. (Here $\langle i \rangle_{\lceil \log(\lambda) \rceil}$ denotes a binary encoding of value i with $\lceil \log(\lambda) \rceil$ bits.) Given answers $\mathbf{y}[i]$ we can recover m bit by bit via the same process, that is, recomputing $\text{HASH}(1 \| \langle i \rangle_{\lceil \log(\lambda) \rceil})$ and comparing the result to $\mathbf{y}[i]$. Of course, a source making such queries would be predictable. To prevent being predictable, BST prepend a secret key k to every query which they hardcode as secret into the above circuit. This allows the circuit to recover m while the source remains unpredictable as k remains hidden. Furthermore, the source can construct the obfuscation of the circuit without having to query oracle HASH allowing it to be split. In the following we make this intuition formal.

Proof of Theorem 10.6. We construct a split source $\mathsf{S} = \text{Splt}[\mathsf{S}_0, \mathsf{S}_1]$ as follows

$\underline{\mathsf{S}_0(1^\lambda)}$

$\mathsf{k} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$m \leftarrow\!\!\$\ \{0,1\}^\lambda$
$s \leftarrow \mathsf{PRG}(m)$
$\overline{C} \leftarrow\!\!\$\ \mathsf{iO}(C[\mathsf{k}, s])$
$L_0 \leftarrow \overline{C}$
for $i = 1, \ldots, \lambda$ **do**
  $\mathbf{x} \leftarrow \mathsf{k} \| m[i] \| \langle i \rangle_{\lceil \log(\lambda) \rceil}$
return $(L_0, \mathbf{x})$

$\underline{\mathsf{S}_1(\mathbf{y})}$

$L_1 \leftarrow \mathbf{y}$
return $\mathbf{y}$

$\underline{C[\mathsf{k}, s](\mathsf{hk}, \mathbf{y})}$

1 : **for** $i = 1, \ldots, \lambda$ **do**
2 : **if** $\mathsf{H.Eval}(\mathsf{hk}, \mathsf{k} \| 1 \| \langle i \rangle_{\lceil \log(\lambda) \rceil}) = \mathbf{y}[i]$ **then**
3 : $m[i] \leftarrow 1$
4 : **else** $m[i] \leftarrow 0$
5 : **if** $\mathsf{PRG}(m) = s$ **then**
6 : **return** 1
7 : **return** 0

We use a pseudorandom generator PRG which is length doubling and note that this is without loss of generality as we assume the existence of one-way functions.

It is easily seen that $\mathsf{S} = \text{Splt}[\mathsf{S}_0, \mathsf{S}_1]$ is split as $\mathbf{x}$ only contains distinct values. Thus, to complete the proof we need to show that there is a distinguisher such that (S, D) win in the UCE game with good probability and that, furthermore, source S is computationally unpredictable. The latter follows directly from Lemma 7.4 (page 145), that is, it follows using the PRG trick, that we have seen now several times. For this note that circuit $C[\mathsf{k}, s]$ and $C[\mathsf{k}, r]$ for s being chosen as by S_0 and r being chosen uniformly at random in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ are computationally indistinguishable down to the security of the PRG. As we furthermore assume PRG to be length-doubling we have that $C[\mathsf{k}, r]$ is functionally equivalent to the all-zero circuit with high probability: with overwhelming probability a random value $r \in \{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ is outside the image of the PRG and, thus, circuit $C[\mathsf{k}, r]$ is 0 on every input as the test in line 5 always evaluates to false. The proof then follows down to the security of the indistinguishability obfuscator and noting that the probability of guessing secret key k is negligible. Hence, $\mathsf{S} \in \mathcal{S}^{\text{cup}}$ that is source S is computationally unpredictable.

It remains to specify distinguisher D. Distinguisher D takes as input hash key hk and leakage $L = (L_0, L_1) = (\overline{C}, \mathbf{y})$. It runs $\overline{C}$ on input hk and vector $\mathbf{y}$ and outputs whatever the circuit returns:

$\underline{\mathsf{D}(\mathsf{hk}, L)}$

$(\overline{C}, \mathbf{y}) \leftarrow L$
$b' \leftarrow \overline{C}(\mathsf{hk}, \mathbf{y})$
return b'

In case $b = 1$ and oracle HASH implements hash function H then distinguisher D outputs 1 with probability 1. On the other hand, in case $b = 0$ we can bound the probability of D outputting 1 by the security of the pseudorandom generator. For this note that now all entries in $\mathbf{y}$ are uniformly random strings in $\{0,1\}^{\mathsf{H.ol}(\lambda)}$. We construct a PRG adversary $\mathcal{A}$ that takes as input a value s which is either a uniformly random string in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ or which is sampled as the PRG image for a uniformly random seed in $\{0,1\}^{\mathsf{PRG.il}(\lambda)}$. Adversary $\mathcal{A}$ operates as follows:

$\underline{\mathcal{A}(1^\lambda, s)}$
$\mathsf{k} \leftarrow\!\!\$\ \{0,1\}^\lambda$
$\overline{C} \leftarrow\!\!\$\ \mathsf{iO}(C[\mathsf{k}, s])$
for $i = 1, \ldots, \lambda$ **do**
  $\mathbf{y}[i] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
$\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.KGen}(1^\lambda)$
$b' \leftarrow\!\!\$\ \mathsf{D}(\mathsf{hk}, (\overline{C}, \mathbf{y}))$
return b'

In case s is chosen uniformly at random in $\{0,1\}^{\mathsf{PRG.ol}(\lambda)}$ then adversary $\mathcal{A}$ outputs 1 only with negligible probability as circuit $C[\mathsf{k}, s]$ is with overwhelming probability the all-zero circuit. That is,

$$\Pr\left[\mathcal{A}(1^\lambda, s) = 1 \ :\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}\right] \leq \mathsf{negl}(\lambda)\,.$$

On the other hand, if s is chosen as $\mathsf{PRG}(m)$ for a uniformly random $m \in \{0,1\}^{\mathsf{PRG.il}(\lambda)}$ then adversary $\mathcal{A}$ perfectly simulates the UCE game (with hidden bit $b = 0$) for distinguisher D. We can rewrite advantage $\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{PRG},\mathcal{A}}(\lambda)$ as

$$\left|\Pr\left[\mathcal{A}(1^\lambda, s) = 1 \ :\ s \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.ol}(\lambda)}\right] - \Pr\left[\mathcal{A}(1^\lambda, s) = 1 \ :\ m \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.il}(\lambda)};\ s \leftarrow \mathsf{PRG}(m)\right]\right|$$

which, by assumption, is negligible. We, thus, have that also

$$\Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1 \;\middle|\; b = 0\right] = \Pr\left[\mathcal{A}(1^\lambda, s) = 1 \ :\ m \leftarrow\!\!\$\ \{0,1\}^{\mathsf{PRG.il}(\lambda)};\ s \leftarrow \mathsf{PRG}(m)\right]$$

must be negligible, which concludes the proof. □

10.4 Conclusion

While we have seen that indistinguishability obfuscation and UCEs based on computational unpredictable sources do not seem to go well together we want to close this chapter on a positive note. The negative result for split sources leaves open one loophole that we will exploit in the next chapter. The number of oracle queries made by source S_0 in the above proof corresponds to the number of bits in message m and is, thus, linear in the security parameter. In fact, we can do better and increase the size of m only sublinear, but for the PRG argument we need the size to grow super-logarithmically in the security parameter. However, the only known application that uses split source UCEs is that any UCE$[\mathcal{S}^{\mathrm{cup}} \cap \mathcal{S}^{\mathrm{splt}} \cap \mathcal{S}^{\text{1-query}}]$ secure function is a universal hardcore function (cf. Section 9.4.1).

Let us emphasize that for this it is sufficient to consider split source UCEs that are resistant against a single (i.e. constantly many) queries. In the following chapter we will show how to construct such functions from indistinguishability obfuscation and AIPOs.

Chapter 11

Constructing UCEs in the Standard Model

> "Achieving UCE under other assumptions is an interesting and important direction for future work. We suggest to begin by targeting restricted versions of UCE, starting with independent sources (ones whose oracle queries consist of uniform, independent strings) and moving on to block sources (each oracle query retains high min-entropy even given previous ones) [53]. In these cases, we may hope to achieve security under first-degree assumptions. [...] Full UCE security would, of course, require second-degree assumptions."
>
> Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi [BHK13:2]

Summary. In this chapter we show how to construct various forms of UCEs based on indistinguishability obfuscation, puncturable PRFs and point-function obfuscation. Our constructions yield the first standard model constructions for various cryptographic primitives that previously only had constructions in the random oracle model. These include: universal hardcore functions, deterministic public-key encryption and correlated-input secure hash functions (we give an overview in Table 11.1). The results in this chapter are based on [BM14c] and manuscript [BM15b].

Chapter content

11.1 Introduction

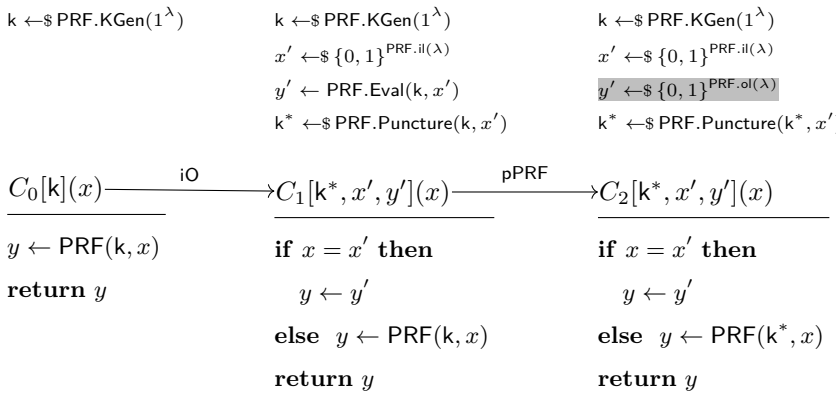
We have seen constructions for the strongest UCE notions in the random oracle model (Section 9.3) but, assuming the existence of indistinguishability obfuscation we know that such strong UCEs—UCEs with respect to *computationally unpredictable sources*—do not exist. As we have shown, this negative result does not only apply to the unrestricted version, i.e., UCE[$\mathcal{S}^{\text{cup}}$] but also holds for various restrictions such as bounded parallel or split sources (Theorems 10.1 and 10.6).

In this chapter we ask what forms of UCEs we can build from strong assumptions (such as obfuscation based assumptions) in the standard model. When employing obfuscation techniques, a

natural candidate to replace random oracles is an obfuscated pseudorandom function with a hard-coded key. Indeed, our hash-function construction will consist only of a puncturable pseudorandom function that is obfuscated via an indistinguishability obfuscator:

Hash Construction: $\mathsf{iO}(\mathsf{PRF}(\mathsf{k},.))$.

Puncturable pseudorandom functions (see Section 5.5.5 for an introduction) have become a de-facto standard technique for constructions based on indistinguishability obfuscation. One reason is that they allow to mimic some of the features of random oracles, in particular, they allow for a restricted programming of the construction which is called the *punctured programs technique* and which was introduced by Sahai and Waters [SW14]. Using puncturable (or more generally, constrained) pseudorandom functions within an obfuscated program allows to change the behavior of the program at key points. An example of the basic puncturable program's technique is given by the following three circuits where the first block describes the sampling of keys and values x' and y':



Here the first two circuits are functionally equivalent and, thus, indistinguishability-obfuscations of C_0 and C_1 are computationally indistinguishable. Then, in a second step value y' is sampled at random and down to the security of the puncturable PRF, the circuits C_1 and C_2 are computationally indistinguishable (even without obfuscation). The result is that in C_2 we have some control over the image the circuit returns on input x', that is, we can program the circuit on that input.

There are two major differences to programming a random oracle. On the one hand, puncturing and, thus, programming needs to be performed before obfuscation and, thus, usually upfront because as soon as the obfuscated program is given to the adversary no more changes to the program can be introduced. Random oracles, on the other hand, can be programmed adaptively (cf. Section 3.3.2) as any point of the oracle that has not yet been queried remains hidden from the adversary and is completely undetermined by any of the queries seen by the adversary. A second and major difference is that using a standard-model function means that the adversary is given the code of the function and that programming may affect the size of the resulting program. Consider again our first two circuits from the previous example:

CIRCUIT $C_0[\mathsf{k}](x)$	CIRCUIT $C_1[\mathsf{k}^*, x', y'](x)$
$y \leftarrow \mathsf{PRF}(\mathsf{k}, x)$	**if** $x = x'$ **then**
return y	$y \leftarrow y'$
	else $y \leftarrow \mathsf{PRF}(\mathsf{k}^*, x)$
	return y

We earlier said that under indistinguishability obfuscation the two circuits are computationally indistinguishable. This is, however, only half true as we explain next. It is easily seen that circuit C_1 needs additional space to allow for the extra if-branch. In particular, if values x' and y' are chosen uniformly at random then these cannot be compressed and, thus, $|C_1| \geq |C_0| + |x'| + |y'|$. If now, we wanted to use indistinguishability obfuscation to argue that obfuscations of the two circuits are indistinguishable we are stuck because obfuscation cannot hide the size of the circuit and, thus, there exists a trivial distinguisher which distinguishes merely based on the size of the code that it is given.

To work around such size differences we need to artificially increase the size of C_0 such that then obfuscations of the extended C_0 and C_1 are of the same size. For this we can introduce sufficiently many NOP (no-operations) into C_0 (for example, pairs of NOT operations which cancel one another out) until we get to the following picture where the size of the two circuits is identical.

CIRCUIT $C_0[\mathsf{k}, \_, \_](x)$	CIRCUIT $C_1[\mathsf{k}^*, x', y'](x)$
_____	**if** $x = x'$ **then**
_____	$y \leftarrow y'$
$y \leftarrow \mathsf{PRF}(\mathsf{k}, x)$	**else** $y \leftarrow \mathsf{PRF}(\mathsf{k}^*, x)$
return y	**return** y

We briefly discussed the necessity of padding when introducing techniques to work with indistinguishability obfuscation in Section 5.5.5. Also in all previous (negative) results we needed to take padding into account. There, however, we only used the *zero circuit technique* and, hence, padding was mostly used to ensure that a zero circuit was padded sufficiently to be of the same size as some other, given, circuit. For many positive construction, and this will also be the case for us, the role of padding is much more prominent. The reason is that security proofs of the construction may need to perform puncturing steps which depend on the adversary, that is, depending on the adversary a different number of puncturing steps need to be performed. However, as during the course of the proof the basic construction is transformed step by step to incorporate the puncturings we need to artificially increase the size of the base construction to allow for these transformations. What is worse is that we need to specify a fixed amount of padding for the construction and, thus, the construction may become insecure against adversaries who would have needed additional padding for the security proof to go through.

Remark. It is interesting to note that in case we do not perform padding we do not necessarily end up with insecure constructions but our proof techniques do no longer work. That is, not using padding may not lead to attacks—we do not know how or if this can be exploited—but we also have no more guarantees.

We will get back to the question of padding and discuss the necessity for padding in greater detail in Chapter 13. For the purpose of constructing hash functions from obfuscation this, however,

means that we need a slightly more complicated construction than directly obfuscating a PRF. We obfuscate a padded PRF:

$$\textbf{Hash Construction: } \mathsf{iO}(\mathsf{PAD}(s(\lambda), \mathsf{PRF}(\mathsf{k}, .)) \ .$$

Here the deterministic function PAD takes as input an integer and a circuit description C and outputs a circuit description C' of size $|C| + s(\lambda)$ which is functionally equivalent to circuit C.

Before we turn back to UCEs we note that similar constructions, that is, an obfuscated (padded) puncturable pseudorandom function has been used by Bellare, Stepanovs, and Tessaro [BST14] to construct hardcore functions from any one-way function, by Hohenberger, Sahai and Waters [HSW14] to instantiate the random oracle in full-domain hash, as well as by Canetti, Chen, and Reyzin [CCR15] who study and construct correlation intractable function ensembles.

Remark. The role of padding in obfuscation based constructions is particularly prominent in the construction of Bellare, Stepanovs, and Tessaro [BST14] which they show is hardcore for any injective one-way function if padded appropriately. In particular, the padding depends on the one-way function. For their result they rely on indistinguishability obfuscation and puncturable pseudorandom functions. Interestingly, under the same assumptions and additionally assuming the existence of strong point obfuscation (i.e., AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]) we can show that the above construction with a *fixed* padding is hardcore for *any* one-way function. Thus, in some cases it seems that padding can be removed down to additional assumptions. We discuss such a *universal* assumption that allows to remove padding from certain iO based constructions in Chapter 13.

11.1.1 Constructing UCEs

In this chapter we show that the above construction achieves two forms of UCE security. On the one hand, we show that the construction is mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] secure, that is, UCE secure with respect to sources that are *statistically* unpredictable and make at most q many queries. The polynomial q is needed for key generation and is due to the padding problematic discussed above. Even though we do not achieve full mUCE[$\mathcal{S}^{\text{sup}}$] security but only q-bounded security our result yields the first standard model construction for a variety of primitives, including, q-query correlated input secure hashing and deterministic public-key encryption, as well as instantiations of the BRS scheme [BRS03] to obtain key-dependent message and related-key security. We note that for the latter two applications the q-bound is sufficient as these only require UCEs secure against a single query.[1] We provide an overview of known applications that can be obtained from mUCE[$\mathcal{S}^{\text{sup}}$] security in Table 11.1.

While we can show that the construction yields mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$]-security, we can show that it simultaneously achieves UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]-security, yet under different assumptions. That is, the construction is a secure UCE with respect to *computationally strongly* unpredictable sources that make at most a single query. This complements our previous negative results, that is, we show that UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] security cannot be achieved (Section 9.7.2) and, thus, our positive result, in some sense, is the best we can hope for in terms of *computational unpredictability*. As shown by Bellare, Hoang, and Keelveedhi [BHK13:2] and noting that split sources are a strict subclass of strongly computationally restricted sources, functions that are UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] are universal

[1] The reason for this is that for each encryption a fresh hash key is chosen.

UCEs for strongly computationally unpredictable sources		UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]
HC	UCEs are hardcore for any one-way function.	[BHK13:2]
BR93	UCEs are sufficient to instantiate the BR93 PKE scheme [BR93].	[BHK13:2]

implies

UCEs for strongly statistically unpredictable sources		UCE[$\mathcal{S}^{\text{s-sup}}$]
CIH	UCEs are correlated input secure hash functions.	[BHK13:2]
STORE	Such UCEs can instantiate the proof of storage scheme from [RSS11].	[BHK13:2]
IMMU	UCEs can be used to construct secure immunizers as countermeasure for backdoored PRGs.	[DGG+15]

implies

UCEs for statistically unpredictable sources		UCE[$\mathcal{S}^{\text{sup}}$]
MLE	UCEs can instantiate message-locked encryption schemes.	[BHK13:2]
D-PKE	UCEs can instantiate deterministic PKE schemes.	[BH15]
CCA	UCEs can be used to build CCA secure PKE schemes.	[MH14b]
Multi-key UCEs for statistically unpredictable sources		mUCE[$\mathcal{S}^{\text{sup}}$]
KDM	UCEs can instantiate the BRS scheme [BRS03] to obtain a standard-model symmetric encryption scheme which offers security against key-dependent messages.	[BHK13:2]
RKA	The same instantiation as for KDM can be shown to be also secure against related-key attacks.	[BHK13:2]
PFOB	UCEs can be used to instantiate the RO point-function obfuscation scheme due to Lynn et al. [LPS04].	[BHK13:2]

Standard model constructions for UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] and mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}$] in this chapter.

UCEs for statistically reset-secure sources		UCE[$\mathcal{S}^{\text{srs}}$]
GB	UCEs can be used to obtain an adaptive garbling scheme.	[BHK13:2]
HPKE	UCEs can be used to instantiate a PKE scheme which hedges against bad randomness.	[BH15]

Table 11.1: A list of UCE notions and corresponding applications. In this chapter we show how to construct a UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] (used for the first block) as well as a mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}$] secure function (used for blocks two and three). We note that for the applications HC, KDM, RKA and PFOB we only needs UCEs which are secure with respect to sources making a single query and, thus, there the q-query (resp. 1-query) restriction does not infringe applicability.

hardcore functions and, furthermore, are sufficient to instantiate the random oracle in the BR93 public-key encryption scheme.

Remark. For an overview of the various applications see Chapter 4 and the references given in Table 11.1.

11.1.2 Techniques

Although the construction that uses an obfuscated pseudorandom function is natural, proving its security is non-trivial. For this note that various applications of such UCEs (for example, correlated-input secure hash functions) are defined via a two-stage game and are thus, potentially, subject to Wichs' simulatable inefficient attacks [Wic13]. Wichs shows that security for various two-stage primitives cannot be reduced to standard (single-stage) cryptographic assumption such as factoring, discrete log, DDH, LWE, etc. In particular, this is the case for correlation input secure hash functions

and deterministic public-key encryption. We discuss his result in the context of correlation input security in Section 4.2.2.

Our construction is an indistinguishability obfuscation of a puncturable pseudorandom function and, thus, security is down to iO and puncturable PRFs. As both assumptions are single-stage—note that an equivalence circuit sampler can freely share state with the distinguisher and similarly the two parts of a puncturable PRF adversary can freely share state—they are not sufficient to bypass Wichs' result. Towards proving the security of our construction we, therefore, need to include an additional assumption. In our case this assumption is the existence of point-function obfuscators that secure under auxiliary input (AIPO). The security game of AIPOs (cf. Definition 6.6) is a two-stage game and, thus, potentially able to work around the impossibility result by Wichs [Wic13].

Remark. It is interesting to note that point obfuscators do not appear in the construction but only within the proof. Thus, it might be possible that the same construction can be proven secure through some other two-stage assumption and without making use of AIPOs.

In order to prove that a hash function is UCE[$\mathcal{S}$] secure we need to show that the leakage of any source $\mathsf{S} \in \mathcal{S}$ generated with oracle access to the hash function is computationally indistinguishable from the leakage generated with oracle access to a random oracle. This even needs to hold in case the distinguisher additionally gets to see the hash key (or a uniformly random key). In other words, if $\mathsf{H}(\mathsf{hk}, \cdot)$ is a hash function with random key and RO is a random oracle we need to prove that

$$(\mathsf{hk}, \mathsf{S}^{\mathsf{H}(\mathsf{hk},\cdot)}(1^\lambda)) \approx_c (\mathsf{hk}, \mathsf{S}^{\mathsf{RO}}(1^\lambda)) \tag{11.1}$$

for any source S in $\mathcal{S}$. To this end, we start with the left distribution and transform it via a series of steps into the right distribution. Note that our hash function is an obfuscated puncturable PRF, that is, $\mathsf{H}(\mathsf{hk}, \cdot)$ is, in fact, an obfuscation of the following circuit

$C_0[\mathsf{k}](x)$

$y \leftarrow \mathsf{PRF.Eval}(\mathsf{k}, x)$

return y

where k is a uniformly random PRF key.

Starting on the left hand side of Equation 11.1, in a first step, we apply the punctured programs technique of Sahai and Waters [SW14]. For this, we record the queries the source makes to its oracle (which implements the hash function as we started with the left hand distribution) as well as the answers. Let (x_i^*, y_i^*) denote the i-th query answer pair and note that we only consider a bounded number of queries (at most q). We can, thus, change our circuit and hardcode these query answer pairs to obtain:

$C_1[\mathsf{k}^*, (x_i^*, y_i^*)](x)$

for $i = 1, \ldots, \mathsf{q}$ **do**

  if $x_i^* = x$ **then**

    return y_i^*

$y \leftarrow \mathsf{PRF.Eval}(\mathsf{k}^*, x)$

return y

Note that we have syntactically changed the circuit, but as the oracle answers were generated by the PRF with key k we have not changed the functionality. In addition we exchanged key k in the circuit by a punctured key k^* which is punctured on all queries x_i^*. That is, using k^* one cannot compute the PRF value on x_i^*. As we have hard-coded the corresponding values into the circuit this change, however, does not change the functionality of our circuit. Hence, with appropriate padding the two circuits C_0 and C_1 are indistinguishable under indistinguishability obfuscation.

Next, we use the security of the puncturable PRF to embed random values instead of the correct PRF images. To this end, we let the oracle answer with random values y_i^*, and as before embed them in the circuit. Note that while this changes the circuit one can prove down to the security of the puncturable PRF that such a change cannot be detected as we only replaced values on the punctured out points.

This concludes the application of the punctured programs technique and this is where we introduce a twist (i.e., point obfuscation). An obfuscation of circuit C_1 may leak the query points, or in other words, we need a good argument to argue otherwise. We use point obfuscations and instead of hard-coding values x_i^* we encode point obfuscations $\overline{p}_i$. In addition we change the punctured key k^* back to the original key k. Consequently the circuit changes to:

$C_2[\mathsf{k}, (\overline{p}_i, y_i^*)](x)$

for $i = 1, \ldots, \mathsf{q}$ **do**
  if $\overline{p}_i(x) = 1$ **then**
    return y_i^*
$y \leftarrow \mathsf{PRF.Eval}(\mathsf{k}, x)$
return y

Again, we only changed the syntax but not the function itself and, thus, the change remains indistinguishable down to iO. Note that, at this point we have changed the oracle of source S to answer randomly and, thus, have so far shown the indistinguishability of the following distributions:

$$\left(\mathsf{iO}(C_0[\mathsf{k}]), \mathsf{S}^{\mathsf{PRF.Eval}(\mathsf{k},\cdot)}(1^\lambda)\right) \qquad \text{and} \qquad \left(\mathsf{iO}(C_2[\mathsf{k}, (\overline{p}_i, y_i^*)]), \mathsf{S}^{\mathsf{RO}}(1^\lambda)\right).$$

In order to complete the proof, we need to change our function back to the original, that is, we need to again obtain circuit C_0:

$C_0[\mathsf{k}](x)$

$y \leftarrow \mathsf{PRF.Eval}(\mathsf{k}, x)$
return y

Circuits C_0 and C_2 are, however, not functionally equivalent: they differ on exactly the query points x_i^*. In order to transform circuit C_2 back into C_0 we, thus, require a stronger assumption than indistinguishability obfuscation, namely, differing-inputs obfuscation. Intuitively, the use of point obfuscations for points x_i^* should make it difficult to find differing inputs for circuits C_2 and C_0. Hence, obfuscations of the two are indistinguishable in case the obfuscator is a differing-inputs obfuscator. Here is where we rely on the beautiful result of Boyle, Chung, and Pass [BCP14] (see Theorem 5.13 in Section 5.6.1) who show that any general-purpose indistinguishability obfuscator is also a mildly secure differing-inputs obfuscator. In more detail, they show that any indistinguishability obfuscator

is also differing-inputs secure for circuits that differ on at most polynomially many points. For us, this is the case: C_0 and C_2 differ on at most q many points and, thus, we can move back to C_0 which concludes the proof.

In the above sketch we skipped over many details, in particular, we left open what form of point obfuscation is required for the result. We note that this choice is crucial for the strength of the resulting UCE function. If we choose an AIPO$[\mathcal{S}^{\text{cup}}_{\mathsf{po}}]$ we obtain a UCE$[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}]$ secure function. Note that here we need to restrict the number of queries by the source to a constant as we have shown in Chapter 7 that such AIPOs cannot be composed. Note also that we only obtain resistance against *strong* computationally unpredictable sources in accordance with our negative result for (plain) computationally unpredictable sources (see Theorem 9.6 on page 201). This restriction is necessary to argue the final game hop from circuit C_2 back to circuit C_0.

If we, on the other hand, chose a (composable) AIPO$[\mathcal{S}^{\text{sup}}_{\mathsf{po}}]$, that is, a point obfuscator secure in the presence of *statistically* hard-to-invert leakage we obtain a mUCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$-secure function. We next make our ideas formal.

11.2 Constructing UCEs for Statistically Unpredictable Sources

In this section we present our construction and show that it yields a UCE$_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ secure function, that is, we do not yet consider multiple keys, but first present the proof for the single keyed version as this admits for simpler notation. We then, in Section 11.3 extend the result to show that it similarly holds for the multi-key version. Also note that we only show security if the hash function is such that it only outputs a single bit. In order to obtain a secure function with larger output, i.e., a UCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ secure function, one needs to run the construction in counter mode as described in Section 9.7.1.

We begin with presenting the construction and subsequently prove its security.

11.2.1 The Construction

We construct a UCE hash function by obfuscating a puncturable pseudorandom function with a general-purpose indistinguishability obfuscator. Thus, a hash key will be an obfuscated program and the hash function itself will simply "evaluate the hash key". Key generation of our construction crucially depends on a polynomial q which describes the number of queries a source is allowed to make. This is needed for padding the circuit before obfuscation to allow for the transformations within the security proof. The amount of padding necessary for each query depends on the choice of puncturable PRF and the point obfuscation scheme used in the proof.

Construction 11.1. *Let* $\mathsf{q} : \mathbb{N} \to \mathbb{N}$ *and* $s : \mathbb{N} \to \mathbb{N}$ *be polynomials. Let* F *be a puncturable PRF and let* iO *be an indistinguishability obfuscator for all circuits in* P/poly*. We define our hash function family* H *as*

$\mathsf{H.KGen}(1^\lambda)$	$\mathsf{H.Eval}(\mathsf{hk}, x)$
$\mathsf{k} \leftarrow\!\!\$\ \mathsf{F.KGen}(1^\lambda)$	$\overline{C} \leftarrow \mathsf{hk}$
$\overline{C} \leftarrow\!\!\$\ \mathsf{iO}(\mathsf{PAD}(s(\lambda), \mathsf{F.Eval}(\mathsf{k}, \cdot)))$	**return** $\overline{C}(x)$
$\mathsf{hk} \leftarrow \overline{C}$	
return hk	

Here, $\mathsf{PAD} : \mathbb{N} \times \{0,1\}^* \longrightarrow \{0,1\}^*$ *denotes a deterministic padding algorithm that takes as input an integer and a description of a circuit* C *and outputs a functionally equivalent circuit padded to length* $|C| + s(\lambda)$*. Function* s *needs to be chosen in accordance with the puncturable PRF and a point obfuscation scheme* PO *to allow for puncturing* F *on* q *points and embedding* q *many point obfuscations within circuit* $\overline{C}$*.*

11.2.2 Construction 11.1 is UCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ Secure

In the following we show that the above construction when implemented with a puncturable PRF that has a single output bit yields a secure $\text{UCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ function. (Recall that UCE_1 denotes a UCE function which only has a single output bit.)

Construction 11.2. *The construction is identical to Construction 11.1 with the exception that PRF* F *has an output length of* $\mathsf{F.ol}(\lambda) = 1$*.*

Theorem 11.1. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable AIPO*$[\mathcal{S}^{\text{sup}}_{\mathsf{po}}]$*) exist then Construction 11.2 is* $\text{UCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ *secure.*

Proof. We prove the theorem via a sequence of 5 games (depicted in Figure 11.1) where Game_1 denotes the original $\text{UCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ game with hidden bit b fixed to 1. We first present the games and subsequently the analysis of the individual game hops. Let $\mathsf{S} \in \mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$. Without loss of generality we assume that source S does not repeat queries to its oracle HASH.

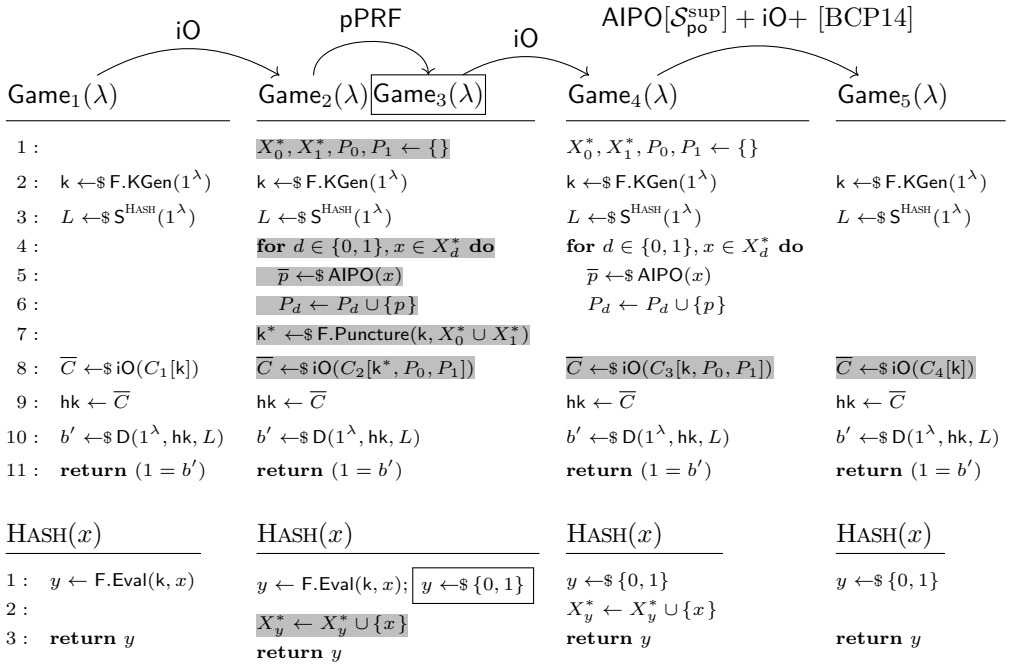
$\mathsf{Game}_1(\lambda)$**:** The first game is the original $\text{UCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ game with hidden bit b set to 1. Here, the hash key hk is an obfuscation of the padded circuit $C_1[\mathsf{k}](x) := \mathsf{F.Eval}(\mathsf{k}, x)$ where k is a key for the puncturable PRF.

iO

$\mathsf{Game}_2(\lambda)$**:** Let X_0^* denote the queries by source S to its HASH oracle that were answered with 0 and let X_1^* denote the queries answered with 1. Game_2 is identical to Game_1 with the exception that circuit $C_2[\mathsf{k}^*, P_0, P_1]$ is obfuscated instead of circuit $C_1[\mathsf{k}]$. Here k^* is the punctured key for set $X_0^* \cup X_1^*$ (i.e., punctured on all of the queries made by S). Sets P_d contain point obfuscations for the queries in X_d. Finally, circuit $C_2[\mathsf{k}^*, P_0, P_1]$ is functionally equivalent to $C_1[\mathsf{k}]$. On input x it first checks whether any point obfuscation in P_d outputs 1 on x (note this can be at most one). If so it outputs d. Otherwise it uses the punctured key to output $\mathsf{F.Eval}(\mathsf{k}^*, x)$. Note that in this case $x \notin X_0^* \cup X_1^*$ and hence $\mathsf{F.Eval}(\mathsf{k}^*, x) = \mathsf{F.Eval}(\mathsf{k}, x)$.

pPRF

$\mathsf{Game}_3(\lambda)$**:** The game is equivalent to Game_2 except that oracle HASH now samples y uniformly at random instead of invoking $\mathsf{F.Eval}(\mathsf{k}, \cdot)$.



CIRCUIT C_1[k](x)
return F.Eval(k, x)

CIRCUIT C_2[k*, P_0, P_1](x)
for $d \in \{0,1\}, \overline{p} \in P_d$ do
if $\overline{p}(x) = 1$ then
return d
return F.Eval(k*, x)

CIRCUIT C_3[k, P_0, P_1](x)
for $d \in \{0,1\}, \overline{p} \in P_d$ do
if $\overline{p}(x) = 1$ then
return d
return F.Eval(k, x)

CIRCUIT C_4[k](x)
return F.Eval(k, x)

Figure 11.1: A pseudocode description of the game steps in Theorem 11.1. The boxed Game$_3$ is to be understood with the boxed statement included. Furthermore the highlighted lines are emphasize the difference to the previous game.

iO

Game$_4(\lambda)$**:** The game is equivalent to the previous game except that we now use an obfuscation of circuit C_3[k, P_0, P_1]. The circuit is functionally equivalent to C_2[k*, P_0, P_1] as the puncturable PRF is only called on values that were not punctured out.

AIPO[$\mathcal{S}^{\text{sup}}_{\text{po}}$] + iO + [BCP14]

Game$_5(\lambda)$**:** The game is equivalent to the previous game except that now an obfuscation of circuit C_4[k] is used as hash key. Circuit C_4[k] is our original circuit again, that is, C_4[k]$(\cdot)$:= F.Eval(k, $\cdot$). Game$_5$ is our intended target. It is the UCE-security game for our construction in the random oracle world (that is, oracle HASH implements a random oracle).

In Game$_5$ we have reached the target setting, i.e., the UCE-game with the hidden bit set to 0 while Game$_1$ denotes the real UCE-game with the hidden bit set to 1. Thus, we can write the advantage

of an adversary (S, D) in the UCE-security game as

$$\begin{aligned}\mathsf{Adv}^{\mathrm{uce}}_{\mathsf{S},\mathsf{D},\mathsf{H}}(\lambda) &= \Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1 \;\middle|\; b = 1\right] + \Pr\left[\mathrm{UCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda) = 1 \;\middle|\; b = 0\right] - 1 \\ &= \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{1}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{5}(\lambda) = 1\right] \\ &\leq \sum_{i=1}^{4} \left|\Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{i}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{i+1}(\lambda) = 1\right]\right|\end{aligned}$$

In the following we show that the individual games are negligibly close.

$\mathsf{Game}_1(\lambda)$ to $\mathsf{Game}_2(\lambda)$. In order to reduce to the security of the indistinguishability obfuscator iO, we show that, by construction, the circuits $C_1[\mathsf{k}]$ and $C_2[\mathsf{k}^*, P_0, P_1]$ compute the same function. If $\overline{p}(x) = \perp$, then $C_2[\mathsf{k}^*, P_0, P_1]$ returns $\mathsf{F.Eval}(\mathsf{k}^*, x)$. If $\overline{p}(x) = 1$ for some $p \in P_d$ then d is returned. Note that in this case $\overline{p}$ was placed in P_d because $\mathsf{F.Eval}(\mathsf{k}, x) = d$. Hence, on all inputs x, $C_2[\mathsf{k}^*, P_0, P_1]$ returns $\mathsf{F.Eval}(\mathsf{k}, x)$ and so does $C_1[\mathsf{k}]$. Having established the functional equivalence between the two circuits we can bound the difference between games Game_1 and Game_2 by the distinguishing advantage against the indistinguishability obfuscator iO. We now formalize this intuition.

We consider a circuit sampler Sam which runs the steps of Game_2 up to and including line 7. Sampler Sam then outputs (the functionally equivalent) circuits $C_1[\mathsf{k}]$ and $C_2[\mathsf{k}^*, P_0, P_1]$ and auxiliary information $\mathsf{aux} \leftarrow L$. Obfuscation distinguisher D_{iO} gets as input aux and an obfuscated circuit $\overline{C}$ which is either an obfuscation of $C_1[\mathsf{k}]$ or of $C_2[\mathsf{k}^*, P_0, P_1]$. It sets $\mathsf{hk} \leftarrow \overline{C}$, $L \leftarrow \mathsf{aux}$ and runs $\mathsf{D}(1^\lambda, \mathsf{hk}, L)$. It outputs whatever D outputs.

If $C = C_1[\mathsf{k}]$ then adversaries $(\mathsf{Sam}, \mathsf{D}_{\mathsf{iO}})$ perfectly simulate game $\mathsf{Game}_1(\lambda)$ and if $C = C_2[\mathsf{k}^*, P_0, P_1]$ then the adversaries simulate $\mathsf{Game}_2(\lambda)$. Thus, we can rewrite the difference between the two games' distributions as:

$$\left|\Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{1}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{2}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}_{\mathsf{iO}}}(\lambda).$$

$\mathsf{Game}_2(\lambda)$ to $\mathsf{Game}_3(\lambda)$. We reduce the difference between Game_2 and Game_3 to the security of the puncturable PRF F. We define an adversary $(\mathcal{A}_1, \mathcal{A}_2)$ against the puncturable PRF as follows. Adversary $\mathcal{A}_1$ runs source $\mathsf{S}(1^\lambda)$ on the security parameter answering its queries to HASH with its own oracle CHALLENGE. It records the queries of S in sets X_0^* and X_1^* (as in procedure HASH) storing queries that were answered with a 0 in X_0^* and queries that were answered with 1 in X_1^*. When S stops it records leakage L. It then executes lines 4 to (but not including) line 7 of Game_2. It then stops and outputs $\mathsf{state} \leftarrow (L, P_0, P_1)$. Adversary $\mathcal{A}_2$ is then run on input state and the punctured key k^*. It parses $(L, P_0, P_1) \leftarrow \mathsf{state}$ and constructs circuit $C_2[\mathsf{k}^*, P_0, P_1]$. It sets $\mathsf{hk} \leftarrow\!\!\$\, \mathsf{iO}(C_2[\mathsf{k}^*, P_0, P_1])$ and runs distinguisher D on input $(1^\lambda, \mathsf{hk}, L)$. It outputs whatever D outputs.

If the challenge oracle answers honestly using the puncturable PRF F then adversary $(\mathcal{A}_1, \mathcal{A}_2)$ perfectly simulates Game_2 and otherwise it perfectly simulates Game_3. Thus, we have that

$$\left|\Pr\left[\mathsf{Game}_2(\lambda)^{\mathsf{S},\mathsf{D}} = 1\right] - \Pr\left[\mathsf{Game}^{\mathsf{S},\mathsf{D}}_{3}(\lambda) = 1\right],\right| \leq \mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F},\mathcal{A}_1,\mathcal{A}_2}(\lambda)$$

which by the security of the puncturable PRF F is negligible.

$\mathsf{Game}_3(\lambda)$ to $\mathsf{Game}_4(\lambda)$. As circuits $C_2[\mathsf{k}^*, P_0, P_1]$ and $C_3[\mathsf{k}, P_0, P_1]$ use key k (resp. k^*) only on values $x \notin X_0^* \cup X_1^*$ the two circuits are functionally equivalent. An analogous analysis as from Game_1 to Game_2 yields that

$$\left|\Pr\left[\mathsf{Game}_3^{\mathsf{S},\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_4^{\mathsf{S},\mathsf{D}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}_{\mathsf{iO}}}(\lambda).$$

$\mathsf{Game}_4(\lambda)$ to $\mathsf{Game}_5(\lambda)$. By construction, the circuits $C_3[\mathsf{k}, P_0, P_1]$ and $C_4[\mathsf{k}]$ only differ on points $x \in X_0^* \cup X_1^*$. We will bound the difference between games Game_4 and Game_5 by the differing-inputs security of the indistinguishability obfuscator iO. For this, we build on a result by Boyle, Chung, and Pass (given as Theorem 5.13 on page 92) who show that any indistinguishability obfuscator is also a differing-inputs obfuscator for differing-inputs circuits which differ on at most polynomially many points [BCP14]. As explained above, our circuits can differ only on points $x \in X_0^* \cup X_1^*$ with $|X_0^* \cup X_1^*| \in \mathsf{poly}$ (where $\mathsf{poly}(\lambda)$ is fixed polynomial) and hence we can apply their theorem.

In order to argue with the security property of differing-inputs obfuscation, we need to present a differing-inputs circuit sampler Sam generating circuits $(C_3[\mathsf{k}, P_0, P_1], C_4[\mathsf{k}])$. For this we consider an algorithm Sam that runs the same steps as game Game_4 up-to line 7 and which sets $\mathsf{aux} \leftarrow L$ and outputs $(C_3[\mathsf{k}, P_0, P_1], C_4[\mathsf{k}], \mathsf{aux})$.

Claim 11.2. *If* AIPO *is a secure composable AIPO*$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$ *obfuscator (see Definition 6.6), then* Sam *is a differing-inputs circuit sampler which outputs circuits that differ on at most* q *many points.*

Before proving Claim 11.2, we show how to use it to prove that the difference between $\mathsf{Game}_4(\lambda)$ and $\mathsf{Game}_5(\lambda)$ is small. Theorem 5.13 by Boyle et al. [BCP14] says that, if a family is differing-inputs and only differs on at most polynomially many points, then their indistinguishability obfuscations are indistinguishable. Claim 11.2 establishes that Sam is a differing-inputs sampler, and we already observed that circuits $C_3[\mathsf{k}, P_0, P_1]$ and $C_4[\mathsf{k}]$ only differ on polynomially many points. Hence, Theorem 5.13 allows us to do an analysis similar to the one from the first game hop and we get that

$$\left|\Pr\left[\mathsf{Game}_4^{\mathsf{S},\mathsf{D}}(\lambda)\right] - \Pr\left[\mathsf{Game}_5^{\mathsf{S},\mathsf{D}}(\lambda)\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}_{\mathsf{iO}}}(\lambda) + [\mathrm{BCP14}] \leq \mathsf{negl}(\lambda)\,.$$

We now proceed to proving Claim 11.2. Assume there exists an adversary (i.e., an extractor) Ext against the differing-inputs sample Sam. Then, intuitively, if Ext succeeds to find some target value τ this should help in breaking the obfuscation scheme AIPO since there must be one obfuscation $\overline{p} \in P_0 \cup P_1$ such that $\overline{p}(\tau) = 1$. Let us now make this intuition formal.

We construct adversary $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ where $\mathsf{Sam}_{\mathsf{PO}}$ describes a statistically unpredictable auxiliary input point sampler. On input the security parameter, sampler $\mathsf{Sam}_{\mathsf{PO}}$ runs the steps of Game_4 up-to line 4. It constructs a sequence of points X which first contains all points from X_0^* and then all the points from X_1^*. It sets $j \leftarrow |X_0|$, that is, j is set to the number of points that were answered with 0. Sampler $\mathsf{Sam}_{\mathsf{PO}}$ chooses an index $\ell \leftarrow\!\!\$\, [|X|]$ at random, samples a uniformly random value $r \leftarrow\!\!\$\, \{0,1\}^{\mathsf{H.il}(\lambda)}$ and sets $b := \langle r, X[\ell]\rangle$ (i.e., b is set to the inner product of values r and $X[\ell]$, that is, $b := \oplus_{i=1}^{\mathsf{H.il}(\lambda)} r[i] \cdot X[\ell][i]$). It then sets $\mathsf{aux} \leftarrow (\ell, j, r, b, L)$ and outputs (X, aux).

Distinguisher D_{PO} gets as input the security parameter, auxiliary input $(\ell, j, r, b, L) \leftarrow \mathsf{aux}$ and a list of obfuscations $\overline{\mathbf{p}}$ which either contains point obfuscations for points in X or for randomly generated points. Distinguisher D_{PO} then constructs sets P_0 and P_1 as

$P_0, P_1 \leftarrow \{\}$
for $i = 1, \ldots, |\overline{\mathbf{p}}|$ **do**
  if $i \leq j$ **then**
    $P_0 \leftarrow P_0 \cup \{\overline{\mathbf{p}}[i]\}$
  else $P_1 \leftarrow P_1 \cup \{\overline{\mathbf{p}}[i]\}$

Distinguisher $\mathsf{D_{PO}}$ then samples a random key $\mathsf{k} \leftarrow\!\!\$\ \mathsf{F.KGen}(\lambda)$ and constructs circuits $C_3[\mathsf{k}, P_0, P_1]$ and $C_4[\mathsf{k}]$. It then calls Ext on input $(C_3[\mathsf{k}, P_0, P_1], C_4[\mathsf{k}], L)$ to receive a value τ. If Ext outputs $\tau = \bot$, then $\mathsf{D_{PO}}$ returns 1 with probability $\frac{1}{2\mathsf{q}}$ (and 0 otherwise). If extractor Ext succeeds and τ is such that $C_3[\mathsf{k}, P_0, P_1](\tau) \neq C_4[\mathsf{k}](\tau)$ but $\overline{\mathbf{p}}[\ell](\tau) \neq 1$ (i.e., Ext does not succeed for the ℓ-th point obfuscation) then $\mathsf{D_{PO}}$ always returns 0. Finally, if $\overline{\mathbf{p}}[\ell](\tau) = 1$ then $\mathsf{D_{PO}}$ outputs 1 if $\langle r, \tau \rangle$ equals b and 0 otherwise.

If $\overline{\mathbf{p}}$ is an obfuscation of the points in X and Ext outputs τ such that $\overline{\mathbf{p}}[\ell](\tau) = 1$ then $\mathsf{D_{PO}}$ will output 1 with probability 1. If, on the other hand, $\overline{\mathbf{p}}$ is a sequence of obfuscations of random points and Ext outputs τ such that $\overline{\mathbf{p}}[\ell](\tau) = 1$ then $\mathsf{D_{PO}}$ will only output 1 with probability $\frac{1}{2}$ (since $\Pr[\langle u, r \rangle = b] = \frac{1}{2}$ for a random point u).

Let us make the probability analysis formal. Let $d = 0$ describe the event that the values in set X get obfuscated, and $d = 1$ describe the event that instead uniformly random values get obfuscated. Let ϵ be the probability that Ext returns a value $\tau \neq \bot$ in the differing-inputs game, that is, $\epsilon := \Pr[d = 0 \mid \bot \neq \mathsf{Ext}]$. Note, that for readability we do not specify the input of adversaries Ext and $\mathsf{D_{PO}}$ in the following treatment. We now consider the distinguishing probability of distinguisher $\mathsf{D_{PO}}$

$$\Pr[\mathsf{D_{PO}} = 1 \mid d = 0] - \Pr[\mathsf{D_{PO}} = 1 \mid d = 1]$$

which can be rewritten as

$$\begin{aligned} &= \Pr[\mathsf{D_{PO}} = 1 \mid d = 0, \mathsf{Ext} \neq \bot] \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 0] + \\ &\quad \Pr[\mathsf{D_{PO}} = 1 \mid d = 0, \mathsf{Ext} = \bot] \cdot \Pr[\mathsf{Ext} = \bot \mid d = 0] - \\ &\quad \Pr[\mathsf{D_{PO}} = 1 \mid d = 1] \end{aligned} \tag{11.2}$$

If extractor Ext succeeds to extract a value τ on which the circuits differ then for one of the q point obfuscations in $\overline{\mathbf{p}}$ we have that $\overline{p}(\tau) = 1$ (i.e., for some $j \in [\mathsf{q}]$ we have that $\overline{\mathbf{p}}[j](\tau) = 1$). As Ext has no information which was the ℓ chosen by $\mathsf{Sam_{PO}}$ we have that with probability $\frac{1}{\mathsf{q}}$ that $\overline{\mathbf{p}}[\ell](\tau) = 1$. Consequently line (11.2) reduces to $\frac{1}{\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 0]$ since in case that $d = 0$ then $\langle r, \tau \rangle$ will always equal b by construction. If, on the other hand, extractor Ext does not succeed $\mathsf{D_{PO}}$ outputs 1 with probability $\frac{1}{2\mathsf{q}}$. We, thus, can rewrite the above as:

$$\begin{aligned} &= \frac{1}{\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 0] + \frac{1}{2\mathsf{q}} \cdot \Pr[\mathsf{Ext} = \bot \mid d = 0] - \Pr[\mathsf{D_{PO}} = 1 \mid d = 1] \\ &= \frac{1}{\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 0] + \frac{1}{2\mathsf{q}} \cdot \Big(1 - \Pr[\mathsf{Ext} \neq \bot \mid d = 0]\Big) - \Pr[\mathsf{D_{PO}} = 1 \mid d = 1] \\ &= \frac{1}{2\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 0] + \frac{1}{2\mathsf{q}} - \Pr[\mathsf{D_{PO}} = 1 \mid d = 1] \end{aligned}$$

Setting $\epsilon := \Pr[d = 0 \mid \bot \neq \mathsf{Ext}]$ yields

$$
\begin{aligned}
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1] \\
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1, \mathsf{Ext} \neq \bot] \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 1] + \\
&\qquad\qquad \Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1, \mathsf{Ext} = \bot] \cdot \Pr[\mathsf{Ext} = \bot \mid d = 1]
\end{aligned}
$$

If $d = 1$ and the extractor succeeds then D_{PO} will only output 1 if the extractor succeeds for the ℓ-th point obfuscation, that is, if $\overline{\mathbf{p}}[\ell](\tau) = 1$ and furthermore if $\langle \tau, r \rangle = b$. Note that the latter condition in case $d = 0$ is always true. In case $d = 1$ we, however, have that τ is uniformly random and r is outside the view of Ext and, thus, $\Pr[\langle \tau, r \rangle = b] = \frac{1}{2}$. Combined, this yields that $\Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1, \mathsf{Ext} \neq \bot]$ is equal to $\frac{1}{2\mathsf{q}}$.

$$
\begin{aligned}
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \frac{1}{2\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 1] + \\
&\qquad\qquad \Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1, \mathsf{Ext} = \bot] \cdot \Pr[\mathsf{Ext} = \bot \mid d = 1]
\end{aligned}
$$

Again, if extractor Ext does not succeed, distinguisher D_{PO} outputs 1 with probability $\frac{1}{2\mathsf{q}}$ which allows us to rewrite the above as

$$
\begin{aligned}
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \frac{1}{2\mathsf{q}} \cdot \Pr[\mathsf{Ext} \neq \bot \mid d = 1] + \frac{1}{2\mathsf{q}} \cdot \Pr[\mathsf{Ext} = \bot \mid d = 1] \\
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \frac{1}{2\mathsf{q}} \cdot \Big(\Pr[\mathsf{Ext} \neq \bot \mid d = 1] + \Pr[\mathsf{Ext} = \bot \mid d = 1] \Big) \\
&= \frac{1}{2\mathsf{q}}\epsilon + \frac{1}{2\mathsf{q}} - \frac{1}{2\mathsf{q}} \cdot 1 \\
&= \frac{1}{2\mathsf{q}}\epsilon
\end{aligned}
$$

To wrap up, we have shown that the distinguishing probability for D_{PO} is

$$\Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 0] - \Pr[\mathsf{D}_{\mathsf{PO}} = 1 \mid d = 1] = \frac{1}{2\mathsf{q}}\epsilon,$$

where ϵ is the success probability of extractor Ext in the differing-inputs game.

To finish the proof of Claim 11.2 we need to argue that $\mathsf{Sam}_{\mathsf{PO}}$ implements a statistically unpredictable auxiliary input point sampler. By assumption, source S is statistically unpredictable (i.e., $\mathsf{S} \in \mathcal{S}^{\mathrm{sup}}$) and hence leakage L information-theoretically does not contain any value in X. Thus, to see that $\mathsf{Sam}_{\mathsf{PO}}$ defines an unpredictable point sampler, we need to argue that X remains statistically unpredictable even if a predictor is additionally given (ℓ, j, r, b). But a single bit b and two indexes $\ell, j \in [\mathsf{q}]$ can be guessed with probability $\frac{1}{2\mathsf{q}^2}$ and r is a uniformly random value. Hence, $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ breaks the security of the AIPO obfuscation, which concludes the proof of Claim 11.2 and the proof of Theorem 11.1. □

$\underline{\text{mUCE}^{\mathsf{S},\mathsf{D}}_{\mathsf{H}}(\lambda)}$

$(1^n, \text{state}) \leftarrow\$\ \mathsf{S}(1^\lambda, \varepsilon)$
$b \leftarrow\$\ \{0,1\}$
for $i = 1, \ldots, n,$ **do**
  $\mathsf{hk}[i] \leftarrow\$\ \mathsf{H.KGen}(1^\lambda)$
$L \leftarrow\$\ \mathsf{S}^{\text{HASH}}(1^n, \text{state})$
$b' \leftarrow\$\ \mathsf{D}(1^\lambda, \mathsf{hk}, L)$
return $(b = b')$

$\underline{\text{HASH}(x, i)}$

if $T[x,i] = \bot$ **then**
  if $b = 1$ **then**
    $T[x,i] \leftarrow \mathsf{H.Eval}(\mathsf{hk}[i], x)$
  else $T[x,i] \leftarrow\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x,i]$

$\underline{\mathsf{mPred}^{\mathsf{P}}_{\mathsf{S}}(\lambda)}$

$(1^n, \text{state}) \leftarrow\$\ \mathsf{S}(1^\lambda, \varepsilon)$
$\text{done} \leftarrow \textbf{false};\ Q \leftarrow \{\}$
$L \leftarrow\$\ \mathsf{S}^{\text{HASH}}(1^n, \text{state})$
$\text{done} \leftarrow \textbf{true}$
$Q' \leftarrow\$\ \mathsf{P}^{\text{HASH}}(1^\lambda, 1^n, L)$
return $(Q \cap Q' \neq \{\})$

$\underline{\text{HASH}(x, i)}$

if $\text{done} = \textbf{false}$ **then**
  $Q \leftarrow Q \cup \{x\}$
if $T[x,i] = \bot$ **then**
  $T[x,i] \leftarrow\$\ \{0,1\}^{\mathsf{H.ol}(\lambda)}$
return $T[x,i]$

Figure 11.2: The multi-key UCE game (mUCE) on the left and the corresponding unpredictability game on the right. Similarly to the plain UCE game we consider two variants of unpredictability: computationally unpredictable sources where predictor P is restricted to be PPT and statistically unpredictable sources where predictor P is unbounded.

11.3 Multi-Key UCEs

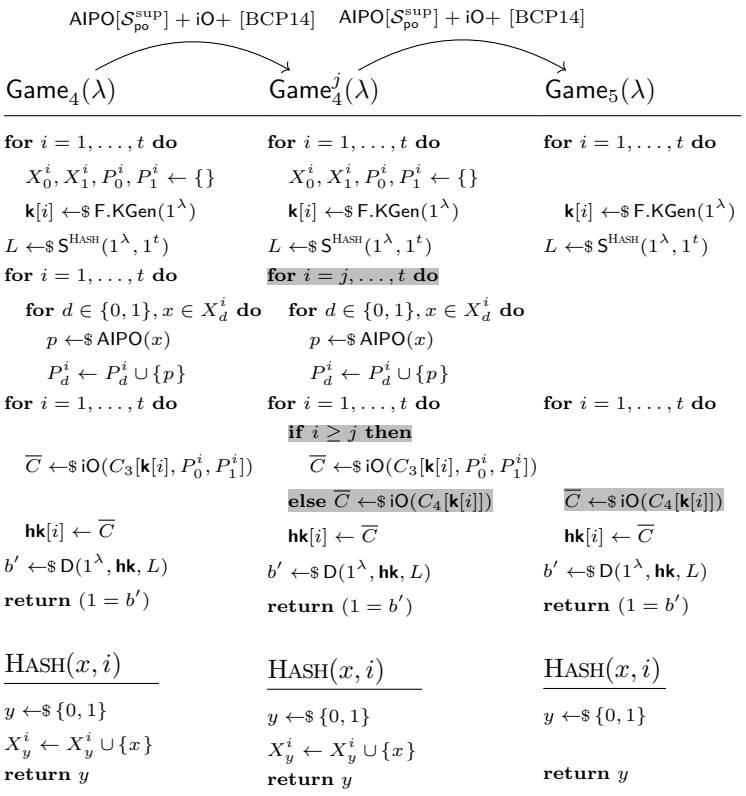
In this section we consider the changes needed to show that the previous construction also achieves multi-key UCE security. Recall that in the multi-key version called mUCE the source S can decide with how many keys it wants to work and oracle HASH takes as input an index specifying the key. Furthermore, the unpredictability game is extended such that a predictor succeeds if it guesses any of the queries for any key. We discuss multi-key UCEs in detail in Section 9.2.1 and for convenience repeat the game definition and corresponding unpredictability definition in Figure 11.2.

We note that the relationship between UCEs and multi-key UCEs is still an open research question. While it is easy to see that $\text{mUCE}[\mathcal{S}^{\text{cup}}]$ (and its statistical variant) imply the corresponding single key variants the other direction is not known to hold. On the other hand, we are not aware of any separating example and believe that such an example would be interesting. Our construction here as well as the UCE constructions in idealized models discussed in Section 9.3 both also achieve multi-key security.

Theorem 11.3. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable AIPO$[\mathcal{S}^{\text{sup}}_{\mathsf{po}}]$) exist then Construction 11.2 is* $\text{mUCE}_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ *secure.*

Note that we do not need to take care of the number of keys in the construction since for each key we allow at most q-many queries and each key corresponds to a new circuit.

The proof is analogous to the proof for the single-keyed version. Instead of a single key, now t keys need to be handled and constructed. The number t of keys in the system can, for example, be upper bounded by the runtime of the source S allowing us to keep it fixed (the source may use only a fraction of the keys, if it so chooses). In the game steps, the sets X_0^*, X_1^*, P_0, P_1 are multiplied by the number of keys in the system. The first game hop works analogously while the second game hop can be done one key at a time. The third game hop again is analogously as in the single-key version and the final game hop can again be done one key at a time. We have depicted the final game hop in Figure 11.3. Game Game_4^j is an intermediate game with $\mathsf{Game}_4^1 = \mathsf{Game}_4$ and $\mathsf{Game}_4^{t+1} = \mathsf{Game}_5$. In Game_4^j the first $j-1$ keys are already transformed to be generated as an obfuscation of the correct circuit $C_4[\mathsf{k}]$. The difference between Game_4^j and Game_4^{j+1} is hence that key $\mathsf{hk}[j]$ is either generated



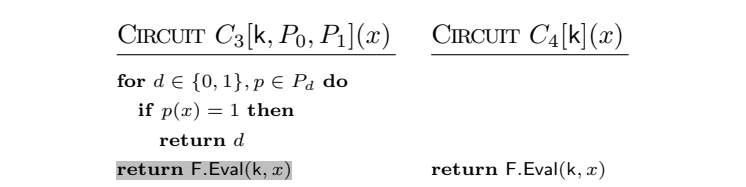


Figure 11.3: Relevant game steps for proof of Theorem 11.3.

as

$$\mathsf{iO}(C_3[\mathbf{k}[i], P^i_0, P^i_1])$$

or as

$$\mathsf{iO}(C_4[\mathbf{k}[i]])$$

which is analogous to the single-key game. Hence we have that

$$\left|\Pr\left[\mathsf{Game}^j_4(\lambda) = 1\right] - \Pr\left[\mathsf{Game}^{j+1}_4(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{D}_{\mathsf{iO}}}(\lambda) + [\text{BCP14}] \leq \mathsf{negl}(\lambda)\,.$$

11.4 UCEs for Strongly Unpredictable Sources

In the previous sections we constructed a mUCE$_1[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$ secure hash function. For this, we assumed the existence of composable AIPO$[\mathcal{S}^{\text{sup}}_{\text{po}}]$ point obfuscator, that is, a point obfuscator that is

secure in the presence of *statistically* hard-to-invert auxiliary information and which is, furthermore, composable. To adapt our construction to also be a *computational* UCE the straight forward change is to assume the existence of a composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] point obfuscator. Indeed, this would directly yield a mUCE$_1$[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] function and when running it in counter mode we would obtain also a mUCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\mathsf{p}\text{-query}}$] function with long output. (For this remember that we can trade output length for oracle queries as discussed in Section 9.7.1.)

Having shown in Section 9.5 that a UCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{1\text{-query}}$] secure function cannot exist if indistinguishability obfuscation exists the above cannot work. Indeed, in Chapter 7 we have shown that composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] do not exist. So what about non-composable point obfuscators?

If we exchange composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] for non-composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$], then the last game hop in the proof of Theorem 11.1 does not go through any longer; in more detail, we are no longer able to show Claim 11.2. On the other hand, if we restrict the number of queries to be constant, then the proof again works and we can show that the construction is UCE$_1$[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{1\text{-query}}$] secure. This result is, however, much weaker than what we bargained for as we now have a UCE function with only a single bit output and which admits a single (or constantly many) queries. While this form of UCE security still has interesting applications, for example, the function is a universal hardcore predicate, what we ideally would like is to not restrict the output size of the function. Of course, simply removing the restriction cannot work because then we again run into our impossibility result from Section 9.5. Thus, the idea is to trade the restriction of having short (i.e., constant) outputs for only obtaining *strong* unpredictability (instead of plain unpredictability).

Aiming for strong unpredictability does not only allow us to increase the output length of the UCE function in the computational setting but also in the statistical setting. For this note that before, in the statistical setting, we went from mUCE$_1$[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] to mUCE[$\mathcal{S}^{\text{cup}} \cap \mathcal{S}^{\mathsf{p}\text{-query}}$] by running the construction in counter mode; consequently we loose the gain in output length in terms of oracle queries (see Section 9.7.1). The alternative is that we can directly prove Construction 11.1 secure, that is, the construction with potentially large output length, if we restrict the unpredictability condition to be *strong unpredictable*.

11.4.1 Construction 11.1 is UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}$] Secure

Theorem 11.4. *If indistinguishability obfuscation exists and if AIPO for computationally unpredictable distributions (non-composable AIPO*[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]*) exist then Construction 11.1 is* UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}$] *secure.*

The proof is similar to our previous proof adapted to only keep track of a single query. Thus, instead of remembering the answers 0 and 1 by grouping the queries into two sets we now simply store the single answer in variable y^*. The proofs then differ in the final game hop where we exploit the *strong* unpredictability.

Proof. We consider a sequence of five games with pseudocode given in Figure 11.4:

$\mathsf{Game}_1(\lambda)$**:** The first game is the original UCE$_1$[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}$]-game with hidden bit b set to 1. Here, the hash key hk is an obfuscation of the circuit $C_1[\mathsf{k}](x) := \mathsf{F.Eval}(\mathsf{k}, x)$ where k is a key for the puncturable PRF.

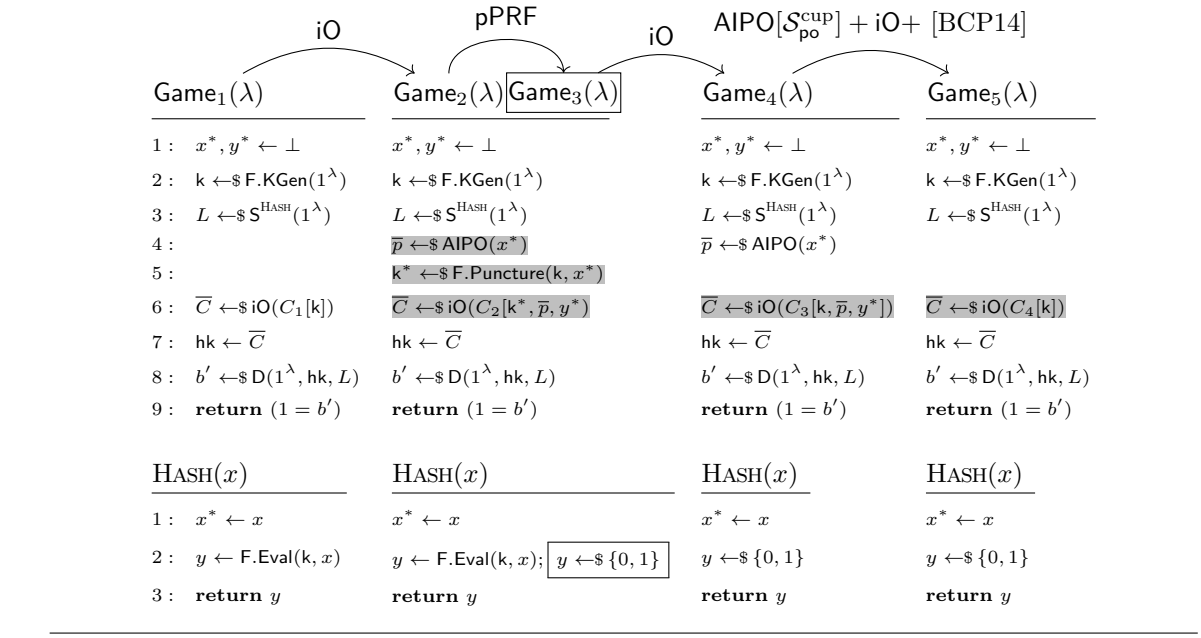


Figure 11.4: A pseudocode description of the game steps in Theorem 11.4. The boxed Game$_3$ is to be understood with the boxed statement included. Furthermore the highlighted lines are emphasize the difference to the previous game.

iO

Game$_2(\lambda)$**:** Let x^* denote the single query by source S to its HASH oracle and let y^* denote the answer. Game$_2$ is identical to Game$_1$ with the exception that circuit $C_2[\mathsf{k}^*, \overline{p}, y^*]$ is obfuscated instead of circuit $C_1[\mathsf{k}]$. Here k^* is the punctured key which is punctured on the single query point x^*. Note that circuit $C_2[\mathsf{k}^*, \overline{p}, y^*]$ is, by construction, functionally equivalent to $C_1[\mathsf{k}]$. On input x it first checks whether the input is equivalent to x^* and if so outputs y^*. Otherwise, it uses the punctured key to output $\mathsf{F.Eval}(\mathsf{k}^*, x)$.

pPRF

Game$_3(\lambda)$**:** The game is equivalent to Game$_2$ except that oracle HASH now samples y^* uniformly at random instead of invoking $\mathsf{F.Eval}(\mathsf{k}, \cdot)$.

iO

Game$_4(\lambda)$**:** The game is equivalent to the previous game except that we now use an obfuscation of circuit $C_3[\mathsf{k}, \overline{p}, y^*]$. The circuit is functionally equivalent to $C_2[\mathsf{k}^*, \overline{p}, y^*]$ as the puncturable PRF is only called on values that were not punctured out.

AIPO[$\mathcal{S}^{\text{cup}}_{\text{po}}$] + iO + [BCP14]

Game$_5(\lambda)$**:** The game is equivalent to the previous game except that now an obfuscation of circuit $C_4[\mathsf{k}]$ is used as hash key. Circuit $C_4[\mathsf{k}]$ is our original circuit again, that is, $C_4[\mathsf{k}](\cdot) := \mathsf{F.Eval}(\mathsf{k}, \cdot)$. Game$_5$ is our intended target. It is the UCE-security game for our construction in the random oracle world (that is, oracle HASH implements a random oracle).

The first three game hops are proved analogously to before. The difference is the last game hop where we now need to prove the following claim:

Claim 11.5. *If* AIPO *is a secure AIPO*$[\mathcal{S}^{\text{cup}}_{\mathsf{po}}]$ *obfuscator (see Definition 6.6), then* Sam *is a differing-inputs circuit sampler which outputs circuits that differ on at most a single point.*

That is, again we bound the difference between Game_4 and Game_5 by differing-inputs obfuscation and use the result by Boyle, Chung, and Pass ([BCP14]; see Section 5.6.1) to argue that this is down to indistinguishability obfuscation as the two circuits only differ by a single point. Also Claim 11.5 is similar to Claim 11.2 with an important distinction. Previously we constructed an AIPO adversary $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ that ordered the query points such that the first block corresponds to queries answered with 0 and all the remaining points correspond to queries that were answered with 1. Now we only have a single query point x^*, however, the output y^* can be of arbitrary length (instead of a single bit). Thus, we construct an AIPO adversary as follows: On input the security parameter, sampler $\mathsf{Sam}_{\mathsf{PO}}$ runs the steps of Game_4 up-to line 4 storing the single query in x^* and the single answer in y^*. It then samples a uniformly random value $r \leftarrow\!\!\$\, \{0,1\}^{\mathsf{H.il}(\lambda)}$ and sets $b := \langle r, x^* \rangle$. Finally, it sets $\mathsf{aux} \leftarrow (r, b, L, y^*)$ and outputs (x^*, aux).

Distinguisher D_{PO} gets as input the security parameter, auxiliary input $(r, b, L, y^*) \leftarrow \mathsf{aux}$ and a point obfuscation $\overline{p}$ which is either a point obfuscation for point in x^* or for a uniformly random point u. Distinguisher D_{PO} samples a random key $\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(\lambda)$ and constructs circuits $C_3[\mathsf{k}, \overline{p}, y^*]$ and $C_4[\mathsf{k}]$.

Distinguisher D_{PO} then calls Ext on input $(C_3[\mathsf{k}, \overline{p}, y^*], C_4[\mathsf{k}], L)$ to receive a value τ. If Ext outputs $\tau = \bot$, then D_{PO} flips a bit and returns the result.[2] On the other hand, if extractor Ext succeeds and consequently $\overline{p}(\tau) = 1$ then D_{PO} outputs 1 if $\langle r, \tau \rangle$ equals b and 0 otherwise.

In case $\overline{p}$ is an obfuscation of x^* and Ext succeeds then D_{PO} will always output 1 with probability 1. If, on the other hand, $\overline{p}$ is an obfuscation of a random point and Ext outputs τ such that $\overline{p}(\tau) = 1$ then D_{PO} will only output 1 with probability $\frac{1}{2}$ (since $\Pr[\langle u, r \rangle = b] = \frac{1}{2}$ for a random point u).

The formal analysis is analogous to before. What remains to show is that $\mathsf{Sam}_{\mathsf{PO}}$ implements a computationally unpredictable point sampler, that is, $\mathsf{Sam}_{\mathsf{PO}} \in \mathcal{S}^{\text{cup}}_{\mathsf{po}}$. By assumption, the source S is strongly computationally unpredictable (i.e., $\mathsf{S} \in \mathcal{S}^{\text{s-cup}}$) and hence leakage L hides x^* even in the presence of y^*. Thus, to see that $\mathsf{Sam}_{\mathsf{PO}}$ defines an unpredictable auxiliary input point sampler, we need to argue that x^* remains unpredictable if additionally given b and r. But a single bit can be guessed with probability $\frac{1}{2}$ and r is a uniformly random string. Hence, $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ breaks the security of the AIPO$[\mathcal{S}^{\text{cup}}_{\mathsf{po}}]$ obfuscation, which concludes the proof of Claim 11.5. □

It is instructive to clearly understand where the strong unpredictability comes into play. In the previous proof, as the UCE function only output a single bit, it was sufficient for point obfuscation adversary $\mathsf{Sam}_{\mathsf{PO}}$ to leak the number of queries that were answered with 0 in order for D_{PO} to perfectly simulate the environment for extractor Ext. In case the output of the UCE function is large, we need to additionally let $\mathsf{Sam}_{\mathsf{PO}}$ leak the actual output y^* and in order to remain unpredictable we require the source to be strongly unpredictable.

Multi-Key UCE

On first inspection one might be tempted to think that we can adapt the above proof also to obtain a multi-key variant, that is, to obtain a mUCE$[\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}]$ secure function. This is, however,

[2] Note that this is analogous to before where the distinguisher returned 1 with probability $\frac{1}{2\mathsf{q}}$ by noting that now $\mathsf{q} = 1$.

not the case as to make the final game hop we again need a composable point obfuscator as we have now many circuits $C_3[\mathsf{k}, \overline{p}, y^*]$ each containing one point obfuscation.

11.4.2 Construction 11.1 is UCE$[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{q-query}}]$ Secure

In the previous section we needed to restrict sources to make only constantly many queries as composable AIPO$[\mathcal{S}^{\text{cup}}_{\mathsf{po}}]$ and indistinguishability obfuscation are mutually exclusive (see Chapter 7). When switching to the statistical setting and assuming the existence of a AIPO$[\mathcal{S}^{\text{sup}}_{\mathsf{po}}]$ we do not have this restriction, that is, composable obfuscators may exist (and we actually have candidate constructions; see Chapter 6). Thus, we can adapt the argument from the previous section to argue that Construction 11.1 is UCE$[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{q-query}}]$-secure. In this case the point obfuscation adversary $\mathsf{Sam}_{\mathsf{PO}}$ needs to leak all the query results $y^*_1, \ldots, y^*_{\mathsf{q}}$. Strong unpredictability on the other hand guarantees unpredictability if the predictor gets the set $\{y^*_1, \ldots, y^*_{\mathsf{q}}\}$ and not necessarily if the predictor gets the list $(y^*_1, \ldots, y^*_{\mathsf{q}})$. While this distinction is irrelevant in case all the answers are distinct it may make a difference if collisions occur. Thus, to be able to argue that collisions only occur with low probability we additionally need to require that the output length of the hash function is sufficiently long (i.e., super-logarithmic in the security parameter). Putting it all together yields the following theorem.

Theorem 11.6. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable AIPO*$[\mathcal{S}^{\text{sup}}_{\mathsf{po}}]$*) exist then Construction 11.1 is* UCE$[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{q-query}}]$ *secure if* $\mathsf{H.ol}(\lambda) \in \omega(\log \lambda)$.

Chapter 12

Point Obfuscation is Necessary for UCE Security

"Frustra fit per plura, quod potest fieri per pauciora."—"It is vain to do with more what can be done with less."

William of Occam

Summary. Our UCE constructions in the previous chapter are based on strong assumptions on the existence of point-function obfuscation schemes and in this chapter we ask whether these assumptions are necessary. We answer this question in the affirmative and show that UCEs imply point-function obfuscators. More specifically, mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] implies composable AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$] and UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}$] implies the existence of non-composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]. We also show how to construct multi-bit output point obfuscators from UCEs. The results in this chapter are based on a manuscript [BM15b].

Chapter content

12.1 Introduction

The UCE constructions presented in the previous chapter relied heavily on point obfuscation. Indeed, we argued that some form of two-stage security notion is necessary due the negative results of Wichs' [Wic13] which apply to some of the primitives that we can construct from UCEs. For example, UCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}$] implies q-query correlation-input secure hash functions and q-query deterministic public-key encryption; two primitives for which Wichs showed that these cannot be proven secure down to single-stage assumptions (see Section 4.2.2).

This representation is not truly accurate. On the one hand, not all our UCE constructions are directly affected by Wichs' result. For example, currently the only known application for UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{1\text{-query}}$] are universal hardcore functions which are defined via a single-stage definition and, thus, not subject to Wichs' techniques. On the other hand, there may be also other ways around Wichs' result, for example, non-black-box techniques. Here, by non black-box, we mean reductions that exploit being given the *code* of the adversary instead of treating the adversary simply as a black-box for which they can only observer input-output behavior. In a celebrated work, Barak presents a non black-box technique for the realm of zero-knowledge argument systems and shows how to

construct constant-round public-coin zero-knowledge arguments with negligible soundness error and with simulators running in strict polynomial time for any language in NP [Bar01]. Furthermore, his protocol remains zero-knowledge under concurrent composition. Simultaneously achieving these properties is known to be impossible with black-box simulation techniques [CKPR01, BL02].

While we do not know of any non black-box techniques that may help in the construction of UCE functions, we want to ask whether our assumptions on the existence of strong point obfuscators are reasonable. In this chapter we answer this question in the affirmative, that is, we will show that strong point obfuscation is indeed necessary for the construction of most UCE functions and, in particular, necessary for the variants that we construct. Let us note that there is a small loop-hole. We will only show that point obfuscation is necessary when constructing *injective* UCE functions.

Point Obfuscation from UCEs

UCEs are designed to take the role of random oracles. In order to construct point obfuscation schemes from UCEs it is, thus, natural to start with a scheme in the random oracle model. The first such scheme was presented by Lynn, Prabhakaran, and Sahai ([LPS04]; LPS). LPS consider multi-bit output point obfuscation for point functions $p_{x,m}$ while for our results on UCEs we only needed obfuscation schemes for plain point functions, i.e., p_x defined as

$$p_x(x') = \begin{cases} 1 & \text{if } x = x' \\ 0 & \text{otherwise} \end{cases}$$

Let us recall the LPS obfuscation scheme (for a detailed description we refer to Chapters 6 and 7): To obfuscate a point (x, m) one computes

$\mathsf{PO}(x, m)$

$\mathsf{hk} \leftarrow\!\$\, \mathsf{H.KGen}(1^\lambda)$

$\overline{x} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, 0\|x)$

$\overline{m} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, 1\|x) \oplus m$

return $(\mathsf{hk}, \overline{x}, \overline{m})$

Given $(\mathsf{hk}, \overline{x}, \overline{m})$ and knowing x one can recover m by recomputing $\mathsf{H.Eval}(\mathsf{hk}, 1\|x)$ and xoring this to $\overline{m}$. Value $\overline{x}$ can be used to test whether, indeed, x is correct. LPS show that the scheme in the random oracle model is VBB-secure and we show that this even holds in the presence of auxiliary information (Theorem 7.1 on page 138). Note that for correctness we explicitly require the hash function to be injective (and, thus, have long outputs).

Bellare, Hoang, and Keelveedhi (BHK; [BHK13:2]) show that one can instantiate the random oracle by a UCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{2-query}}]$ (or, if one requires composability, by the multi-key variant mUCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{2-query}}]$) to achieve a form of point obfuscation in the standard model without auxiliary input security.

As mentioned, we only use plain point obfuscators in our constructions and not multi-bit output point obfuscator. The LPS scheme can, however, easily be simplified by just dropping the message part. That is, to obfuscate a point x we compute:

$\underline{\mathsf{PO}(x)}$

$\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.KGen}(1^\lambda)$

$\overline{x} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$

return $(\mathsf{hk}, \overline{x})$

In the following we will show that the hash function can be implemented by a UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] secure UCE function to obtain a (non-composable) AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]. Similarly, if implemented by a mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}$] we obtain a composable AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$] obfuscator. We note that these results can be strengthened to multi-bit output obfuscators. That is, mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}$] can be shown to be sufficient to obtain a MB-AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$]. On the other hand, UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] is not quite sufficient to obtain the computational MB-AIPO counterpart, that is, MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$]. For this note also, that we have shown that MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] and indistinguishability obfuscation are mutually exclusive. The reason that UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$] is not sufficient for full MB-AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] security is that a strongly unpredictable source cannot necessarily simulate the environment for the MB-AIPO adversary as it needs to be unpredictable even in the presence of oracle answers. In case of the LPS scheme, being given the two oracle answers allows to retrieve point message m and, thus, break the MB-AIPO thereby potentially providing sufficient means to inverting the leakage of an MB-AIPO adversary. In more detail, consider an MB-AIPO adversary that picks as point address x and message m as two random values and as auxiliary information leaks value $\mathsf{aux} \leftarrow x \oplus m$. Being able to recover m, thus, leads to x which in case of the LPS scheme leads to an oracle query. In turn, we can weaken the MB-AIPO definition and, similarly to strong unpredictability for UCEs, consider *strongly unpredictable auxiliary-input multi-bit output point samplers* where we require that point x remains hidden even in the presence of m. We introduce such strongly unpredictable point samplers in Section 12.3.

12.2 Point Obfuscation is Necessary for UCE Security

We start by showing that an injective mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{1-query}}$] secure function also suffices to construct a composable AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$], that is, AIPO secure in the presence of statistically hard-to-invert auxiliary input. We note that we can actually give the stronger statement, that already mUCE[$\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{1-query}}$], that is, strong statistical unpredictability is sufficient to imply AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$]. For this we will consider the simple construction of a plain point obfuscator relative to a hash function H which obfuscates a point x as:

$\underline{\mathsf{PO}(x)}$

$\mathsf{hk} \leftarrow\!\!\$\ \mathsf{H.KGen}(1^\lambda)$

$\overline{x} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$

$\overline{p} \leftarrow (\mathsf{hk}, \overline{x})$

return $\overline{p}$

Theorem 12.1. *Let* H *be an injective hash function family. If* H *is* mUCE[$\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{1-query}}$]*-secure then the above construction yields a composable AIPO*[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$] *obfuscator. If* H *is* UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]*-secure then the above construction yields a (non-composable) AIPO*[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] *obfuscator.*

$\mathsf{Game}_1^{\mathsf{Sam},\mathsf{D}}(\lambda)$	$\xrightarrow{\text{UCE}}$	$\mathsf{Game}_2^{\mathsf{Sam},\mathsf{D}}(\lambda)$	$\xrightarrow{\text{UCE}}$	$\mathsf{Game}_3^{\mathsf{Sam},\mathsf{D}}(\lambda)$
1 : $(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)$		$(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)$		$(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda)$
2 : **for** $i = 1, \ldots, \lvert\mathbf{x}\rvert$ **do**		**for** $i = 1, \ldots, \lvert\mathbf{x}\rvert$ **do**		**for** $i = 1, \ldots, \lvert\mathbf{x}\rvert$ **do**
3 : $\quad \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$		$\quad \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$		$\quad \mathsf{hk} \leftarrow\!\!\$\, \mathsf{H.KGen}(1^\lambda)$
4 :				$\quad x \leftarrow\!\!\$\, \{0,1\}^{\lvert\mathbf{x}[i]\rvert}$
5 : $\quad \overline{x} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, \mathbf{x}[i])$		$\quad \overline{x} \leftarrow\!\!\$\, \{0,1\}^{\lvert\mathsf{H.Eval}(\mathsf{hk},\mathbf{x}[i])\rvert}$		$\quad \overline{x} \leftarrow \mathsf{H.Eval}(\mathsf{hk}, x)$
6 :				
7 :				
8 : $\quad \overline{\mathbf{p}}[i] \leftarrow (\mathsf{hk}, \overline{x})$		$\quad \overline{\mathbf{p}}[i] \leftarrow (\mathsf{hk}, \overline{x})$		$\quad \overline{\mathbf{p}}[i] \leftarrow (\mathsf{hk}, \overline{x})$
9 : $b' \leftarrow\!\!\$\, \mathsf{D}(1^\lambda, \overline{\mathbf{p}}, \mathsf{aux})$		$b' \leftarrow\!\!\$\, \mathsf{D}(1^\lambda, \overline{\mathbf{p}}, \mathsf{aux})$		$b' \leftarrow\!\!\$\, \mathsf{D}(1^\lambda, \overline{\mathbf{p}}, \mathsf{aux})$
10 : **return** $1 = b'$		**return** $1 = b'$		**return** $1 = b'$

Figure 12.1: Game hops for proof of Lemma 12.1.

We note that the injectivity requirement of hash function H is not a requirement necessary for the security proof but for the correctness of the resulting point obfuscator.

Proof. We prove the statement for the statistical case. The computational case follows analogously. For this we present a reduction from an adversary $(\mathsf{Sam}, \mathsf{D})$ against the AIPO scheme to an adversary against a $\text{mUCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{1-query}}]$ secure function.

Suppose there exists an adversary $(\mathsf{Sam}, \mathsf{D})$ against the above obfuscation scheme. We will prove the claim via two game hops visualized in Figure 12.1. The first game Game_1 is the original AIPO game with hidden bit b set to 0, that is, the point functions are generated for the points output by Sam. From there we gradually move to Game_3 which is the AIPO game with hidden bit b set to 1. In the following we first describe the games and then show that the steps reduce to $\text{mUCE}[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{1-query}}]$ security:

$\mathsf{Game}_1(\lambda)$**:** The original AIPO game with hidden bit b set to 0, that is, the point functions are generated for the points output by Sam.

UCE

$\mathsf{Game}_2(\lambda)$**:** The game is as before but in line 5 values $\overline{x}$ is generated as a uniformly random bit string of the same length $\lvert\mathsf{H.Eval}(\mathsf{hk}, \mathbf{x}[i])\rvert$ (for the i-th value).

UCE

$\mathsf{Game}_3(\lambda)$**:** The original AIPO game with hidden bit b set to 1, that is, the point functions are generated for uniformly random points.

We can write the advantage of adversary Sam, D as

$$\begin{aligned}
\mathsf{Adv}^{\text{po}}_{\mathsf{Sam},\mathsf{D},\mathsf{H}}(\lambda) &= \Pr\left[\mathsf{PO}_{\mathsf{H}}^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1 \;\middle|\; b = 0\right] + \Pr\left[\mathsf{PO}_{\mathsf{H}}^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1 \;\middle|\; b = 1\right] - 1 \\
&= \Pr\left[\mathsf{Game}_1^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_5^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] \\
&\leq \sum_{i=1}^{2} \left|\Pr\left[\mathsf{Game}_i^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{i+1}^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right]\right|
\end{aligned}$$

What remains to show is that the distance between any two games is negligible which we show next, reducing the first two steps to mUCE security; the last step follows due to the injectivity of the hash function.

$\mathsf{Game}_1(\lambda)$ to $\mathsf{Game}_2(\lambda)$. We construct an mUCE adversary (S, D). Without loss of generality we assume that Sam outputs t many points (where t is some polynomial in λ) and can hence consider a source that always works on t many keys. We consider (S, D) as

$\mathsf{S}^{\text{HASH}}(1^\lambda)$	$\mathsf{D}(1^\lambda, \mathbf{hk}, L)$
$(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$	$(\overline{\mathbf{x}}, \mathsf{aux}) \leftarrow L$
for $i = 1, \ldots, t$ **do**	**for** $i = 1, \ldots, t$ **do**
$\quad \overline{\mathbf{x}}[i] \leftarrow \text{HASH}(\mathbf{x}[i], i)$	$\quad \overline{\mathbf{p}}[i] \leftarrow (\mathbf{hk}[i], \overline{\mathbf{x}}[i])$
$L \leftarrow (\overline{\mathbf{x}}, \mathsf{aux})$	$b' \leftarrow\!\!\$\ \mathsf{D}(\overline{\mathbf{p}}, \mathsf{aux})$
return L	**return** b'

In case HASH implements the actual hash function, then (S, D) simulate Game_1 and in case HASH implements a random function they simulate Game_2. Thus, we have that

$$\left|\Pr\left[\mathsf{Game}_2^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda).$$

Furthermore, note that source S is strongly unpredictable. In fact, it is easier to note that source S is split and (plain) unpredictable where the latter follows from the unpredictability of Sam.

$\mathsf{Game}_2(\lambda)$ to $\mathsf{Game}_3(\lambda)$. We can bound the distance between games Game_2 and Game_3 again down to mUCE. For this we consider (S, D), where D is as before, but S now asks random queries.

$\mathsf{S}^{\text{HASH}}(1^\lambda)$	$\mathsf{D}(1^\lambda, \mathbf{hk}, L)$
$(\mathbf{x}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda)$	$(\overline{\mathbf{x}}, \mathsf{aux}) \leftarrow L$
for $i = 1, \ldots, t$ **do**	**for** $i = 1, \ldots, t$ **do**
$\quad x \leftarrow\!\!\$\ \{0,1\}^{\lvert\mathbf{x}[i]\rvert}$	$\quad \overline{\mathbf{p}}[i] \leftarrow (\mathbf{hk}[i], \overline{\mathbf{x}}[i])$
$\quad \overline{\mathbf{x}}[i] \leftarrow \text{HASH}(x, i)$	$b' \leftarrow\!\!\$\ \mathsf{D}(\overline{\mathbf{p}}, \mathsf{aux})$
$L \leftarrow (\overline{\mathbf{x}}, \mathsf{aux})$	**return** b'
return L	

Now, the view given by (S, D) when HASH is implemented as a random function is exactly as in Game_2 and in case it is implemented by the actual hash function HASH it is as in Game_3. It follows that

$$\left|\Pr\left[\mathsf{Game}_3^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_2^{\mathsf{Sam},\mathsf{D}}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}}(\lambda).$$

To complete the argument we note that the source is still split and, in fact, statistically unpredictable (independent of Sam). □

Remark. In the above application we used the multi-key version of UCEs mainly because we did not define obfuscators to simultaneously obfuscate multiple circuits or to come with some form of public parameters. We note that then we could simplify the above and reuse a single hash key for all obfuscations.

12.3 Multi-Bit Output Point Obfuscation from UCE

We end this chapter showing that UCEs can also implement the random oracle in the LPS scheme to achieve either composable MB-AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{mbpo}}$], that is, MB-AIPO secure with respect to statistically unpredictable multi-bit point samplers, or (non-composable) MB-AIPO[$\mathcal{S}^{\text{s-cup}}_{\mathsf{mbpo}}$]. The latter sampler restriction is called *strongly unpredictable* and is analogous to the restriction for UCEs. Let us capture this formally and note that the only difference to Definition 6.8 for sampler class $\mathcal{S}^{\text{cup}}_{\mathsf{mbpo}}$ is that the predictor additionally gets m as input:

Definition 12.1 (Strongly Unpredictable auxiliary-inputs multi-bit output point sampler)**.** *An algorithm* Sam *that on input the security parameter* 1^λ *outputs three strings* (x, m, aux) *is called a* strongly computationally unpredictable *(resp.* strongly statistically unpredictable*)* auxiliary-input multi-bit output point sampler *if no* PPT *algorithm (resp. unbounded algorithm) can predict* x *from* aux *and message* m*. That is, for every* PPT *(resp. unbounded) algorithm* P *and for all large enough* λ*:*

$$\Pr_{(x,m,\mathsf{aux}) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda)} \left[\mathsf{P}(1^\lambda, \mathsf{aux}, m) = x\right] \leq \mathsf{negl}(\lambda)$$

We let $\mathcal{S}^{\text{s-cup}}_{\mathsf{mbpo}}$ *denote all efficient computationally unpredictable auxiliary-input multi-bit output point samplers and denote by* $\mathcal{S}^{\text{s-sup}}_{\mathsf{mbpo}}$ *all efficient statistically unpredictable auxiliary-input multi-bit output point samplers.*

With that we can state our result on MB-AIPOs from UCEs.

Theorem 12.2. *Let* H *be an injective hash function family. If* H *is* mUCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{2-query}}$]*-secure then instantiating the LPS construction yields a composable MB-AIPO*[$\mathcal{S}^{\text{sup}}_{\mathsf{mbpo}}$] *obfuscator. If* H *is* UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{2-query}}$]*-secure*[1] *then the instantiated LPS construction yields a (non-composable) MB-AIPO*[$\mathcal{S}^{\text{s-cup}}_{\mathsf{mbpo}}$] *obfuscator.*

Proof. We again prove the statement only for the statistical case and note that the computational case follows analogously noting that in the computational case we consider strongly unpredictable UCEs and strongly unpredictable MB-AIPO.

Suppose there exists an adversary $(\mathsf{Sam}, \mathsf{D})$ against the above obfuscation scheme. We will prove the claim via three game hops visualized in Figure 12.2. The first game Game_1 is the original MB-AIPO game for the LPS construction and with hidden bit b set to 0, that is, the point functions are generated for the points as output by Sam. From there we gradually move to Game_4 which

[1] We note that we have formally shown only the existence of a UCE[$\mathcal{S}^{\text{s-cup}} \cap \mathcal{S}^{\text{1-query}}$]-secure function. To allow for two queries we would need an AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] that is t-composable for $t = \mathcal{O}(1)$. In case points are sampled independently it can be shown that any AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$] is also constantly composable. If the points are correlated we do not know if the implication holds. In [BM14a] we present a different construction of an (non-composable) MB-AIPO[$\mathcal{S}^{\text{s-cup}}_{\mathsf{mbpo}}$] from non composable AIPO[$\mathcal{S}^{\text{cup}}_{\mathsf{po}}$].

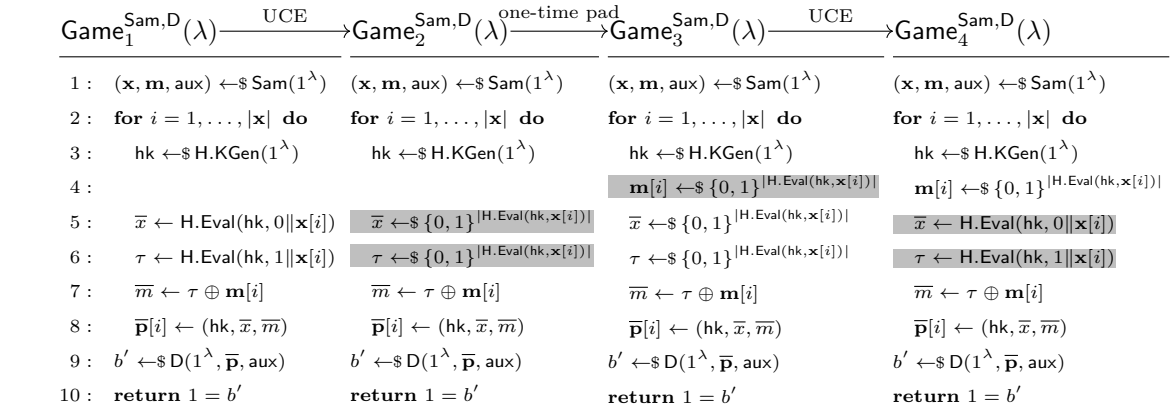


Figure 12.2: Game hops for proof of Lemma 12.2.

is the MB-AIPO game with hidden bit b set to 1 where point messages m are sampled uniformly at random. In the following we first describe the games and then show that the steps reduce to mUCE$[\mathcal{S}^{\text{s-sup}} \cap \mathcal{S}^{\text{1-query}}]$ security:

UCE

$\mathsf{Game}_1(\lambda)$**:** The original MB-AIPO game with hidden bit b set to 0, that is, the point functions are generated for the points output by Sam.

$\mathsf{Game}_2(\lambda)$**:** The game is as before but now instead of real hash evaluations random values are chosen. That is in lines 5 and 6 values $\overline{x}$ and τ are generated as a uniformly random bit string of the same length $|\mathsf{H.Eval}(\mathsf{hk}, \mathbf{x}[i])|$ (for the i-th value).

OTP

$\mathsf{Game}_3(\lambda)$**:** The game is as before but now message $\mathbf{m}[i]$ are chosen as uniformly random bit strings. For this in line 4 the entry in message vector $\mathbf{m}$ is overwritten.

UCE

$\mathsf{Game}_4(\lambda)$**:** The game is identical to the MB-AIPO game with hidden bit b set to 1, that is, the point functions are generated for uniformly random point messages. Note that the difference to the previous game is that values $\overline{x}$ and τ (lines 5 and 6) are again generated according to the hash function.

As usual, we next discuss the distinguishing probability of each step in turn.

$\mathsf{Game}_1(1^\lambda)$ to $\mathsf{Game}_2(1^\lambda)$. We construct a UCE adversary $(\mathsf{S}, \mathsf{D}')$ to bound the distinguishing probability between games Game_1 and Game_2. Source S runs the steps of game Game_1 until and including line 6 using its HASH oracle to generate $\overline{x}$ and τ in lines 5 and 6 and "encrypts" message $\mathbf{m}[i]$ as in line 7. That is, it does not set $\overline{p}$ as it does not know the hash key. Instead it stores values $\overline{x}$ and $\overline{m}$ in vectors $\overline{\mathbf{x}}$ and $\overline{\mathbf{m}}$. After processing all the messages output by Sam source S stops and outputs as leakage $L \leftarrow (\overline{\mathbf{x}}, \overline{\mathbf{m}}, \mathsf{aux})$.

Distinguisher D' gets as input hash keys $\mathbf{hk}$ as well as $(\overline{\mathbf{x}}, \overline{\mathbf{m}}, \mathsf{aux})$. It constructs vector $\overline{\mathbf{p}}$ as in line 8 for all entries in vectors $\overline{\mathbf{x}}$ and $\overline{\mathbf{m}}$. It then runs distinguisher D on input the security parameter vector $\overline{\mathbf{p}}$ and aux and outputs whatever it outputs.

First note that together S and D' perfectly simulate the environment for D in Game_1 if the HASH oracle implements the actual hash function and they perfectly simulate Game_2 otherwise. We, thus,

have that

$$\left|\Pr\left[\mathsf{Game}_2^{\mathsf{Sam},\mathsf{D}}(\lambda)\right] - \Pr\left[\mathsf{Game}_1^{\mathsf{Sam},\mathsf{D}}(\lambda)\right]\right| \leq \mathsf{Adv}^{\mathrm{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}'}(\lambda),$$

if we can show that source S is statistically unpredictable. This, however is the case as by assumption sampler Sam is statistically unpredictable and hence aux does information theoretically not contain any of the values in vector $\mathbf{x}$. Furthermore, values $\overline{x}$ and τ are uniformly random values (note that the unpredictability game is relative to a random oracle) and hence $\overline{\mathbf{m}}[i]$ is a one-time pad for message $\mathbf{m}[i]$.

$\mathsf{Game}_2(1^\lambda)$ to $\mathsf{Game}_3(1^\lambda)$. For the difference between games Game_2 and Game_3 we note that "encrypted" message $\overline{m}$ is a de-facto one-time pad of message $\mathbf{m}[i]$ and thus

$$\Pr\left[\mathsf{Game}_2^{\mathsf{Sam},\mathsf{D}}(\lambda)\right] = \Pr\left[\mathsf{Game}_3^{\mathsf{Sam},\mathsf{D}}(\lambda)\right].$$

$\mathsf{Game}_3(1^\lambda)$ to $\mathsf{Game}_4(1^\lambda)$. For the final game hop we note that the difference is identical to the difference between games Game_1 and Game_2 and, thus, an identical analysis can be done. It follows that

$$\left|\Pr\left[\mathsf{Game}_4^{\mathsf{Sam},\mathsf{D}}(\lambda)\right] - \Pr\left[\mathsf{Game}_3^{\mathsf{Sam},\mathsf{D}}(\lambda)\right]\right| \leq \mathsf{Adv}^{\mathrm{muce}}_{\mathsf{H},\mathsf{S},\mathsf{D}'}(\lambda)$$

which concludes the proof. □

Chapter 13

On the Necessity of Padding in Indistinguishability Obfuscation

> "It doesn't seem like that would make much sense to do that, to add useless padding and then apply your iO on top of that. [...] I mean it accomplishes something, certainly in terms of the proof; in the real world I don't know what that accomplishes."
>
> Craig Gentry, Simons Institute, 2015 Cryptography Boot Camp

Summary. In this chapter we initiate a study of padding in obfuscation-based techniques. As seen in the previous chapters, due to the use of padding in our construction we can only prove them q-query secure. We discuss whether padding is necessary when using indistinguishability obfuscation-based techniques and formulate a parameterizable assumption called *superfluous padding assumption* (SuPA) which intuitively states that padding under certain restrictions is not necessary. As we will see, for many reasonable restrictions it can be shown that SuPA cannot hold, hinting at padding being a necessary evil when working with obfuscation. On the positive side we present certain restricted forms of SuPA which are sufficient to lift the q-query bound from our UCE constructions and which we do not know not to hold. The results in this chapter are based on manuscript [BM15b]. The counter-examples to the general applicability of SuPA are due to Bitansky et al. [BCC+14] and Holmgren [Hol15].

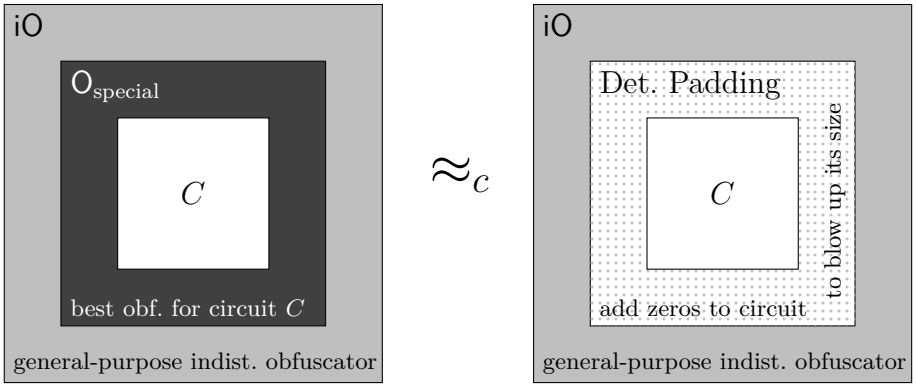
Chapter content

13.1 Introduction

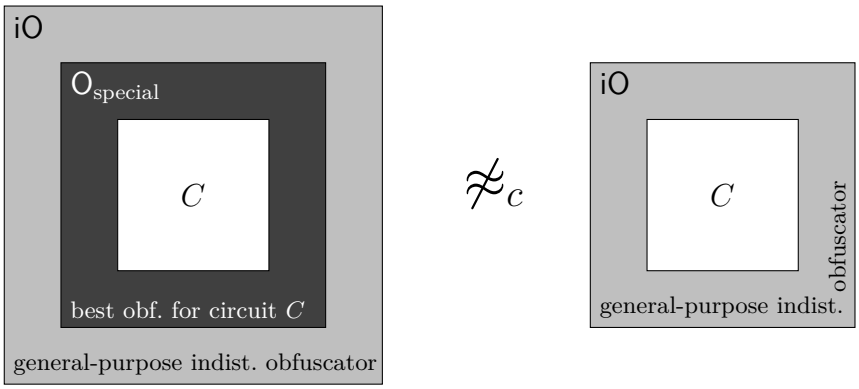
In the previous chapter we asked whether point-function obfuscation is necessary for our UCE constructions to work. In this chapter we turn to the padding that we need to perform in order for our proofs to go through. Remember that, especially for the q-bounded statistically unpredictable UCE (formally, UCE[$\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}$]), we need to first pad the puncturable PRF with NOP gates such that we obtain a much larger circuit which, however, still computes the very same function. Only after we have performed the padding operation can we obfuscate it as otherwise our proofs would not go through. Ideally, for padding we would like to argue that it is a proof artifact which

can be safely ignored. Indeed, if we could argue that padding is superfluous and only an artifact of our proof techniques we would immediately strengthen our results and loose the q-bound from our construction.

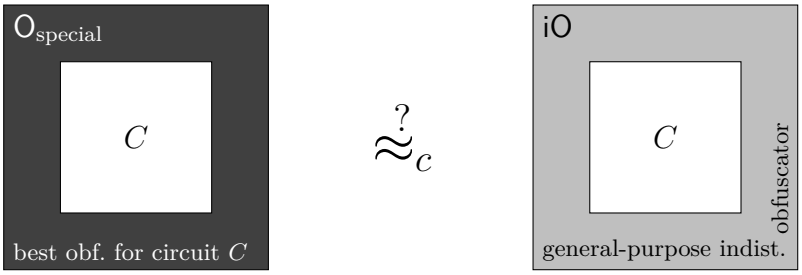
A good intuition as to where padding is used in proofs is given by the formalization of indistinguishability obfuscation as best-possible obfuscation by Goldwasser and Rothblum [GR14] (we provide a more detailed description in Section 5.5.4). Theoretically, for each circuit C we could have a different form of obfuscator that best hides the inner workings of said circuit and we call this obfuscator the best-possible obfuscator for C. Goldwasser and Rothblum argue that an indistinguishability obfuscator is always *as good* as this best-possible obfuscator *if sufficient padding is applied first.* The idea is conveyed by the following picture where on the left a circuit C is first obfuscated according to the best-possible obfuscator $\mathsf{O}_{\text{special}}$ and the result then once more with an indistinguishability obfuscator iO. On the right, the circuit is only padded to then be obfuscated directly with iO.



The security of the indistinguishability obfuscator tells us that the two distributions are computationally indistinguishable and hence iO plus padding (or more precisely padding plus iO) is as good as the best-possible obfuscator. Without the correct amount of padding, on the other hand, the distributions can be easily distinguished as exemplified next:



Here a distinguisher can simply go by size of the obfuscation to distinguish. A more plausible direction which bypasses the size problematic is to compare the distribution induced by the best-possible obfuscator directly with the distribution of the indistinguishability obfuscator. Here, the picture would like like this.



One possibility might be to prove that for every indistinguishability obfuscator there exists a best-possible obfuscator such that their distributions are computationally indistinguishable. While this question on first sight might look like a fruitful research direction, on closer inspection, it turns out that we just transformed the problem. Consider two indistinguishability obfuscators iO_1 and iO_2 where the second obfuscator is defined as first padding its input and then running iO_1. We might now have the situation that for iO_2 there exists a best-possible obfuscator that generates indistinguishable obfuscations but for iO_1 no such best-possible obfuscator exists simply because for some reason the best-possible obfuscator needs some additional space. Also note that we consider the best-possible obfuscator *for* a circuit C. This means that the output size of the best-possible obfuscator may depend on circuit C while the indistinguishability obfuscator needs to work *for all* circuits C and, thus, has a maximum output length depending only on the length of the input circuit.

While best-possible obfuscation provides a good intuition where padding is used in proofs based on indistinguishability obfuscation it may not be the best notion to further study padding as there we are dealing with *two different types* of obfuscators: the best-possible and the indistinguishability obfuscator. On the other hand, we may ask whether an obfuscation scheme O exists such that, whenever two circuits are indistinguishable under obfuscation with prior padding, that is, if

$$\mathsf{O}(\mathsf{PAD}(C_0)) \approx_c \mathsf{O}(\mathsf{PAD}(C_1))$$

that then also the obfuscations without padding are indistinguishable. In other words, we may ask whether there exists an obfuscator O such that

$$\Big(\mathsf{O}(\mathsf{PAD}(C_0)) \approx_c \mathsf{O}(\mathsf{PAD}(C_1))\Big) \implies \Big(\mathsf{O}(C_0) \approx_c \mathsf{O}(C_1)\Big)\,.$$

Bitansky, Garg, and Telang [BGT14] were the first to formalize this question and who called such an obfuscator O a *padding-free obfuscator*. As we will see VBB obfuscators are padding-free but indistinguishability obfuscators are not, at least, not in general.

Padding and UCEs

Our motivation in studying the connection between padding and indistinguishability obfuscation-based constructions are the UCE constructions from Chapter 11 where due to the use of padding we obtain only q-bounded security. In very brought terms UCE security asks whether the following two distributions are indistinguishable:

$$(\mathsf{aux}, \mathsf{hk}) \quad \text{and} \quad (\mathsf{aux}', \mathsf{hk})$$

Here, hk is a uniformly random hash key and aux is generated by an efficient algorithm (source S) with access to an oracle implementing $\mathsf{H}(\mathsf{hk}, \cdot)$ and aux' is generated by the same algorithm S but with oracle access to a random oracle. (Additionally, we require the algorithm to be unpredictable, but this is not important for the current discussion.) For our hash function construction from indistinguishability obfuscation and puncturable PRFs, hash keys hk are obfuscated (and padded) programs. Thus, for us, UCE security boils down to whether the distributions

$$(\mathsf{aux}, \mathsf{O}(\mathsf{PAD}(C))) \quad \text{and} \quad (\mathsf{aux}', \mathsf{O}(\mathsf{PAD}(C)))$$

are computationally indistinguishable where C is the evaluation algorithm of a puncturable pseudorandom function with a randomly chosen PRF key that is hard-coded into the circuit. That is

$$C[\mathsf{k}](\cdot) := \mathsf{PRF}(\mathsf{k}, \cdot).$$

We have shown that if padded sufficiently the above distributions are indeed computationally indistinguishable in case O is an indistinguishability obfuscator (and if certain point-function obfuscators exist; see Chapter 11). Ideally, what we would like to show is that also for the unpadded circuit the distributions

$$(\mathsf{aux}, \mathsf{O}(C)) \quad \text{and} \quad (\mathsf{aux}', \mathsf{O}(C))$$

are computationally indistinguishable as this would immediately lift the q-bound from our constructions. As we currently cannot prove that this is the case we could still conjecture that for a *good obfuscator* O

$$\Big((\mathsf{aux}, \mathsf{O}(\mathsf{PAD}(C)) \approx_c (\mathsf{aux}', \mathsf{O}(\mathsf{PAD}(C)))\Big) \implies \Big((\mathsf{aux}, \mathsf{O}(C)) \approx_c (\mathsf{aux}', \mathsf{O}(C))\Big)$$

thereby extending the *padding-free obfuscation* notion to also factor in auxiliary information.

Remark. Note that in the above, we fixed circuit C to be the same in both distributions. A more relaxed assumption would claim the implication to hold even for distinct circuits C_0 and C_1.

Outline. In the remainder of this chapter we formulate a framework on assumptions called *Superfluous Padding Assumption* (SuPA for short) that allows us to study the role of padding within obfuscation-based proofs. We will see that for VBB obfuscation, padding is, indeed, superfluous but that, on the other hand, for indistinguishability obfuscation even very restricted SuP assumptions cannot be shown to hold. For this, we recall counter-examples due to Holmgren [Hol15] and Bitansky et al. [BCC+14] and present extensions thereof. Finally, we note that not all is lost (yet)[1] and we will show very strong restrictions that are not covered by existing counter examples but which are still sufficient for removing the q-bound from the UCE constructions.

[1] When we originally formulated SuPA [BM15b] we were rather positive that at least a restricted form that is sufficient for our application can be shown to hold (or at least not be shown not to hold). The counter-example due to Holmgren [Hol15], on the other hand, is very strong and it is, thus, unclear if even very restricted forms of SuPA may survive. Indeed, for the case of Turing machine obfuscation we will show that even restrictions that require auxiliary information aux to be empty do not suffice to show that padding is superfluous. For this, however, we require that the two programs are distinct which differs from all application scenarios where we consider identical programs (cf. UCEs) and, thus, also for TM-obfuscation there may be interesting restrictions.

$\underline{\mathsf{SuP}[\mathsf{s}]^{\mathsf{D}}_{\mathsf{O},\mathsf{Sam}}(\lambda)}$

$b \leftarrow\!\!\$\ \{0,1\}$
$(\mathsf{aux}, C) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda, b)$
$\overline{C} \leftarrow\!\!\$\ \mathsf{O}(\mathsf{PAD}(\mathsf{s}(\lambda), C))$
$b' \leftarrow\!\!\$\ \mathsf{D}(1^\lambda, \mathsf{s}, \mathsf{aux}, \overline{C}, |C|)$
return $b = b'$

Figure 13.1: The SuP game is parameterized by a polynomial s. It runs the sampler Sam on input a bit b which outputs a pair of (aux_b, C_b) consisting of auxiliary information and a circuit. Then, according to s the circuit is padded (if $\mathsf{s} = 0$ then the original circuit is used) before it is obfuscated and given to distinguisher D which additionally gets as input auxiliary input aux_b as well as s and the size of the original circuit $|C_b|$. The task of distinguisher D is to guess b.

13.2 The Superfluous Padding Assumption

In the following we extend the padding-free iO notion of Bitansky et al. [BGT14] and present a framework to capture various strength assumptions on the (un)necessity of padding for obfuscation-based techniques. We call our parameterized assumption the *Superfluous Padding Assumption* (SuPA for short) which can be stated for any obfuscator and which is parameterized by a class of efficient samplers. Thus, fixing an obfuscator (for example, an indistinguishability obfuscator) and a class of samplers yields a specific instance of the SuP assumption.

The SuP assumption. We state the SuP assumption in two steps. In the first step we define admissible samplers that are used to parameterize the SuP assumption. We consider sample algorithms that on input a bit b output a pair (aux, C) where aux is a string (i.e., auxiliary information) and C is the description of a circuit. We call a sampler admissible for an obfuscator O, if with sufficient padding s the distributions

$$\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda, 0)\\ \overline{C} \leftarrow\!\!\$\ \mathsf{O}(\mathsf{PAD}(\mathsf{s}, C))\\ \textbf{return}\ (\mathsf{aux}, C)\end{array}\right) \quad \text{and} \quad \left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\ \mathsf{Sam}(1^\lambda, 1)\\ \overline{C} \leftarrow\!\!\$\ \mathsf{O}(\mathsf{PAD}(\mathsf{s}, C))\\ \textbf{return}\ (\mathsf{aux}, C)\end{array}\right)$$

are computationally indistinguishable. This is formally captured as follows:

Definition 13.1. *Let* O *be an obfuscation scheme and let* $\mathsf{PAD} : \mathbb{N} \times \{0,1\}^* \to \{0,1\}^*$ *be a deterministic padding algorithm that takes as input an integer* s *and a description of a circuit* C *and outputs a functionally equivalent circuit of size* $\mathsf{s} + |C|$*. We say that a* PPT *sampler* Sam *is SuP-admissible for obfuscator* O*, if there exists a polynomial* s *such that for any* PPT *distinguisher* D *its advantage in the* $\mathsf{SuP}[\mathsf{s}]$ *game (formalized in Figure 13.1) is negligible:*

$$\mathsf{Adv}^{\mathrm{sup}[\mathsf{s}]}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda) = 2 \cdot \Pr\left[\mathsf{SuP}[\mathsf{s}]^{\mathsf{D}}_{\mathsf{O},\mathsf{Sam}}(\lambda)\right] - 1 \leq \mathsf{negl}(\lambda)$$

Remark. While we did not fully define padding algorithm PAD we assume it to be a simple and canonical padding algorithm that itself does not do any obfuscation and which is easily revertible. In particular we assume $\mathsf{PAD}(0, C) = C$ for any circuit C.

Remember that the size of a circuit is the number of its gates (we consider only Boolean circuits with NOT, AND and OR gates) plus the number of input and output nodes. As an AND gate where both inputs are connected to the same wire is functionality preserving—note that 1 AND 1 = 1 and 0 AND 0 = 0—we could, thus, fix the deterministic padding scheme to add such AND-gates to the first input wire.

SuP-admissibility does not place restrictions on samplers beyond indistinguishability, but potentially, one can put various reasonable additional restrictions on the samplers. One could, for example, require that the marginal distribution on the circuits is identical, or that aux is generated only with oracle access to the functionality provided by circuit C and, thus, cannot depend on the description of C, but only on the functionality of C. In the following we state the *Superfluous Padding Assumption* (SuP assumption) which captures restrictions via restraining the class of admissible samplers that are considered.

Assumption 13.1 (Superfluous Padding Assumption)**.** *Let* O *be an obfuscation scheme, let* $\mathcal{S}$ *be a class of SuP admissible samplers. Then, the* Superfluous Padding Assumption *for class* $\mathcal{S}$ *(for short SuP*$[\mathcal{S}]$*) states that for any sampler* $\mathsf{Sam} \in \mathcal{S}$ *no efficient distinguisher* D *has a non-negligible advantage in the* $\mathsf{SuP}[0]$ *game:*

$$\mathsf{Adv}^{\mathrm{sup}[0]}_{\mathsf{O},\mathsf{Sam},\mathsf{D}}(\lambda) = 2 \cdot \Pr\left[\mathsf{SuP}[0]^{\mathsf{D}}_{\mathsf{O},\mathsf{Sam}}(\lambda)\right] - 1 \leq \mathsf{negl}(\lambda)\,.$$

SuP for UCEs. Before we further analyze the assumption let us establish how it lifts Construction 11.2 from achieving $\mathrm{UCE}_1[\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-}\mathrm{query}}]$ to achieving $\mathrm{UCE}_1[\mathcal{S}^{\mathrm{sup}}]$.

Under the SuP assumption for class $\mathcal{S}$ and obfuscator O it follows that if, for some choice of polynomial s, the two distributions

$$\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 0)\\ \overline{C} \leftarrow\!\!\$\, \mathsf{O}(\mathsf{PAD}(\mathsf{s}, C))\\ \textbf{return } (\mathsf{aux}, C)\end{array}\right) \quad \text{and} \quad \left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 1)\\ \overline{C} \leftarrow\!\!\$\, \mathsf{O}(\mathsf{PAD}(\mathsf{s}, C))\\ \textbf{return } (\mathsf{aux}, C)\end{array}\right)$$

are computationally indistinguishable and, furthermore, sampler Sam is in class $\mathcal{S}$ then we have that also the distributions without additional padding are computationally indistinguishable, that is:

$$\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 0)\\ \overline{C} \leftarrow\!\!\$\, \mathsf{O}(C)\\ \textbf{return } (\mathsf{aux}, C)\end{array}\right) \approx_c \left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 1)\\ \overline{C} \leftarrow\!\!\$\, \mathsf{O}(C)\\ \textbf{return } (\mathsf{aux}, C)\end{array}\right)$$

In other words, if we sample $(\mathsf{aux}_0, C_0) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 0)$ and $(\mathsf{aux}_1, C_1) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 1)$ then we have that

$$(\mathsf{aux}_0, \mathsf{O}(C_0)) \approx_c (\mathsf{aux}_1, \mathsf{O}(C_1))$$

where the probability is over the coins of sampler Sam and the coins of obfuscator O.

In the proof of Theorem 11.1 we show that for any q and any source $\mathsf{S} \in [\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{q}\text{-query}}]$, there exists a polynomial s such that no efficient distinguisher can tell the following two distributions apart

$$\Big(L_1, \mathsf{iO}(\mathsf{PAD}(\mathsf{s}(\lambda), \mathsf{F.Eval}(\mathsf{k}, \cdot))\Big) \approx_c \Big(L_5, \mathsf{iO}(\mathsf{PAD}(\mathsf{s}(\lambda), \mathsf{F.Eval}(\mathsf{k}, \cdot))\Big).$$

Here L_1 is generated as in Game_1 by running source S relative to the actual construction and L_5 is generated as in Game_5 by running source S relative to a random oracle. Applying the SuP assumption for indistinguishability obfuscation and identifying a sampler Sam that on input bit $b = 0$ constructs (aux_0, C_0) by running Game_1 until, and including, line 7 and outputting $(L, \mathsf{F.Eval}(\mathsf{k}, \cdot))$ (see Figure 11.1) and, similarly, on input $b = 1$ constructs (aux_1, C_1) by running Game_5 until line 7 we get that also

$$\Big(L_1, \mathsf{iO}(\mathsf{F.Eval}(\mathsf{k}, \cdot)\Big) \approx_c \Big(L_5, \mathsf{iO}(\mathsf{F.Eval}(\mathsf{k}, \cdot)\Big)$$

are computationally indistinguishable. Consequently, we get that:

Theorem 13.1. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable* $\mathsf{AIPO}[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$*) exist, if the superfluous padding assumption holds for a secure indistinguishability obfuscation scheme (and any SuP-admissible samplers), then Construction 11.2 is* $\mathrm{UCE}_1[\mathcal{S}^{\mathrm{sup}}]$ *secure.*

Remark. For the above application we can further restrict the class of samplers to a class that captures in a much more restricted way what games Game_1 and Game_5 do to sample (L, hk). We'll get back to such restrictions in Section 13.5.

13.3 On the Validity of the Superfluous Padding Assumption

In this section, we show that the Superfluous Padding Assumption holds in an idealized model, namely, we show that it holds for virtual black-box obfuscation.

Theorem 13.2. *The Superfluous Padding Assumption holds for virtual black-box obfuscators.*

Proof. The idea of the proof is to exploit that the simulator of the virtual black-box obfuscator only has black-box access to the obfuscated circuit (and its size) but does not get the circuit itself. Hence, given an *obfuscated* version of the circuit, one can simulate the oracle for the simulator regardless of how the obfuscation looks like and whether the circuit was padded before being obfuscated or not.

Let O be a virtual black-box obfuscator and let Sam be a SuP-admissible sampler. Let D be an efficient distinguisher. We need to prove that

$$\left|\Pr\left[\left(\begin{array}{l}(\mathsf{aux}_0, C_0) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda, 0)\\ \overline{C_0} \leftarrow\!\$\, \mathsf{O}(C_0)\\ \textbf{return } \mathsf{D}(1^\lambda, \mathsf{aux}_0, \overline{C_0}, |C_0|)\end{array}\right) = 1\right] - \Pr\left[\left(\begin{array}{l}(\mathsf{aux}_1, C_1) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda, 1)\\ \overline{C_1} \leftarrow\!\$\, \mathsf{O}(C_1)\\ \textbf{return } \mathsf{D}(1^\lambda, \mathsf{aux}_1, \overline{C_1}, |C_1|)\end{array}\right) = 1\right]\right| \tag{13.1}$$

is negligible. Let Sim be the virtual black-box simulator for D. Note that Sim depends on D, but that Sim does not depend on the circuit. By the security of virtual black-box obfuscation, we have

that for $b \in \{0,1\}$ it holds that

$$\Pr\left[\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\$\, \mathsf{Sam}(1^\lambda, b)\\ \overline{C} \leftarrow\$\, \mathsf{O}(C)\\ \textbf{return } \mathsf{D}(1^\lambda, \mathsf{aux}, \overline{C}, |C|)\end{array}\right) = 1\right] \approx_c \Pr\left[\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\$\, \mathsf{Sam}(1^\lambda, b)\\ \textbf{return } \mathsf{Sim}^{C}(1^\lambda, \mathsf{aux}, |C|)\end{array}\right) = 1\right]. \tag{13.2}$$

Now, if we can prove that

$$\epsilon := \left|\Pr\left[\left(\begin{array}{l}(\mathsf{aux}_0, C_0) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 0)\\ \textbf{return } \mathsf{Sim}^{C_0}(1^\lambda, \mathsf{aux}_0, |C_0|)\end{array}\right) = 1\right] - \Pr\left[\left(\begin{array}{l}(\mathsf{aux}_1, C_1) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 1)\\ \textbf{return } \mathsf{Sim}^{C_1}(1^\lambda, \mathsf{aux}_1, |C_1|)\end{array}\right) = 1\right]\right| \tag{13.3}$$

is negligible, then the result follows by summing up Equations (13.1) and (13.3) to get

$$\begin{aligned}(13.1) + (13.3) \leq \Bigg|&\Pr\left[\left(\begin{array}{l}(\mathsf{aux}_0, C_0) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 0)\\ \overline{C_0} \leftarrow\$\, \mathsf{O}(C_0)\\ \textbf{return } \mathsf{D}(1^\lambda, \mathsf{aux}_0, \overline{C_0}, |C_0|)\end{array}\right) = 1\right] - \Pr\left[\left(\begin{array}{l}(\mathsf{aux}_0, C_0) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 0)\\ \textbf{return } \mathsf{Sim}^{C_0}(1^\lambda, \mathsf{aux}_0, |C_0|)\end{array}\right) = 1\right] +\\ &\Pr\left[\left(\begin{array}{l}(\mathsf{aux}_1, C_1) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 1)\\ \overline{C_1} \leftarrow\$\, \mathsf{O}(C_1)\\ \textbf{return } \mathsf{D}(1^\lambda, \mathsf{aux}_1, \overline{C_1}, |C_1|)\end{array}\right) = 1\right] - \Pr\left[\left(\begin{array}{l}(\mathsf{aux}_1, C_1) \leftarrow\$\, \mathsf{Sam}(1^\lambda, 1)\\ \textbf{return } \mathsf{Sim}^{C_1}(1^\lambda, \mathsf{aux}_1, |C_1|)\end{array}\right) = 1\right]\Bigg|\end{aligned}$$

which by Equation (13.2) is negligible and consequently also Equation (13.1) must describe a negligible function. Thus, it remains to show that ϵ (Equation 13.2) is negligible. Assume not, then we show that Sam is not SuP-admissible. Towards this goal, we construct a distinguisher D^* in the SuP game as follows. Distinguisher D^* receives as inputs $(1^\lambda, \mathsf{s}, \mathsf{aux}, \overline{C}, |C|)$. It then runs $\mathsf{Sim}^{\overline{C}}(1^\lambda, \mathsf{aux}, |C|)$ to receive a bit b' and outputs b'.

As the simulator only has black-box access to the circuit, $\mathsf{Sim}^{C_b}(1^\lambda, \mathsf{aux}_b, |C_b|)$ has exactly the same behavior as $\mathsf{Sim}^{\mathsf{O}(\mathsf{PAD}(\mathsf{s}(\lambda), C_b))}(1^\lambda, \mathsf{aux}_b, |C_b|)$, and thus, the advantage $\mathsf{SuP}[\mathsf{s}]^{\mathsf{D}^*}_{\mathsf{O},\mathsf{Sam}_0,\mathsf{Sam}_1}(\lambda)$ is equal to ϵ. □

13.4 On SuPA for Indistinguishability Obfuscation

We have seen that SuPA holds for VBB obfuscators, but what about indistinguishability obfuscation? Consider the case that it does not hold for iO and let us assume an obfuscator always outputs circuits which are at least twice the size of the input circuit. Then we have constructions (such as our UCE constructions) which might be insecure when instantiated with the above obfuscator but which become secure when the obfuscator is run repeatedly, for example, if we instantiate the construction with $\mathsf{iO}(\mathsf{iO}(C))$. Such a "security amplification" (from an inverse polynomial advantage to negligible advantage for any efficient adversary), however, seems unreasonable for any good obfuscator and indeed seems rare for a security notion based on a distinguishing game.

As it turns out, the above intuition is not correct. Indeed, this follows by a corollary of a result by Bitansky et al. [BCC+14, Theorem 1] as well as by direct counter-example which was independently suggested by Holmgren [Hol15]. The intuition in both cases is that we can embed an obfuscated distinguisher into the auxiliary input such that it reveals a secret (for example, a hidden bit b) only if it gets a "short" description of the obfuscated circuit. If we define *short* such that the unpadded obfuscator outputs short descriptions and such that the padded obfuscator outputs long descriptions,

then the adversary can trivially distinguish in the case where circuits were obfuscated without padding. The crux is to prove that the adversary is not successful, if the circuits were padded. Towards this goal, a counting argument is used.

13.4.1 A Counter-Example for General SuPA due to [BCC+14]

Bitansky et al [BCC+14, Theorem 1] show that assuming the existence of a witness encryption scheme [GGSW13] no function family with super-polynomial pseudoentropy has an average-case VBB obfuscator with dependent auxiliary information. Let us explain how this implies that SuPA without further restrictions does not hold.

Let F be a pseudorandom function and k a uniformly random key, and now consider the function table T for $\mathsf{F.Eval}(\mathsf{k},\cdot)$ that only contains the first q values for some polynomial $\mathsf{q}:\mathbb{N}\to\mathbb{N}$:

$$\begin{aligned} T[1] &:= \mathsf{F.Eval}(\mathsf{k},1)\\ T[2] &:= \mathsf{F.Eval}(\mathsf{k},2)\\ &\vdots\\ T[\mathsf{q}(\lambda)] &:= \mathsf{F.Eval}(\mathsf{k},\mathsf{q}(\lambda)) \end{aligned}$$

A witness encryption scheme allows us to encrypt a value (for example, a bit b) such that anybody who knows a witness to an NP instance can decrypt. For example, we could encrypt a bit b such that anybody who knows a short description for function table T can decrypt. Indistinguishability obfuscation implies witness encryption [SW14] and, thus, in the following we continue the example without formally introducing witness encryption but instead work directly with iO. For this consider the following circuit $C_{\mathsf{aux}}[b,\mathsf{q},T]$ which has a bit b, polynomial q, and function table T hard-coded and which works as follows

$\underline{C_{\mathsf{aux}}[b,\mathsf{q}(\lambda),T](C)}$
for $i\in[\mathsf{q}(\lambda)]$ **do**
  if $T[i]\neq C(i)$ **do**
    return $\bot$
return b

That is, circuit $C_{\mathsf{aux}}[b,\mathsf{q},T]$ takes as single input a circuit description and then checks if this circuit agrees with function table T on inputs 1 to $\mathsf{q}(\lambda)$. There are two things to note: 1) circuit C_{aux} only takes inputs of a specific length and, thus, it only reveals b if it is run on input a *short* description of function table T where we can freely specify what we mean by short; and 2) an indistinguishability obfuscation of C_{aux} is computationally indistinguishable from an obfuscation of the constant zero circuit and, thus, hides b. For this note that if we replace table T with a table T' which consists of truly random values then down to the security of the pseudorandom function, tables T and T' are computationally indistinguishable. However, for T' there cannot be any short program that encodes it and, thus, circuit $C_{\mathsf{aux}}[b,\mathsf{q}(\lambda),T']$ is the constant zero circuit.

With this we are almost at the counter-example. What is left to note is that a polynomial sized function table can be trivially VBB obfuscated since the function table is learnable. However, we also know that there exists a short description for function table T, namely the following:

$C[\mathsf{k}, \mathsf{q}(\lambda)](x)$

if $x \in [\mathsf{q}(\lambda)]$ **then**

  return $\mathsf{F.Eval}(\mathsf{k}, x)$

return $\perp$

Furthermore, by the security of an indistinguishability obfuscator we have that there exists some padding factor s such that

$$\mathsf{iO}(\mathsf{PAD}(\mathsf{s}, C[\mathsf{k}, \mathsf{q}(\lambda)])) \approx_c \mathsf{iO}(T)$$

since function table T and circuit $C[\mathsf{k}, \mathsf{q}(\lambda)]$ are functionally equivalent and with sufficient padding we can blow up the size of $C[\mathsf{k}, \mathsf{q}(\lambda)]$ to match that of table T. It follows that the distributions

$$\left(\mathsf{iO}\left(C_{\mathsf{aux}}[0, \mathsf{q}(\lambda), T]\right), \mathsf{iO}\left(C[\mathsf{k}, \mathsf{q}(\lambda)]\right)\right) \quad \text{and} \quad \left(\mathsf{iO}\left(C_{\mathsf{aux}}[1, \mathsf{q}(\lambda), T]\right), \mathsf{iO}\left(C[\mathsf{k}, \mathsf{q}(\lambda)]\right)\right)$$

are distinguishable if we choose the number of input wires for C_{aux} as $|\mathsf{iO}(\mathsf{F.Eval}(\mathsf{k}, \cdot))|$. On the other hand, we know that with sufficient padding applied to circuit $C[\mathsf{k}, \mathsf{q}(\lambda)]$ we have that the distributions

$$\left(\mathsf{iO}\left(C_{\mathsf{aux}}[0, \mathsf{q}(\lambda), T]\right), \mathsf{iO}\left(\mathsf{PAD}(\mathsf{s}(\lambda), C[\mathsf{k}, \mathsf{q}(\lambda)])\right)\right)$$

and

$$\left(\mathsf{iO}\left(C_{\mathsf{aux}}[1, \mathsf{q}(\lambda), T]\right), \mathsf{iO}\left(\mathsf{PAD}(\mathsf{s}(\lambda), C[\mathsf{k}, \mathsf{q}(\lambda)])\right)\right)$$

are computationally indistinguishable. To see this note that 1) iO is a best-possible obfuscator; 2) circuit $\mathsf{PAD}(\mathsf{s}(\lambda), C[\mathsf{k}, \mathsf{q}(\lambda)])$ implements the same function as function table T and with sufficient padding it is as large as T; 3) a function table can be VBB obfuscated; and 4) from only oracle access to function table T one cannot extract a short description of T down to the security of the pseudorandom function F.

Wrapping up, what we have done is to embed a circuit into the auxiliary input that contains a secret bit b which is revealed on input a short description of a function that agrees with a pseudorandom function on polynomially many values. Such a short description exists (the PRF itself with the key hard-coded). However, an obfuscated version that is sufficiently padded before obfuscation is indistinguishable from a circuit that contains the function table T itself and down to the security of the PRF we know that given T one cannot efficiently find a short description.

Consequently, SuPA without further restrictions on the class of SuP-admissible samplers for indistinguishability obfuscation cannot hold. In the following we show how Holmgren's counter-example further restricts potential classes of SuP-admissible samplers for which we might hope that SuPA holds.

13.4.2 Holmgren's Counter-Example for General SuPA

Holmgren independently suggests a similar counter-example to general SuPA [Hol15] based on puncturable PRFs which allows to show that also restricting SuP samplers to sample as circuit a puncturable PRF with hard-coded key (as is the case for our UCE constructions) is not sufficient for the SuP assumption to hold.

To this end Holmgren uses a puncturable PRF as the obfuscated circuit, and as the auxiliary information, his sampler again outputs the obfuscation of an auxiliary input circuit C_{aux} that has a secret bit b and a polynomial sized list of pseudorandom values hard-coded and that takes as input a short program. On input a short program, circuit C_{aux} tests if this program agrees on all the input/output pairs and if so it outputs the hard-coded bit b. In contrast to the previous counter example, we now consider a circuit that implements a pseudorandom function instead of only a polynomial fraction of a pseudorandom function. That is, we consider the circuit

$\underline{C_1[\mathsf{k}](x)}$
return $\mathsf{F.Eval}(\mathsf{k}, x)$

By $\mathbf{y}$ we denote a vector that contains a polynomial fraction (q many) of input output pairs of the pseudorandom function, that is, we set

for $i = 1, \ldots, \mathsf{q}(\lambda)$ **do**
  $\mathbf{y}[i] \leftarrow \mathsf{PRF.Eval}(\mathsf{k}, i)$

Now down to indistinguishability obfuscation and with sufficient padding an obfuscation of circuit C_1 is computationally indistinguishable from an obfuscation of

$\underline{C_2[\mathsf{k}^*, \mathbf{y}](x)}$
if $x \in [\mathsf{q}(\lambda) + \lambda)]$
  return $\mathbf{y}[x]$
return $\mathsf{F.Eval}(\mathsf{k}^*, x)$

Here, key k^* denotes a punctured key, that is punctured on values $\{1, \ldots, \mathsf{q}(\lambda)\}$. As auxiliary input, Holmgren uses an indistinguishability obfuscation of the following circuit which similarly has vector $\mathbf{y}$ hard-coded and a bit b.

$\underline{C_{\mathsf{aux}}[\mathbf{y}, b](C)}$
for $i \in [\mathsf{q}(\lambda) + \lambda]$ **do**
  if $\mathbf{y}[i] \neq C(i)$ **then**
    return $\perp$
return b

Now, similarly to before, given an obfuscation of $C_{\mathsf{aux}}[\mathbf{y}, b]$ and a short description of pseudorandom function $\mathsf{F.Eval}(\mathsf{k}, \cdot)$—for example, an obfuscation of $C_1[\mathsf{k}]$—one can easily learn bit b. However, given only an obfuscation of $C_2[\mathsf{k}^*, \mathbf{y}]$ one cannot efficiently construct such a short description and hence bit b remains hidden.

We visualize the steps of the argument once more in Figure 13.2 and refer to the paper by Holmgren [Hol15] for a detailed description. In summary we have that

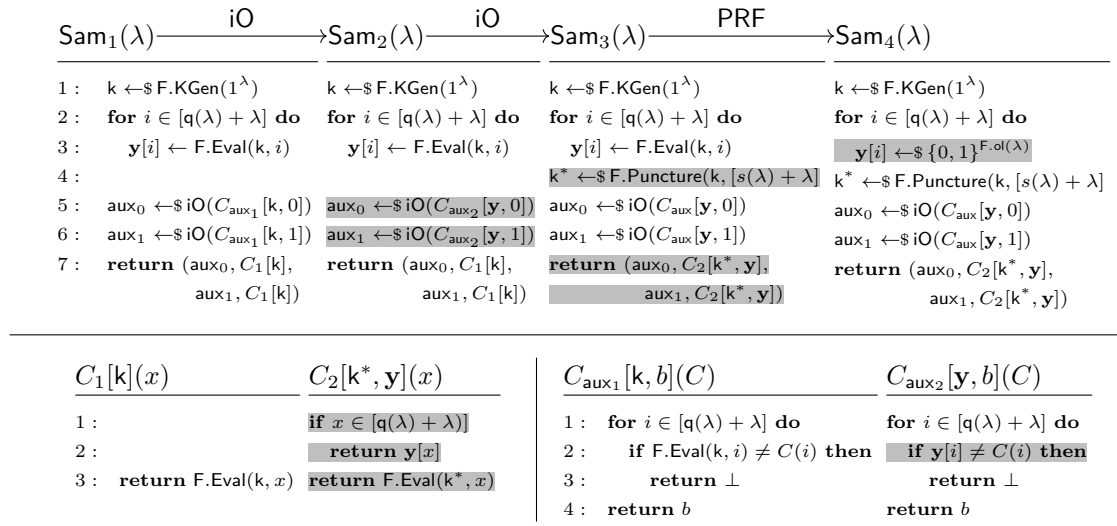


Figure 13.2: The outline of the counter-example of Holmgren showing that there exists an SuP admissible sampler Sam for which SuPA does not hold with respect to indistinguishability obfuscation [Hol15].

Theorem 13.3 ([Hol15], [BCC+14, Theorem 1])**.** *There is an admissible SuP sampler* Sam *such that the SuP assumption with respect to* Sam *does not hold for indistinguishability obfuscators.*

In the following we show that using Holmgren's technique we can rule out a significant class of admissible SuP-samplers when considering indistinguishability obfuscation for Turing machines.

13.4.3 On SuPA for Turing Machine Indistinguishability Obfuscators

A crucial difference between Turing machines and circuits is that Turing machines take arbitrary length inputs. In their seminal paper on obfuscation Barak et al. [BGI+12] use this fact in order to lift their general impossibility result for the impossibility of VBB obfuscation for two Turing machines to the impossibility of one Turing machine by simply encoding the two machines in one (see Section 5.3.2 for a detailed description). If P_0 and P_1 are two Turing machines we denote their combination by $\mathsf{P}_0\#\mathsf{P}_1$ which takes one extra bit as input and if this is b it evaluates P_b on the remainder of the input:

$$\mathsf{P}_0\#\mathsf{P}_1(b\|x) = \begin{cases} \mathsf{P}_0(x) & \text{if } b = 0 \\ \mathsf{P}_1(x) & \text{if } b = 1 \end{cases}$$

Using the same trick one can lift Holmgren's result to not only rule out the SuP assumption in its general form for indistinguishability obfuscators but to further rule out the SuP assumption without auxiliary information for Turing machine indistinguishability obfuscation. For this, we consider a sampler Sam which samples a PRF key $\mathsf{k} \leftarrow\!\!\$\, \mathsf{F.KGen}(1^\lambda)$ and then outputs the combined Turing machine $\mathsf{P}_{\mathsf{aux}}[b]\#\mathsf{F.Eval}(\mathsf{k}, \cdot)$ where $\mathsf{P}_{\mathsf{aux}}$ is similar to Holmgren's distinguishing circuit but which takes an explicit bound for the accepted input length:[2]

[2]Note that as we now consider Turing machines the input length is not a priori fixed.

$\underline{\mathsf{Sam}(1^\lambda, b)}$

$\mathsf{k} \leftarrow\!\!\$\ \mathsf{F.KGen}(1^\lambda)$

$\mathsf{P} \leftarrow \mathsf{P}_{\mathsf{aux}}[b, \mathsf{k}](\cdot)\#\mathsf{F.Eval}(\mathsf{k}, \cdot)$

return $(\varepsilon, \mathsf{P})$

$\underline{\mathsf{P}_{\mathsf{aux}}[b, \mathsf{k}, \mathsf{t}(\lambda)](M)}$

if $|M| \geq \mathsf{t}(\lambda)$ **then**

  return $\perp$

for $i = 1, \ldots, s(\lambda) + \lambda$ **do**

  if $M(i) \neq \mathsf{F.Eval}(\mathsf{k}, i)$ **then**

    return $\perp$

return b

Given P_b as sampled by Sam one can recover b by computing

$$\tilde{\mathsf{P}}_b^0(\tilde{\mathsf{P}}_b^1)$$

for programs $\tilde{\mathsf{P}}_b^d(x) := \mathsf{P}_b(d\|x)$ if polynomial $\mathsf{t}(\lambda) > \left|\tilde{\mathsf{P}}_b^1\right|$. Now, if we fix a Turing machine indistinguishability obfuscator iO and choose t large enough such that $|\mathsf{iO}(\mathsf{P}_b)_1| < \mathsf{t}(\lambda)$ (here $\mathsf{iO}(\mathsf{P}_b)_1$ denotes the obfuscated program with subsequently the first bit fixed) one can still distinguish even after obfuscation. Once we start padding, distinguishing is no longer possible and with sufficient padding Holmgren's puncturable PRF trick can be also applied here. We obtain the following result:

Theorem 13.4. *There exists a SuP-admissible sampler* Sam *that produce empty auxiliary information* $\mathsf{aux}_0 = \mathsf{aux}_1 = \varepsilon$ *such that the SuP assumption with respect to* Sam *does not hold for indistinguishability obfuscators for Turing machines.*

The reason why the attack applies to Turing machines without auxiliary input and might not carry over for circuits without auxiliary input is that the description of a Turing machine is not linear in its running time and that a Turing machine can process arbitrary length inputs. The first property is relevant since a Turing machine with a reasonable encoding of a for-loop with t iterations grows only logarithmically in t, while the corresponding circuit would include t copies of the loop routine. Let us write $\mathsf{P}_b[\mathsf{t}]$ for $\mathsf{P}_{\mathsf{aux}}[b, \mathsf{k}]\#\mathsf{F.Eval}(\mathsf{k}, \cdot)$ when using t iterations of the loop. We need to choose t such that $\mathsf{t} > |\mathsf{iO}(\mathsf{P}_b[\mathsf{t}])_1|$. For a reasonable Turing machine encoding, the latter function is poly-logarithmic in t and hence, when choosing t large enough, it is possible to find a polynomial t that satisfies this inequality. With such a value, the attack goes through against the unpadded version. We then need to choose a padding that is large enough so that we can encode enough uniformly random values so that the counting argument for the padded version allows us to show indistinguishability for the padded version.

As for unbounded input length, note that we need to run the obfuscated program on itself and with circuits we have no guarantee that this is possible. Indeed, in our construction where we first obfuscate the combined program $\mathsf{iO}(\mathsf{P}_b[\mathsf{t}])$ and then fix the first bit, the two resulting programs have the same size, that is, $|\mathsf{iO}(\mathsf{P}_b[\mathsf{t}])_0| = |\mathsf{iO}(\mathsf{P}_b[\mathsf{t}])_1|$ and hence we could not run one on the other if we considered the circuit model of computation.

Not All is Lost

Although at first sight the counter-example for Turing machine obfuscation seems very strong—we do not even require auxiliary information—it is a counter-example for a setting that is different from our applications. In our UCE application we have a fixed program that is obfuscated, namely,

a pseudorandom function with a hard-coded key. Consequently, for our application it is sufficient to consider samplers that output program descriptions that are independent of bit b, that is the marginal distributions

$$\left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 0)\\ \textbf{return } C\end{array}\right) \quad \text{and} \quad \left(\begin{array}{l}(\mathsf{aux}, C) \leftarrow\!\!\$\, \mathsf{Sam}(1^\lambda, 1)\\ \textbf{return } C\end{array}\right)$$

are statistically close or even identical. In this setting, however, auxiliary input is, of course, necessary and, thus, restricted SuP assumptions that bypass the counter-example of Holmgren might work both for circuits as well as for Turing machines. We discuss such possible restrictions next.

13.5 On the Plausibility of Restricted SuP Assumptions

In light of the above counter examples we may ask whether we can further strengthen Holmgren's technique to also rule out the SuP assumption for *circuits* in case we consider samplers without auxiliary information or whether a restricted variant of SuPA might indeed allow us to remove the q-bound from our UCE constructions.

The positive result for VBB obfuscation might be interpreted in favor of some form of SuPA also surviving for indistinguishability obfuscation. Furthermore, very restrictive variations may be considered if not for UCEs, then maybe in different contexts. We will present one such example in the following chapter where we give a direct construction of a deterministic public-key encryption scheme and where we can restrict the auxiliary information to be indistinguishable from uniformly random strings. On the other hand, the techniques in the above counter-examples are very flexible and variants lend themselves to a large class of restricted versions of SuPA. Thus, unlike we thought originally[3], it is difficult to come up with sensible restrictions that are natural/general and allow for interesting applications. We think that coming up with such natural and useful restrictions is an worthwhile task and make a first step in that direction by proposing a (quite narrow) restriction that still suffices for our application of removing the q-bound from our UCE constructions. Namely, we restrict the samplers to be very close to the samplers in the proof of Theorem 11.1: we can require that the circuit is always a PRF (or puncturable PRF or even the GGM construction [GGM84] or even the GGM construction with a particular PRG) and restrict the auxiliary information to be generated with only oracle access to the circuit and to be statistically unpredictable (similarly to the restriction in the UCE setting). (We present a formal definition shortly.) With SuPA restrictions—somewhat similar to concrete UCE notions—we need to be careful not to state restrictions that can no longer be cryptanalyzed. For example, we could restrict the class of samplers so much as to assume that padding in Construction 11.2 is unnecessary; however, it would be very difficult to prove, analyze or refute as it only applies to a very specific use case. We, thus, hope that our discussion here prompts further research into the role of padding in obfuscation.

Padding in obfuscation is a highly non-trivial artifact of constructions that we do not understand well yet. Maybe further understanding on this front also provides a better understanding of unbounded UCEs. For this note that as we do not have any other candidate construction for unbounded UCEs for statistically unpredictable sources in the standard-model, it is potentially possible that such

[3] As mentioned, when we first proposed SuPA we were not aware of the counter examples.

strong UCEs do not exist and, thus, that the q-bound in Construction 11.2 is, in fact, necessary. We think that an answer to this question is very relevant and also relates to the existence of unbounded versions of deterministic encryption and maybe even unbounded correlation-secure hash-functions. To this end, in Chapter 14 we present an extension to our constructions from Chapter 11 that directly yields a deterministic public-key encryption scheme and which may allow for even more restricted SuP-samplers, namely, circuit samplers where the auxiliary information must be indistinguishable from a random string. This construction, however, does not meet UCE security but directly yields the security of a deterministic public-key encryption scheme and at the same time that of a correlated-input secure hash function.

13.5.1 A Restricted Class of SuPA Samplers

In the following we define the sampler class $\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\mathsf{prf}}$ of *statistically unpredictable PRF samplers*.

13.5.2 PRF Samplers

For PRF samplers ($\mathcal{S}^{\mathsf{prf}}$) we consider only such SuP-admissible samplers Sam which can be decomposed into two parts where the first part samples a PRF key for some PRF family and the second part $\mathsf{Sam}_{\mathsf{aux}}$ samples the auxiliary information only with oracle access to the PRF with the key hard-coded. Formally, we say that $\mathsf{Sam} \in \mathcal{S}^{\mathsf{prf}}$ iff Sam can be written as

$$\begin{array}{l}\underline{\mathsf{Sam}(1^\lambda, b)} \\ C \leftarrow\!\!\$\ \mathsf{Sam}_{\mathsf{circ}}(1^\lambda) \\ \mathsf{aux} \leftarrow\!\!\$\ \mathsf{Sam}_{\mathsf{aux}}^{C(\cdot)}(1^\lambda, b) \\ \textbf{return}\ (\mathsf{aux}, C)\end{array}$$

where $\mathsf{Sam}_{\mathsf{circ}}$ and $\mathsf{Sam}_{\mathsf{aux}}$ are PPT algorithms and $\mathsf{Sam}_{\mathsf{aux}}$ takes as input the security parameter and a single bit and gets oracle access to circuit C. Note that this requires that the sampler outputs the same circuit independently of bit b. Furthermore, we require that no PPT adversary that is given either oracle access to C or oracle access to a random function with the same domain and range as C can distinguish in which world it lives, that is, for all PPT algorithms D it holds that

$$\left|\Pr\left[\mathsf{D}^C(1^\lambda) = 1\right] - \Pr\left[\mathsf{D}^f(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where the probability is over the random coins of D and additionally in the first case of sampler $\mathsf{Sam}_{\mathsf{circ}}$ and in the second case over the choice of f. In other words, we require C to be a PRF.[4]

13.5.3 Statistically Unpredictable Samplers

We define statistically unpredictable samplers similarly to statistically unpredictable sources for UCEs. That is, a sampler is said to be statistically unpredictable ($\mathsf{Sam} \in \mathcal{S}^{\text{sup}}$) iff Sam can be decomposed into a first part that samples a circuit (not necessarily a PRF) and a second part which samples the auxiliary information but only with oracle access to the first part. In addition we now require that the oracle queries are statistically unpredictable. Formally, we say that $\mathsf{Sam} \in \mathcal{S}^{\text{sup}}$ iff Sam can be written as

[4] A more stringent restriction could be to require that the sampler not only samples a PRF but a puncturable PRF or even a specific puncturable PRF.

$$\frac{\mathsf{Sam}(1^\lambda, b)}{C \leftarrow\!\!\$\, \mathsf{Sam}_{\mathsf{circ}}(1^\lambda)}$$
$$\mathsf{aux} \leftarrow\!\!\$\, \mathsf{Sam}_{\mathsf{aux}}^{C(\cdot)}(1^\lambda, b)$$
$$\textbf{return } (\mathsf{aux}, C)$$

where $\mathsf{Sam}_{\mathsf{circ}}(1^\lambda)$ and $\mathsf{Sam}_{\mathsf{aux}}$ are a PPT algorithms and $\mathsf{Sam}_{\mathsf{aux}}$ takes as input the security parameter and a single bit and gets oracle access to circuit C. Furthermore, we require that for all (even unbounded) predictors Pred the advantage

$$\mathsf{Adv}^{\mathrm{pred}}_{\mathsf{Sam}_{\mathsf{aux}},\mathsf{P}}(\lambda) := \Pr\left[\mathrm{Pred}^{\mathsf{P}}_{\mathsf{Sam}_{\mathsf{aux}}}(\lambda) = 1\right],$$

is negligible, where game $\mathrm{Pred}^{\mathsf{P}}_{\mathsf{Sam}}(\lambda)$ is shown in Figure 13.3. Note that this is almost exactly the same unpredictability game as for UCEs (with the small difference that sampler $\mathsf{Sam}_{\mathsf{aux}}$ takes as input a bit b) and that similarly to UCEs we consider unpredictability only in the random oracle setting.

Remark. Again, we could further restrict the unpredictability restriction to, for example, not provide the auxiliary input sampler $\mathsf{Sam}_{\mathsf{aux}}$ with bit b as input but instead give it access to either the correct functionality in case $b = 1$ or to a random function in case $b = 0$.

Combining both PRF samplers and statistically unpredictable samplers we obtain a rather restricted form of the SuP assumption which, however, is still sufficient to remove the q-bound in our UCE construction.

Theorem 13.5. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable* $\mathsf{AIPO}[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$*) exist, if the superfluous padding assumption holds for all samplers in class* $\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}$*, then Construction 11.2 is* $\mathrm{UCE}_1[\mathcal{S}^{\mathrm{sup}}]$ *secure.*

13.5.4 Plausibility of SuP$[\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}]$

A "good" restriction for the SuP assumption is a restriction that is (1) strong enough for applications and (2) not susceptible to attacks. Restricting samplers to come from class $\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}$ suffices for our application of lifting the q-bound in our construction of UCEs secure against statistically unpredictable sources. Regarding attacks, we note that the sampler that samples the auxiliary input does not get access to the description of the circuit anymore and, thus, does not get to see the PRF key k. Nevertheless, even with oracle access, one can still implement Holmgren's attack circuit using oracle queries. However, once, we require statistical unpredictability, such obfuscation-based attacks seem to be prevented. This is somewhat similar to requiring statistical unpredictability for UCE sources to work around obfuscation-based attacks. In both cases the intuition is that allowing the predictor to run in unbounded time prevents obfuscation from being useful as the predictor can easily recover the original circuit. While for UCEs this intuition so far has proven correct we need further research to also gain confidence in the corresponding SuP assumption. For this we also note that since we still allow $\mathsf{Sam}_{\mathsf{aux}}$ to take bit b as input the SuP restriction is much weaker than in the UCE case and we might find that we further need to strengthen the restriction (see also the remark after the presentation of sampler class $\mathcal{S}^{\mathrm{sup}}$).

$\underline{\mathrm{Pred}^{\mathsf{P}}_{\mathsf{Sam}_{\mathsf{aux}}}(\lambda)}$

$\mathsf{done} \leftarrow \mathbf{false};\ Q \leftarrow \{\}$
$b \leftarrow\!\!\$\ \{0,1\}$
$\mathsf{aux} \leftarrow\!\!\$\ \mathsf{S}^{\mathsf{RO}}_{\mathsf{aux}}(1^\lambda, b)$
$\mathsf{done} \leftarrow \mathbf{true}$
$Q' \leftarrow\!\!\$\ \mathsf{P}^{\mathsf{RO}}(1^\lambda, \mathsf{aux})$
$\mathbf{return}\ \ (Q \cap Q' \neq \{\})$

$\underline{\mathsf{RO}(x)}$

$\mathbf{if}\ \mathsf{done} = \mathbf{false}\ \mathbf{then}$
  $Q \leftarrow Q \cup \{x\}$
$\mathbf{if}\ T[x] \neq \bot$
  $T[x] \leftarrow\!\!\$\ \{0,1\}^{\mathsf{ol}(\lambda)}$
$\mathbf{return}\ T[x]$

Figure 13.3: The prediction game for defining unpredictable samplers. Here function ol describes the number of output wires of circuits sampled by the corresponding $\mathsf{Sam}_{\mathsf{circ}}$.

We would like to stress the feature (inherited from the UCE definition) that considering unpredictability only with respect to a random function makes the sampler independent of a particular choice of function. One might wonder why we additionally need the PRF restriction. Note that for UCEs, a similar restriction (although not exactly the same restriction as UCE security does not necessarily imply PRF security) is implicit in the definition. Moreover, without this restriction, Holmgren's attack can be extended: Consider only the class of statistically unpredictable samplers $\mathcal{S}^{\mathrm{sup}}$ and consider the sampler due to Holmgren (see Section 13.4). First note, that Holmgren's sampler can be decomposed into two parts where the first one samples the pseudorandom function and the second samples the auxiliary information only with oracle access to the PRF. Now consider the following circuit which we use instead of the PRF

$\underline{C[\mathsf{k}](x_1 \| x_2)}$

$\mathbf{return}\ \mathsf{PRF.Eval}(\mathsf{k}, x_2)$

That is, the circuit has a PRF key k hard-coded and on input $x_1 \| x_2$ (such that $|x_1| = |x_2|$) it evaluates the PRF on x_2. Thus, if we slightly adapt Holmgren's sampler and let it make random queries rather than query $(1, 2, \ldots)$ to the oracle and furthermore have the test circuit choose x_1 as the zero-string then it is easy to see that the sampler is indeed statistically unpredictable and the attack still applies.

In conclusion, the restrictions proposed for the application are quite substantial and for many potential restrictions we found that variants of Holmgren's attack still apply. In the light of overwhelming negative examples we put out the restrictions of $\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}$ not so much because we believe that $\mathrm{SuP}[\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}]$ can be shown to hold but to foster further research into padding within obfuscation-based techniques and to provide a concrete target to attack. (Naturally, if it turns out the $\mathrm{SuP}[\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{prf}}]$ or even a more restrictive version that is still sufficient for our applications can be shown to hold that would be fantastic.) With respect to this last point we also want to lead over to the next and final chapter of this thesis where we show that an extension of our UCE construction can be quite powerful and also directly be used to instantiate interesting applications, namely, deterministic public-key encryption and correlated-input secure hash functions. One particularly interesting feature is that this construction which will also be q-bounded may allow for an SuP assumption to lift the q-bound which is quite different from the ones we discussed so far. There we can require that the auxiliary information is computationally indistinguishable from a uniformly random string which, intuitively, should safeguard against exploiting complex obfuscations such as

the ones that we have seen in the counter-examples. As briefly discussed also in Section 8.7 we know little about whether interesting obfuscation schemes can exist that generate random looking code.

Chapter 14

Beyond UCEs—A Direct Construction of D-PKEs

> "Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number — there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method."
>
> John von Neumann, 1951

Summary. In this chapter we present a direct construction of a q-query deterministic public-key encryption scheme from indistinguishability obfuscation, puncturable PRFs and point-function obfuscation. The advantage of our direct construction over an indirect construction via UCE[$\mathcal{S}^{\text{sup}}$] as presented by Bellare and Hoang [BH15] is that we can base our construction on slightly simpler assumptions (no lossy trapdoor functions and less restrictive point obfuscation) but mostly our construction may yield a fully secure D-PKE scheme as we can show that a very restricted SuP assumption—much more restricted than the restricted assumptions we gave for UCEs in Chapter 13—is sufficient to lift the q-bound. The results in this chapter are based on an unpublished manuscript [BM15a].

Chapter content

14.1 Introduction

UCEs, or more precisely, a hash function that is UCE[$\mathcal{S}^{\text{sup}}$]-secure—UCE secure with respect to statistically unpredictable sources—allows to instantiate the construction of a fully IND-secure deterministic public-key encryption scheme by Bellare and Hoang [BH15] if additionally assuming the existence of lossy trapdoor functions (a primitive not known to be implied by indistinguishability obfuscation and one-way functions). In the previous chapters we showed how to construct such UCE[$\mathcal{S}^{\text{sup}}$]-secure functions from indistinguishability obfuscation and point-function obfuscation secure in the presence of statistically hard-to-invert auxiliary information (AIPO[$\mathcal{S}^{\text{sup}}_{\mathsf{po}}$]) if we additionally restrict sources to make at most q-many queries. The q-bound in the construction is due to the use of padding in the proof and we have seen in the previous chapter that it is rather unclear

whether a form of the *superfluous padding assumption* for indistinguishability obfuscation can be shown to hold which allows us to remove the q-bound.

In this chapter we present a direct construction of a deterministic public-key encryption scheme which at the same time will be correlated-input secure. The construction can be seen as an extension to our UCE construction which is noteworthy for three points: (1) our direct construction of D-PKEs does not require the use of lossy trapdoor functions as does the construction by Bellare and Hoang [BH15];[1] (2) the construction uses weaker assumptions on the existence of point function obfuscation schemes, namely, we only require point obfuscators that are secure on any unpredictable distribution but without additional auxiliary information; and (3) it allows for an intriguing and strong restriction on SuP-samplers for removing the q-bound.

More formally, assuming indistinguishability obfuscation, q-composable point-function obfuscation (AIPO[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}$]) and puncturable pseudorandom functions, we construct a

(1) q-bounded, fully \$-IND-secure deterministic public-key encryption.

Note that \$-IND is an even stronger requirement for deterministic public-key encryption schemes than IND and intuitively asks that encryptions of unpredictable messages are indistinguishable from uniformly random strings. (We formally introduce deterministic public-key encryption and the IND and \$-IND notions in Section 4.5.) A simple corollary of our result is that the construction is simultaneously also a

(2) q-bounded, fully IND-secure deterministic public-key encryption and

(3) q-bounded, correlation-secure hash-functions.

We note that for the construction of (2) (that is of a fully IND-secure D-PKE scheme) we can further weaken the requirements on the point obfuscator to be AIPO[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\mathsf{aux}}_{\mathsf{po}}$].

Our constructed primitive is in some respects stronger than our UCE construction since it comes with a trapdoor that allows *efficient* inversion. We here want to stress that *efficient* should not be understood as "polynomial time" but as "efficient in practice". Similarly to our UCE construction, our D-PKE construction will consist of an indistinguishability obfuscation of a pseudorandom function (more details follow shortly) and, thus, evaluating the function will entail evaluating an obfuscated circuit. Inversion, however, will simply involve evaluating an (unobfuscated) pseudorandom function.

In the following we present the high-level ideas of the construction and then in Section 14.3 formally present the security proof.

14.2 Inserting Trapdoors into Obfuscated PRFs

Building on our techniques for constructing UCEs (see Chapter 11), we base our construction on indistinguishability obfuscation and puncturable PRFs and use point-function obfuscation only in the proof to circumvent Wichs' impossibility result. Let us quickly recall our UCE construction.

[1] We note that it is an open research problem whether lossy trapdoor functions can be constructed from indistinguishability obfuscation and one-way functions.

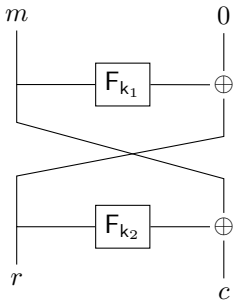
There we use indistinguishability obfuscation to obfuscate a puncturable PRF with a hard-coded key. That is, if F is a puncturable PRF then we construct a UCE$[\mathcal{S}^{\text{sup}} \cap \mathcal{S}^{\text{q-query}}]$ as

$$\mathsf{iO}(\mathsf{PAD}(\mathsf{F}(\mathsf{k}, \cdot))),$$

where padding depends most notably on bound q. Starting from this construction, we need to overcome several obstacles to obtain a deterministic public-key encryption scheme. Most notably, our above construction does not come with a trapdoor which we need to provide in order to be able to decrypt. Indeed, our above construction might not even be injective.

In the symmetric setting a folklore construction of a (invertible) pseudorandom permutation from a pseudorandom function is the Feistel construction. Luby and Rackoff's classic result [LR86] tells us that three Feistel rounds suffice to obtain a pseudorandom permutation and four rounds yield a strong pseudorandom permutation. Applying a two round Feistel to our UCE construction, that is, applying the two round Feistel to the underlying PRF before obfuscating yields the following underlying construction (left we give the pseudocode and on the right we give a schematic view of the two-round Feistel):

$\underline{C'[\mathsf{k}_1, \mathsf{k}_2](m)}$
$r \leftarrow \mathsf{F}(\mathsf{k}_1, m)$
$t \leftarrow \mathsf{F}(\mathsf{k}_2, r)$
$c \leftarrow t \oplus m$
return (r, c)



Knowing *key* k_2 and given *ciphertext* (r, c), one can invert the function and recover m. (Note that k_1 is not needed for inversion.) Our final construction will be similar. The secret key will be k_2 (henceforth called sk) and as public key we output an obfuscation of the above Feistel network but with two independent pseudorandom functions F_r and F_t, where F_r is length doubling (this is crucial for the security proof). The following is the underlying circuit that is obfuscated to become the public key:

$\underline{C[\mathsf{k}, \mathsf{sk}](m)}$
$r \leftarrow \mathsf{F}_r(\mathsf{k}, m)$
$t \leftarrow \mathsf{F}_t(\mathsf{sk}, r)$
$c \leftarrow t \oplus m$
return (r, c)

It is not clear how to argue security, because, giving out r might allow the adversary to compute $t \leftarrow \mathsf{F}_t(\mathsf{sk}, r)$. After all, the public-key contains an obfuscation of $\mathsf{F}_t(\mathsf{sk}, \cdot)$. However, the crux is, that the obfuscated circuit $C[k, \mathsf{sk}]$ only computes $t \leftarrow \mathsf{F}_t(\mathsf{sk}, r)$ when given some m that maps to r. And, in some sense, if the adversary is able to recover m, then it broke the scheme even without running the circuit.

While carrying out this argument formally requires some care and quite a bit of heavy machinery, there is also a catch in the intuition. Namely, if F_r has collisions then the adversary might be able to find a value that is *different* from m, but is also mapped to the same value by F_r. We will counter this issue via two techniques. Firstly, in the proof, we will puncture F_r on m so that F_r returns a random r^*. Secondly, we make F_r to be length-doubling. Then, the random string r^* is unlikely to lie in the image of the pseudorandom function $\mathsf{F}_r(\mathsf{sk}, .)$. Hence, r^* is only output on input m.

14.3 Constructing q-query Deterministic Public-Key Encryption

In the following we present our construction of a deterministic public-key encryption scheme and show that it achieves \$-IND security against adversaries seeing at most q many ciphertexts where q is an arbitrary polynomial that needs to be specified for key-generation. We then discuss in Section 14.4 how restricted SuP assumptions may be used to lift the result from q-bounded to unbounded.

Construction 14.1. *Let $\mathsf{q} : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ be polynomials. Let F_r an F_t be two puncturable pseudorandom functions with $\mathsf{F}_r.\mathsf{ol}(\lambda) = 2 \cdot \mathsf{F}_r.\mathsf{il}(\lambda)$ that is, PRF F_r has a stretch 2. We further require that $\mathsf{F}_t.\mathsf{il}(\lambda) = \mathsf{F}_r.\mathsf{ol}(\lambda)$ and $\mathsf{F}_t.\mathsf{ol}(\lambda) = \mathsf{F}_r.\mathsf{il}(\lambda)$. Let iO be an indistinguishability obfuscator for all circuits. We define a q-query deterministic public-key encryption scheme $\mathsf{D\text{-}PKE} = (\mathsf{D\text{-}PKE.KGen}, \mathsf{D\text{-}PKE.Enc}, \mathsf{D\text{-}PKE.Dec})$ with associated input length function $\mathsf{D\text{-}PKE.il}(\lambda) := \mathsf{F}_r.\mathsf{il}(\lambda)$ as*

$\underline{\mathsf{D\text{-}PKE.KGen}(1^\lambda)}$	$\underline{\mathsf{D\text{-}PKE.Enc}(\mathsf{pk}, m)}$	$\underline{\mathsf{D\text{-}PKE.Dec}(\mathsf{sk}, (r, c))}$	$\underline{C[\mathsf{k}, \mathsf{sk}](m)}$
$\mathsf{k} \leftarrow\!\!\$\ \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$	$\overline{C} \leftarrow \mathsf{pk}$	$t \leftarrow \mathsf{F}_t(\mathsf{sk}, r)$	$r \leftarrow \mathsf{F}_r(\mathsf{k}, m)$
$\mathsf{sk} \leftarrow\!\!\$\ \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$	$(r, c) \leftarrow \overline{C}(m)$	$m \leftarrow c \oplus t$	$t \leftarrow \mathsf{F}_t(\mathsf{sk}, r)$
$\overline{C} \leftarrow\!\!\$\ \mathsf{iO}(\mathsf{PAD}(\mathsf{s}(\lambda), C[\mathsf{k}, \mathsf{sk}](\cdot)))$	**return** (r, c)	**return** m	$c \leftarrow t \oplus m$
$\mathsf{pk} \leftarrow \overline{C}$			**return** (r, c)
return $(\mathsf{pk}, \mathsf{sk})$			

Here, $\mathsf{PAD} : \mathbb{N} \times \{0,1\}^* \longrightarrow \{0,1\}^*$ denotes a deterministic padding algorithm that takes as input an integer s and a description of a circuit C and outputs a functionally equivalent circuit padded to length $|C| + s(\lambda)$. Function s needs to be chosen in accordance with the puncturable PRF and a point obfuscation scheme PO to allow for puncturing F on q points and embedding q many point obfuscations within circuit $\overline{C}$.

Let us first observe that the scheme is correct, that is, for all messages $m \in \{0,1\}^{\mathsf{D\text{-}PKE.il}(\lambda)}$ it holds that

$$\Pr\left[\mathsf{D\text{-}PKE.Dec}\left(\mathsf{sk}, \mathsf{D\text{-}PKE.Enc}(\mathsf{pk}, m)\right) \;\middle|\; (\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{D\text{-}PKE.KGen}(1^\lambda)\right] = 1$$

We, thus, in the following concentrate on proving \$-IND-security. For a formal introduction to the notion of deterministic public-key encryption schemes and the \$-IND-security notion we refer to Section 4.5.

Theorem 14.1. *If F_t and F_r are secure puncturable pseudorandom functions with $\mathsf{F}_r.\mathsf{il}(\lambda) \in \omega(\log(\lambda))$, if iO is a secure indistinguishability obfuscator, and if a q-composable AIPO[$\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}$] (point*

$\mathsf{Game}_1^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$ $\xrightarrow{\mathsf{iO}}$	$\mathsf{Game}_2^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$ $\xrightarrow{\mathsf{pPRF}}$	$\mathsf{Game}_3^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$ $\xrightarrow{\mathsf{iO}}$	$\mathsf{Game}_4^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)$ $\xrightarrow{\mathsf{pPRF}}$
$\mathbf{m} \leftarrow\$\, \mathcal{A}_1(1^\lambda)$	$\mathbf{m} \leftarrow\$\, \mathcal{A}_1(1^\lambda)$	$\mathbf{m} \leftarrow\$\, \mathcal{A}_1(1^\lambda)$	$\mathbf{m} \leftarrow\$\, \mathcal{A}_1(1^\lambda)$
$\mathsf{k} \leftarrow\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$	$\mathsf{k} \leftarrow\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$	$\mathsf{k} \leftarrow\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$	$\mathsf{k} \leftarrow\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$
$\mathsf{sk} \leftarrow\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$	$\mathsf{sk} \leftarrow\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$	$\mathsf{sk} \leftarrow\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$	$\mathsf{sk} \leftarrow\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$
	for $i = 1..\lvert\mathbf{m}\rvert$ **do**	**for** $i = 1..\lvert\mathbf{m}\rvert$ **do**	**for** $i = 1..\lvert\mathbf{m}\rvert$ **do**
	$\overline{\mathbf{p}}[i] \leftarrow\$\, \mathsf{PO}(\mathbf{m}[i])$	$\overline{\mathbf{p}}[i] \leftarrow\$\, \mathsf{PO}(\mathbf{m}[i])$	$\overline{\mathbf{p}}[i] \leftarrow\$\, \mathsf{PO}(\mathbf{m}[i])$
	$\mathbf{r}^*[i] \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, \mathbf{m}[i])$	$\mathbf{r}^*[i] \leftarrow\$\, \{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}$	$\mathbf{r}^*[i] \leftarrow\$\, \{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}$
			$\mathbf{t}^*[i] \leftarrow \mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}, \mathbf{r}^*[i])$
	$\mathsf{k}^* \leftarrow\$\, \mathsf{F}_r.\mathsf{Puncture}(\mathsf{k}, \mathbf{m})$	$\mathsf{k}^* \leftarrow\$\, \mathsf{F}_r.\mathsf{Puncture}(\mathsf{k}, \mathbf{m})$	$\mathsf{k}^* \leftarrow\$\, \mathsf{F}_r.\mathsf{Puncture}(\mathsf{k}, \mathbf{m})$
			$\mathsf{sk}^* \leftarrow\$\, \mathsf{F}_t.\mathsf{Puncture}(\mathsf{sk}, \mathbf{r}^*)$
$\overline{C} \leftarrow\$\, \mathsf{iO}(C_1[\mathsf{k}, \mathsf{sk}])$	$\overline{C} \leftarrow\$\, \mathsf{iO}(C_2[\mathsf{k}^*, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*])$	$\overline{C} \leftarrow\$\, \mathsf{iO}(C_2[\mathsf{k}^*, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*])$	$\overline{C} \leftarrow\$\, \mathsf{iO}(C_3[\mathsf{k}^*, \mathsf{sk}^*, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{t}^*])$
$\mathsf{pk} \leftarrow \overline{C}$	$\mathsf{pk} \leftarrow \overline{C}$	$\mathsf{pk} \leftarrow \overline{C}$	$\mathsf{pk} \leftarrow \overline{C}$
for $i = 1..\lvert\mathbf{m}\rvert$ **do**	**for** $i = 1..\lvert\mathbf{m}\rvert$ **do**	**for** $i = 1..\lvert\mathbf{m}\rvert$ **do**	**for** $i = 1..\lvert\mathbf{m}\rvert$ **do**
$\mathbf{c}[i] \leftarrow \overline{C}(\mathbf{m}_b[i])$	$\mathbf{c}[i] \leftarrow C(\mathbf{m}_b[i])$	$\mathbf{c}[i] \leftarrow C(\mathbf{m}_b[i])$	$\mathbf{c}[i] \leftarrow C(\mathbf{m}_b[i])$
$b' \leftarrow\$\, \mathcal{A}_2(1^\lambda, \mathsf{pk}, \mathbf{c})$	$b' \leftarrow\$\, \mathcal{A}_2(1^\lambda, \mathsf{pk}, \mathbf{c})$	$b' \leftarrow\$\, \mathcal{A}_2(1^\lambda, \mathsf{pk}, \mathbf{c})$	$b' \leftarrow\$\, \mathcal{A}_2(1^\lambda, \mathsf{pk}, \mathbf{c})$
return $(1 = b')$	**return** $(1 = b')$	**return** $(1 = b')$	**return** $(1 = b')$

$C_1[\mathsf{k}, \mathsf{sk}](m)$

$r \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, m)$
$t \leftarrow \mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}, r)$
$c \leftarrow m \oplus t$
return (r, c)

$C_2[\mathsf{k}^*, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*](m)$

if $\exists i \in [|\overline{\mathbf{p}}|] : \overline{\mathbf{p}}[i](m) \neq \perp$ **then**
  $r \leftarrow \mathbf{r}^*[i]$
else
  $r \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}^*, m)$
$t \leftarrow \mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}, r)$
$c \leftarrow m \oplus t$
return (r, c)

$C_3[\mathsf{k}^*, \mathsf{sk}^*, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{t}^*](m)$

if $\exists i \in [|\overline{\mathbf{p}}|] : \overline{\mathbf{p}}[i](m) \neq \perp$ **then**
  $r \leftarrow \mathbf{r}^*[i]$
  $t \leftarrow \mathbf{t}^*[i]$
else
  $r \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}^*, m)$
  $t \leftarrow \mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}^*, r)$
$c \leftarrow m \oplus t$
return (r, c)

Figure 14.1: The games used in the proof of Theorem 14.1 on the top and the corresponding circuit descriptions on the bottom.

obfuscation secure in the presence of statistically hard-to-invert auxiliary information) exists, then Construction 14.1 yields a \$-IND*-secure* q*-query deterministic public-key encryption scheme.*

In case the assumption on point obfuscation is reduced to q*-composable AIPO*$[\mathcal{S}_{\mathsf{po}}^{\mathrm{sup}} \cap \mathcal{S}_{\mathsf{po}}^{\not{\mathsf{aux}}}]$*, then Construction 14.1 yields an IND-secure* q*-query deterministic public-key encryption scheme.*

Before we proof Theorem 14.1 we note that a simple corollary of the above is that Construction 14.1 also yields a correlated-input secure hash function (see Section 4.2 for an introduction).

Corollary 14.2. *If* F_t *and* F_r *are secure puncturable pseudorandom functions with* $\mathsf{F}_r.\mathsf{il}(\lambda) \in \omega(\log(\lambda))$*, if* iO *is a secure indistinguishability obfuscator, and if a* q*-composable AIPO*$[\mathcal{S}_{\mathsf{po}}^{\mathrm{sup}}]$ *(point obfuscation secure in the presence of statistically hard-to-invert auxiliary information) exists, then Construction 14.1 yields a* q*-query CIH-secure hash function.*

Proof of Theorem 14.1. We prove security via a sequence of 10 games and begin with a textual description of all games. We present the accompanying pseudocode in Figures 14.1, 14.2 and 14.3.

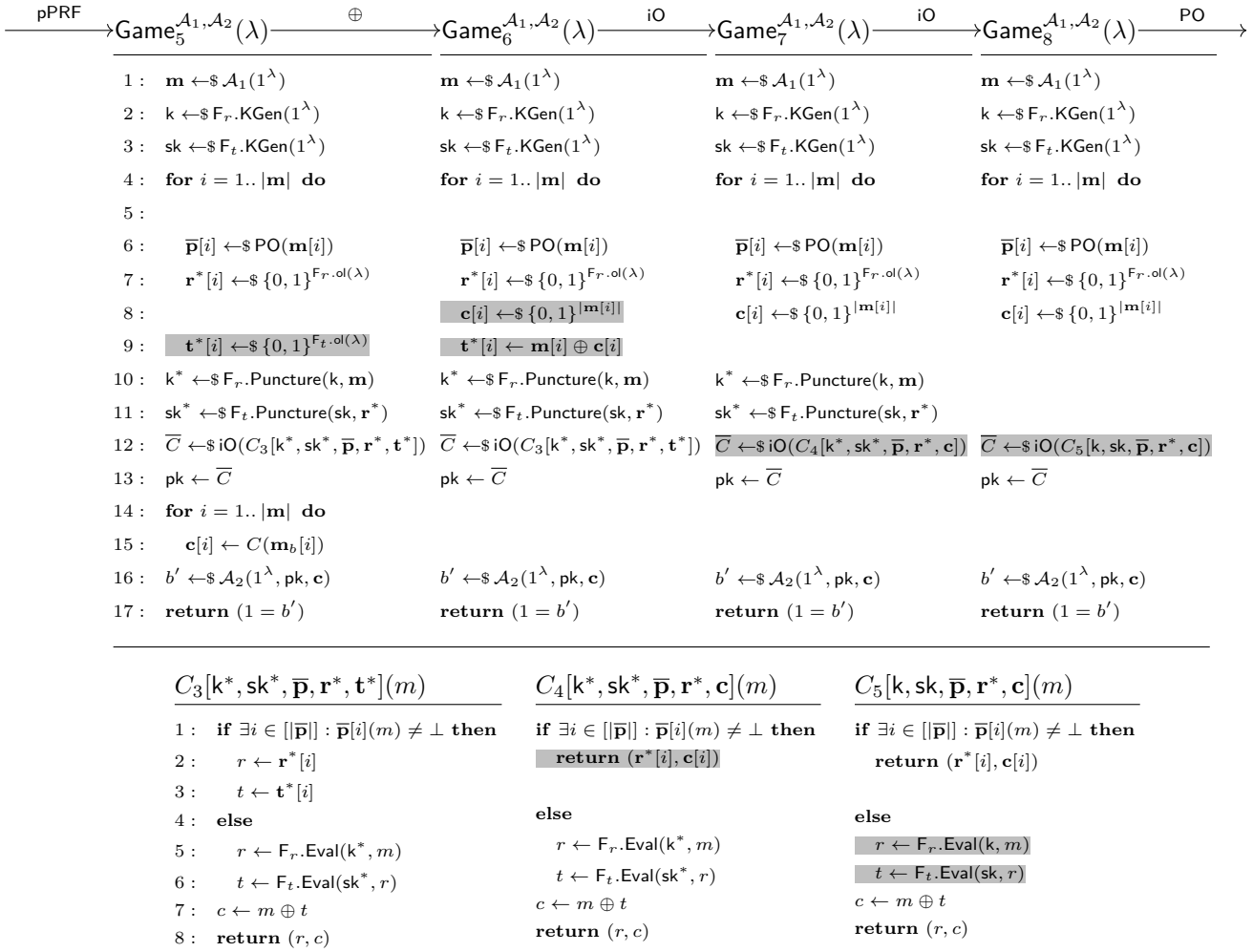


Figure 14.2: The remainder of games and circuits needed for the proof of Theorem 14.1. See also Figure 14.1.

iO

$\mathsf{Game}_1(\lambda)$**:** The first game is the original \$-IND game with hidden bit set to 0, that is, the message vector $\mathbf{m}$ as output by adversary $\mathcal{A}_1$ is encrypted.

$\mathsf{Game}_2(\lambda)$**:** Similar to before, except that a vector $\overline{\mathbf{p}}$ of point obfuscations for messages $\mathbf{m}$ is constructed. Furthermore, a vector $\mathbf{r}^*$ is constructed with $\mathbf{r}^*[i] \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, \mathbf{m}[i])$. The PRF key k is punctured on the messages in $\mathbf{m}$ resulting in punctured key k^*. The circuit underlying the public key is changed to $C_2[\mathsf{k}^*, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*]$ which first tests if the input matches any of the point functions, that is, any of the point functions output 1 on the input. If so, r is chosen as $\mathbf{r}^*[i]$ where i is the index of the matching point function (note that all point functions are for distinct points by requirement on the messages output by the adversary). Otherwise r is generated as before but with the punctured key k^*. Note that the circuits C_1 and C_2 compute identical functions.

pPRF

$\mathsf{Game}_3(\lambda)$**:** Identical to before except that now values $\mathbf{r}^*$ are chosen uniformly at random, that is,

$$\mathbf{r}^*[i] \leftarrow\$ \{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}.$$

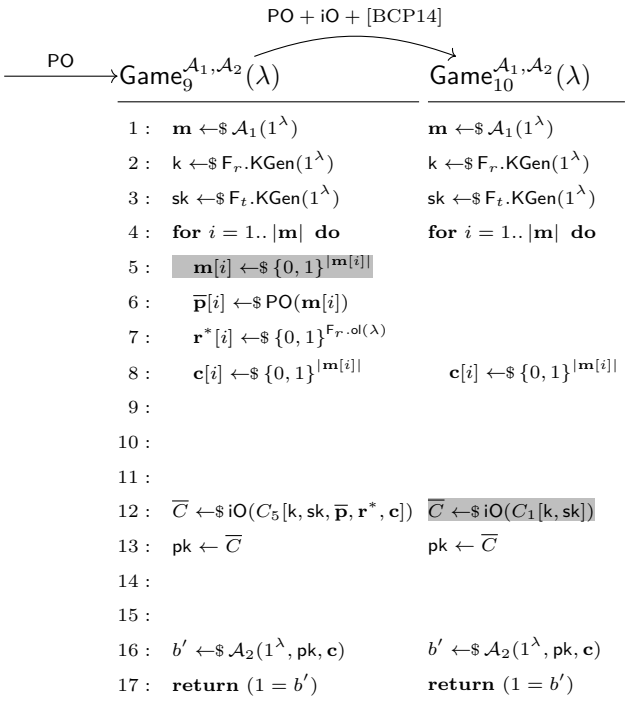


Figure 14.3: The remainder of games and circuits needed for the proof of Theorem 14.1. See also Figures 14.1 and 14.2.

iO

$\mathsf{Game}_4(\lambda)$**:** The game is as before but now with preparations to puncture also the second PRF F_t. That is, a vector $\mathbf{t}^*$ is introduced which stores the precomputed values $\mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}, \mathbf{r}^*[i])$. The secret key sk is punctured on the values in vector $\mathbf{r}^*$. A new circuit $C_3[\mathsf{k}^*, \mathsf{sk}^*, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{t}^*]$ is used which uses $\mathbf{t}^*$ if a value r matches any of the values in $\mathbf{r}^*$.

pPRF

$\oplus$

$\mathsf{Game}_5(\lambda)$**:** As before but now values in $\mathbf{t}^*$ are chosen uniformly at random.

$\mathsf{Game}_6(\lambda)$**:** As before, except that the ciphertexts in $\mathbf{c}$ are chosen uniformly at random and values $\mathbf{t}^*$ are adapted accordingly, that is, we construct $\mathbf{t}^*$ as

$$\mathbf{t}^*[i] \leftarrow \mathbf{m}_b[i] \oplus \mathbf{c}[i],$$

whereas before, $\mathbf{t}^*$ were chosen uniformly at random and $\mathbf{c}$ was computed as

$$\mathbf{c}[i] \leftarrow \mathbf{m}_b[i] \oplus \mathbf{t}^*[i].$$

iO

$\mathsf{Game}_7(\lambda)$**:** As before, but now circuit $C_4[\mathsf{k}^*, \mathsf{sk}^*, \mathbf{p}, \mathbf{r}^*, \mathbf{c}]$ is used as the circuit for the public-key. Circuit C_4 depends on ciphertext vector $\mathbf{c}$ and outputs a value $(\mathbf{r}^*[i], \mathbf{c}[i])$ directly in case the input matches a point function. Note that in case an input matches any of the point functions, then circuit C_4 returns already in line 2. Also note that the circuit is independent of vector $\mathbf{t}^*$.

iO

$\mathsf{Game}_8(\lambda)$**:** As before, except that Circuit $C_5[\mathsf{k}, \mathsf{sk}, \mathbf{p}, \mathbf{r}^*, \mathbf{c}]$ is used which uses the unpunctured keys k and sk instead of k^* and sk^*. Note that this does not change the functionality of the circuit, that is, circuits C_4 and C_5 compute the same function.

PO

$\mathsf{Game}_9(\lambda)$**:** With this game hop we make a switch in message distributions and overwrite all messages in vector $\mathbf{m}$ (line 5) with uniformly random values, that is,

$$\mathbf{m}[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{m}|}.$$

PO + iO + [BCP14]

$\mathsf{Game}_{10}(\lambda)$**:** We have reached the target game, that is, We replace the obfuscated circuit with an obfuscation of the original circuit C_1.

Game_{10} is the \$-IND target game where adversary $\mathcal{A}_2$ receives random strings instead of ciphertexts. We can, thus, write the advantage of adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ as

$$\begin{aligned}
\mathsf{Adv}^{\text{\$-ind}}_{\mathcal{A},\mathsf{D\text{-}PKE}}(\lambda) &= \Pr\left[\text{\$-IND}^{\mathcal{A}}_{\mathsf{D\text{-}PKE}}(\lambda) = 1 \;\middle|\; b = 0\right] + \Pr\left[\text{\$-IND}^{\mathcal{A}}_{\mathsf{D\text{-}PKE}}(\lambda) = 1 \;\middle|\; b = 1\right] - 1 \\
&= \Pr\left[\mathsf{Game}_1^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{10}^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] \\
&\leq \sum_{i=1}^{9} \left|\Pr\left[\mathsf{Game}_i^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_{i+1}^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right]\right|
\end{aligned}$$

To complete the proof of Theorem 14.1 we will show that the distribution induced by any two consecutive games is computationally indistinguishable. We discuss the steps in turn.

$\mathsf{Game}_1(1^\lambda)$ to $\mathsf{Game}_2(1^\lambda)$. The difference between games Game_1 and Game_2 is that adversary $\mathcal{A}_2$ is given an obfuscation of a different but functionally equivalent circuit as public-key pk. In the first game the circuit is as in the construction denoted by $C_1[\mathsf{k}, \mathsf{sk}]$ in Figure 14.1 whereas in Game_2 the circuit is $C_2[\mathsf{k}^*, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*]$ where k^* is a punctured PRF key (punctured on all messages $\mathbf{m}$ as output by $\mathcal{A}_1$), vector $\overline{\mathbf{p}}$ contains obfuscations of point functions for all messages in $\mathbf{m}$ and vector $\mathbf{r}^*$ contains values computed as

$$\mathbf{r}^*[i] \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, \mathbf{m}[i]).$$

Note that the two circuits C_1 and C_2 compute the same function and, thus, we can bound the distinguishing advantage of adversary $(\mathcal{A}_1, \mathcal{A}_2)$ by the security of the indistinguishability obfuscator iO. A formal reduction is analogous to the first game hop in the proof of Theorem 11.1. Consequently, we can bound the distinguishing probability as

$$\left|\Pr\left[\mathsf{Game}_2^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_1^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\text{io}}_{\mathsf{D},\mathsf{Sam}}(\lambda),$$

where $(\mathsf{D}, \mathsf{Sam})$ is the induced adversary against the indistinguishability obfuscator.

$\mathsf{Game}_2(1^\lambda)$ to $\mathsf{Game}_3(1^\lambda)$. From Game_2 to Game_3 the computation of entries in vector $\mathbf{r}^*$ is changed, that is, the entries are chosen uniformly at random in Game_3:

$$\mathbf{r}^*[i] \leftarrow \{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}.$$

It follows that the only difference between the two games is that the images of pseudorandom function F_r on punctured points are "honestly chosen" in Game_2 and chosen uniformly at random in Game_3. This allows us to reduce the distinguishing difference between the two games to the security of the puncturable pseudorandom function F_r. A formal reduction is analogous to the second game hop in the proof of Theorem 11.1. It follows that

$$\left|\Pr\left[\mathsf{Game}_3^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]-\Pr\left[\mathsf{Game}_2^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F}_r,\mathcal{B}_1,\mathcal{B}_2}(\lambda),$$

where $(\mathcal{B}_1,\mathcal{B}_2)$ is the induced adversary against the puncturable PRF F_r.

$\mathsf{Game}_3(1^\lambda)$ to $\mathsf{Game}_4(1^\lambda)$. In Game_4 we make the preparation for puncturing secret key sk. The analysis is similar to the step from Game_1 to Game_2 but noting that the two circuits C_2 and C_3 are only functionally equivalent with overwhelming probability. For this note, that after puncturing secret key sk^* on values $\mathbf{r}^*$ we can no longer compute $\mathsf{F}_t.\mathsf{Eval}(\mathsf{sk},\mathbf{r}^*[i])$ for any of the entries in vector $\mathbf{r}^*$. In order for this situation not to occur, we require that no message $m \in \{0,1\}^{\mathsf{F}_r.\mathsf{il}(\lambda)}$ can be mapped to any of the values in $\mathbf{r}^*$ or in other words, we require that for all $i=1,\ldots,\mathsf{q}(\lambda)$

$$|\{m : \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}^*,m)=\mathbf{r}^*[i]\}|=0.$$

PRF F_r was chosen to be length doubling, and hence a random value in $\{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}$ is in the image of $\mathsf{F}_r.\mathsf{Eval}(\mathsf{k},\cdot)$ for a random key k with probability at most $2^{\mathsf{F}_r.\mathsf{il}(\lambda)}\cdot 2^{-2\mathsf{F}_r.\mathsf{il}(\lambda)}$. It follows that

$$\Pr[\exists i : |\{m : \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}^*,m)=\mathbf{r}^*[i]\}|>0] \leq \frac{\mathsf{q}(\lambda)}{2^{\mathsf{F}_r.\mathsf{il}(\lambda)}}$$

and we can, thus, bound the distinguishing probability with the security of the indistinguishability obfuscator as

$$\left|\Pr\left[\mathsf{Game}_4^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]-\Pr\left[\mathsf{Game}_3^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{D},\mathsf{Sam}}(\lambda)+\frac{\mathsf{q}(\lambda)}{2^{\mathsf{F}_r.\mathsf{il}(\lambda)}}$$

where $(\mathsf{D},\mathsf{Sam})$ is the induced iO-adversary.

$\mathsf{Game}_4(1^\lambda)$ to $\mathsf{Game}_5(1^\lambda)$. In Game_5 we replace the images under punctured points for sk^* with uniformly random values. That is, the values in $\mathbf{t}^*$ are now chosen uniformly at random. The analysis is analogous to the step from Game_2 to Game_3 and thus:

$$\left|\Pr\left[\mathsf{Game}_5^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]-\Pr\left[\mathsf{Game}_4^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathrm{pprf}}_{\mathsf{F}_r,\mathcal{B}_1,\mathcal{B}_2}(\lambda).$$

$\mathsf{Game}_5(1^\lambda)$ to $\mathsf{Game}_6(1^\lambda)$. In Game_6 the order in which vectors $\mathbf{t}^*$ and $\mathbf{c}$ are computed is reversed. That is, in Game_5 the entries in vector $\mathbf{t}^*$ are chosen uniformly at random an then $\mathbf{c}$ is computed as

$$\mathbf{c}[i] \leftarrow \mathbf{m}[i] \oplus \mathbf{t}^*[i].$$

In Game_6, on the other hand, we now choose values in $\mathbf{c}$ uniformly at random and then adapt values in $\mathbf{t}^*$ accordingly, that is, we set

$$\mathbf{t}^*[i] \leftarrow \mathbf{m}[i] \oplus \mathbf{c}[i].$$

Note that the distribution of values in vectors $\mathbf{t}^*$ and $\mathbf{c}$ are identical in both games and we, thus, have that

$$\Pr[\mathsf{Game}_5(\lambda) = 1] = \Pr[\mathsf{Game}_6(\lambda) = 1].$$

$\mathsf{Game}_6(1^\lambda)$ to $\mathsf{Game}_7(1^\lambda)$. In Game_7 the public-key is chosen as an obfuscation of circuit $C_4[\mathsf{k}^*, \mathsf{sk}^*, \mathbf{p}, \mathbf{r}^*, \mathbf{c}]$ which now has vector $\mathbf{c}$ hard-coded and directly returns the tuple $(\mathbf{r}^*[i], \mathbf{c}[i])$ in case the input matches the i-th point function (line 2). Noting that the two circuits C_3 and C_4 are functionally equivalent we can, thus, bound the distinguishing probability by the security of the indistinguishability obfuscator. The formal treatment is analogous to the step from Game_1 to Game_2.

$$\left|\Pr\left[\mathsf{Game}_7^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_6^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{D},\mathsf{Sam}}(\lambda)$$

$\mathsf{Game}_7(1^\lambda)$ to $\mathsf{Game}_8(1^\lambda)$. From Game_7 to Game_8 we are switching from circuit C_4 to C_5 where the only difference is that circuit C_5 uses the unpunctured keys k and sk instead of the punctured keys k^* and sk^*. Noting that the circuits are functionally equivalent if for key k no message m is mapped by $\mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, m)$ to a value in vector $\mathbf{r}^*$ (see also game hop from Game_3 to Game_4) we can bound the distinguishing probability by the security of the indistinguishability obfuscator. As argued before, the probability that a random value in $\{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda)}$ is in the image of $\mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, \cdot)$ for a random key k is at most $2^{\mathsf{F}_r.\mathsf{il}(\lambda)} \cdot 2^{-2\mathsf{F}_r.\mathsf{il}(\lambda)}$. It follows, that

$$\left|\Pr\left[\mathsf{Game}_8^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right] - \Pr\left[\mathsf{Game}_7^{\mathcal{A}_1,\mathcal{A}_2}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathrm{io}}_{\mathsf{D},\mathsf{Sam}}(\lambda) + \frac{\mathsf{q}(\lambda)}{2^{\mathsf{F}_r.\mathsf{il}(\lambda)}}$$

where $(\mathsf{D}, \mathsf{Sam})$ is the induced iO-adversary.

$\mathsf{Game}_8(1^\lambda)$ to $\mathsf{Game}_9(1^\lambda)$. From Game_8 to Game_9 we change the message vector $\mathbf{m}$, that is, we overwrite the entries with uniformly random strings of the same length.

We can bound the distinguishing probability by the security of the point-function obfuscation scheme. For this, we construct adversary $(\mathsf{Sam}, \mathsf{D})$ against the security of scheme PO as follows. Point sampler Sam runs $\mathcal{A}_1$ to receive a message vector $\mathbf{m}$ which it outputs. Adversary D receives as input a vector $\overline{\mathbf{p}}$ which is either a sequence of point functions for messages in $\mathbf{m}$ or for uniformly random values of the same length. It chooses vectors $\mathbf{r}^*$ and $\mathbf{c}$ as in game Game_8 as uniformly random values in $\{0,1\}^{\mathsf{F}_t.\mathsf{ol}(\lambda)}$ and $\{0,1\}^{\mathbf{m}[i]}$, respectively. It then generates two keys $\mathsf{k} \leftarrow\!\!\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$ and $\mathsf{sk} \leftarrow\!\!\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$. It punctures sk on vector $\mathbf{r}^*$ and constructs circuit $C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}]$ and computes $\mathsf{pk} \leftarrow\!\!\$\, \mathsf{iO}(C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}])$. Finally, distinguisher D runs adversary $\mathcal{A}_2$ on input the security parameter, public-key pk and ciphertext vector $\mathbf{c}$. It outputs whatever $\mathcal{A}_2$ outputs.

If $\overline{\mathbf{p}}$ contains obfuscations of points in $\mathbf{m}$ then adversary $(\mathsf{Sam}, \mathsf{D})$ perfectly simulates Game_8 and otherwise, if $\overline{\mathbf{p}}$ contains obfuscations of uniformly random points they perfectly simulate Game_9. We, thus, have that

$$\left|\Pr\left[\mathsf{Game}_9^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]-\Pr\left[\mathsf{Game}_8^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathrm{po}}_{\mathsf{PO},\mathsf{Sam},\mathsf{D}}(\lambda).$$

Remark. At this point we have changed the message vector as sampled by adversary $\mathcal{A}_1$ to uniformly random. Further note that in this game hop we did not require auxiliary information for the point-function obfuscator. Consider the weaker IND-security notion where adversary $\mathcal{A}_1$ outputs two message vectors and it is required that $\mathcal{A}_2$ cannot distinguish between seeing encryptions of $\mathbf{m}_0$ from $\mathbf{m}_1$. Having changed the message distribution to uniform messages we could next change it to $\mathbf{m}_1$ with an identical game hop. Subsequently we could go back step by step to Game_1. This shows that the construction is IND-secure assuming indistinguishability obfuscation, puncturable PRFs and point obfuscation AIPO$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}} \cap \mathcal{S}^{\mathsf{aux}}_{\mathsf{po}}]$ (i.e., point obfuscation that does not need to stay secure in the presence of auxiliary-information).

In the following we show the stronger result, i.e., we show \$-IND-security which requires AIPO$[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$, that is, we need to be able to leak a hardcore bit of one of the messages.

$\mathsf{Game}_9(1^\lambda)$ to $\mathsf{Game}_{10}(1^\lambda)$. In the final game Game_{10} we replace the obfuscation of circuit C_5 with an obfuscation of circuit C_1. Note that C_1 and C_5 differ on only polynomially many inputs, that is, they differ on exactly $\mathsf{q}(\lambda)$ many inputs. We would like to argue that indistinguishability obfuscations of the two circuits are indistinguishable using Theorem 5.13 by Boyle, Chung, and Pass (page 92) who show that for circuits that differ on only polynomially many inputs any general purpose indistinguishability obfuscator is also a differing-inputs obfuscator. To apply Theorem 5.13, we need to argue that a sampler, which runs the steps of Game_{10} up to line 12, sets $\mathsf{aux} \leftarrow \mathbf{c}$ and then outputs $(C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}], C_1[\mathsf{k}, \mathsf{sk}], \mathsf{aux})$ is a differing-inputs sampler.

To this end, let Ext be an extractor that on input $(C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}], C_1[\mathsf{k}, \mathsf{sk}], \mathbf{c})$ as sampled by Sam outputs a value τ such that $C_1[\mathsf{k}, \mathsf{sk}](\tau) \neq C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}](\tau)$. We will bound the success probability of Ext by the security of the point obfuscator PO. For this we construct an adversary $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ against point obfuscator PO as follows: On input the security parameter, sampler $\mathsf{Sam}_{\mathsf{PO}}$ runs adversary $\mathcal{A}_1$ to obtain a vector $\mathbf{m}$. It subsequently overwrites every entry in $\mathbf{m}$ with a uniformly random string, that is, it sets

$$\mathbf{m}[i] \leftarrow\!\!\$\ \{0,1\}^{|\mathbf{m}[i]|}$$

as in line 5. It then chooses a random string $s \in \{0,1\}^{\mathsf{F}_r.\mathsf{il}(\lambda)}$ and a random index $j \in [\mathsf{q}]$. It computes $b \leftarrow \langle \mathbf{m}[j], s\rangle$, sets $\mathsf{aux} \leftarrow (b, j, s)$, then stops and outputs the resulting vector $\mathbf{m}$ and auxiliary information aux.

Distinguisher D_{PO} gets as input a vector of point obfuscations $\overline{\mathbf{p}}$ which either contain point obfuscations for message vector $\mathbf{m}$ or for uniformly random messages. Additionally, it gets auxiliary information $(b, j, s) \leftarrow \mathsf{aux}$. Distinguisher D_{PO} then chooses vectors $\mathbf{r}^*$ and $\mathbf{c}$ with uniformly random entries of lengths as in Game_{10}. It then constructs two PRF keys $\mathsf{k} \leftarrow\!\!\$\ \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$ and $\mathsf{sk} \leftarrow\!\!\$\ \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$. Finally, it constructs circuits $C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}]$ and $C_1[\mathsf{k}, \mathsf{sk}]$ and runs extractor

Ext on input $(C_5[\mathsf{k}, \mathsf{sk}, \overline{\mathbf{p}}, \mathbf{r}^*, \mathbf{c}], C_1[\mathsf{k}, \mathsf{sk}], \mathbf{c})$ to obtain a value τ. If value $\tau \neq \bot$, that is, if extractor Ext succeeded and if, furthermore, $\overline{\mathbf{p}}[j](\tau) = 1$ then distinguisher D_{PO} outputs 1 if $b = \langle \tau, s \rangle$ and 0 otherwise. In case $\overline{\mathbf{p}}[j](\tau) = 0$ distinguisher D_{PO} always outputs 0. Finally, in case $\tau = \bot$ distinguisher D_{PO} outputs 1 with probability $\frac{1}{2\mathsf{q}}$ and 0 otherwise. With a similar analysis as the last game hop in the proof of Theorem 11.1 we can show that the distinguishing advantage of adversary $(\mathsf{Sam}_{\mathsf{PO}}, \mathsf{D}_{\mathsf{PO}})$ is $\frac{1}{2\mathsf{q}} \cdot \epsilon$ where ϵ is the advantage of extractor Ext. This shows that Sam is a differing-inputs sampler and hence

$$\left|\Pr\left[\mathsf{Game}_{10}^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)\right] - \Pr\left[\mathsf{Game}_{9}^{\mathcal{A}_1,\mathcal{A}_2}(\lambda)\right]\right| \leq \mathsf{Adv}^{\mathsf{io}}_{\mathsf{iO},\mathsf{Sam},\mathsf{Dist}}(\lambda) + [\text{BCP14}].$$

This concludes the last game hop (indeed, the last game hop of this thesis) and the proof. □

14.4 The Role of Padding

As with our UCE construction our D-PKE construction only achieves q-bounded security due to the use of padding within the construction. In abstract terms, in order to show \$-IND security for a deterministic public-key encryption scheme $\mathsf{D\text{-}PKE}$ scheme we need to argue that the distributions

$$(\mathbf{c}, \mathsf{pk}) \quad \text{and} \quad (\mathbf{r}, \mathsf{pk})$$

are computationally indistinguishable, where pk is chosen as a uniformly random public key according to the scheme's key generation, $\mathbf{c}$ is a vector of honestly created ciphertexts for high min-entropy and key-independent messages and $\mathbf{r}$ is a vector of uniformly random strings. Our construction (see Construction 14.1 on page 284) chooses public keys as obfuscations of a two-round Feistel network with two puncturable pseudorandom functions F_r and F_t both with a hard-coded key. If we denote by $C[\mathsf{k}, \mathsf{sk}]$ the circuit underlying Construction 14.1 then to argue \$-IND-security we need to show that the distributions

$$(\mathbf{c}, \mathsf{iO}(C[\mathsf{k}, \mathsf{sk}])) \quad \text{and} \quad (\mathbf{r}, \mathsf{iO}(C[\mathsf{k}, \mathsf{sk}]))$$

are computationally indistinguishable.

What we have shown instead is that with sufficient padding and when restricting the length of vectors $\mathbf{c}$ and $\mathbf{r}$ to hold at most q entries, that then these two distributions are computationally indistinguishable. With the superfluous padding assumption for indistinguishability obfuscation (see Chapter 13) we can lift our result from q-query to fully secure. On the other hand, as we have seen the existence of restricted SuP assumptions is unclear.

With our construction of D-PKE here we have, however, gained a significant advantage over our constructions of UCEs in Chapter 11 in regard to the application of SuP assumptions. Note that all known counter examples to SuP for circuits require the auxiliary information to contain an obfuscation. Here, we are now in a situation where we can restrict samplers in such a way as to ensure that aux cannot contain such an obfuscation unless there exist strong obfuscators that generate program descriptions that are computationally indistinguishable from uniformly random strings, which seems unlikely.[2] Let us consider the following SuP-sampler Sam that is parameterized

[2] Consider obfuscations of the all-zero circuit $\overline{C}_0$ and the all-one circuit $\overline{C}_1$. While it might be the case that either

by an adversary $\mathcal{A}$ that generates a vector of distinct and high min-entropy messages as required by the \$-IND-security notion for deterministic public-key encryption:

$\mathsf{Sam}_{\mathcal{A}}(1^\lambda, b)$

$\mathsf{k} \leftarrow\!\$\, \mathsf{F}_r.\mathsf{KGen}(1^\lambda)$
$\mathsf{sk} \leftarrow\!\$\, \mathsf{F}_t.\mathsf{KGen}(1^\lambda)$
$\mathbf{m} \leftarrow\!\$\, \mathcal{A}(1^\lambda)$
for $i = 1, \ldots, |\mathbf{m}|$ **do**
  $r \leftarrow \mathsf{F}_r.\mathsf{Eval}(\mathsf{k}, \mathbf{m}[i])$
  $\mathsf{aux}_0[i] \leftarrow (r, \mathbf{m}[i] \oplus \mathsf{F}_t.\mathsf{Eval}(\mathsf{sk}, r))$
  $\mathsf{aux}_1[i] \leftarrow\!\$\, \{0,1\}^{\mathsf{F}_r.\mathsf{ol}(\lambda) \times \mathsf{F}_t.\mathsf{ol}(\lambda)}$
return $(\mathsf{aux}_b, C[\mathsf{k}, \mathsf{sk}])$

First note that our above proof shows that for any $\mathcal{A}$ there is a padding such that no distinguisher has non-negligible advantage in the SuP game (cf. Definition 13.1). Further note that the above sampler is fine-tuned to our application and, thus, very restricted. On the other hand, what becomes apparent is that sampler Sam samples auxiliary inputs aux_0 as pseudorandom strings and aux_1 as uniformly random strings and the hope is that they, thus, cannot contain obfuscations.

This motivates the following class of SuP samplers that we denote by $\mathcal{S}^{\mathsf{pr\text{-}aux}}$: samplers that produce pseudorandom auxiliary information. More formally, an SuP sampler Sam is in class $\mathcal{S}^{\mathsf{pr\text{-}aux}}$ if no PPT distinguisher D can distinguish between getting as input values $(\mathsf{aux}_0, \mathsf{aux}_1)$ or uniformly random strings of the same length. That is, $\mathsf{Sam} \in \mathcal{S}^{\mathsf{pr\text{-}aux}}$ if for all PPT distinguishers D it holds that

$$\left| \Pr_{(\mathsf{aux}, C) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda, 0)} \left[\mathsf{D}(1^\lambda, \mathsf{aux}) = 1 \right] - \Pr_{\substack{(\mathsf{aux}, C) \leftarrow\!\$\, \mathsf{Sam}(1^\lambda) \\ (r) \leftarrow\!\$\, \{0,1\}^{|\mathsf{aux}|}}} \left[\mathsf{D}(1^\lambda, r) = 1 \right] \right| \leq \mathsf{negl}(\lambda)\,.$$

Combined with the previously suggested restriction of statistically unpredictable SuP samplers and potentially even PRF samplers (note that circuit $C[\mathsf{k}, \mathsf{sk}]$ is a PRF) we can, thus, state the following corollary.

Corollary 14.3. *If indistinguishability obfuscation exists and if composable AIPO for statistically unpredictable distributions (composable* $\mathsf{AIPO}[\mathcal{S}^{\mathrm{sup}}_{\mathsf{po}}]$*) exist, if the restricted superfluous padding assumption SuP*$[\mathcal{S}^{\mathrm{sup}} \cap \mathcal{S}^{\mathsf{pr\text{-}aux}} \cap \mathcal{S}^{\mathsf{prf}}]$ *holds for a secure indistinguishability obfuscation scheme, then Construction 14.1 is* \$-IND *secure.*

is indistinguishable from a random string it cannot be the case that both are as we can easily distinguish between $\overline{C}_0$ and $\overline{C}_1$ by simply evaluating the program.

Conclusion

> "Why, sometimes I've believed as many as six impossible things before breakfast."
>
> Lewis Carroll, Alice in Wonderland

Many fundamental questions of theoretical computer science in general and cryptography in particular revolve around the question of what is possible and what is not. With the research presented in this thesis we took a closer look at the beautiful abstraction of hash functions known as random oracles which allows us to construct elegant, efficient, and often simple schemes; yet ever since the seminal result of Canetti et al. [CGH98] we know that no efficiently computable function can *behave like a random oracle* and, thus, the security guarantees we have for random oracle schemes are unclear. In Part II we have seen various examples where random oracle proofs may fail. On the other hand, taking a purely practical perspective, there is good evidence that the random oracle methodology used as a heuristic for the construction of secure cryptographic schemes works well: practical schemes that have not been designed with the sole purpose of bringing to light inconsistencies seem to work. However, the best argument in favor of using a random oracle scheme is simply an additional security analysis which does not hinge upon the power of random oracles. Strong standard-model assumptions, such as UCEs, provide a promising alternative to the use of random oracles and may well lead to standard-model proofs for many random oracle schemes. UCEs were introduced with the idea of being cryptanalyzable and, indeed, we have shown that the original UCE notions cannot exist if indistinguishability obfuscation does. From this negative result, we were able to extract new and powerful UCE notions for which we do not know any negative results. Instead, we were even able to provide candidate constructions under strong, but reasonable obfuscation-based assumptions.

Our UCE construction should, of course, not be considered practical in any sense of the word, but instead should be understood as validation of the assumption that there exist hash functions that are UCE-secure. The random oracle methodology suggests to instantiate the random oracle with good cryptographic hash functions such as SHA-256, SHA-3, or HMAC. Similarly, in place of UCEs for practical implementations, it makes sense to use good (keyed) cryptographic hash functions such as HMAC or variants of SHA-3. The fundamental difference is that while we know that HMAC does not behave like a random oracle it is plausible that it is UCE-secure.

I hope that with this work I have contributed a small part to the puzzle that is the random oracle model and I would like to sincerely thank you, the reader, and researcher for accompanying me on this journey. I hope that you enjoyed it and maybe even got one or two inspirations to further our understanding of UCEs, random oracles, obfuscation or something completely different.

Appendix A

Cryptographic and Complexity Theoretic Background

In this appendix we provide additional definitions and concepts that we touch upon in this thesis.

A.1 Cryptography

Definition A.1 (Signature scheme)**.** *A* signature scheme *is a tuple of three efficient algorithms* $\mathsf{S} = (\mathsf{S.KGen}, \mathsf{S.Sign}, \mathsf{S.Vf})$ *where the probabilistic key-generation algorithm* $\mathsf{S.KGen}$ *on input the security parameter outputs a pair of keys* $(\mathsf{sk}, \mathsf{vk})$*, the probabilistic signing algorithm* $\mathsf{S.Sign}$ *on input the signing key* sk *and a message* m *outputs a signature* σ *and the deterministic verification algorithm* $\mathsf{S.Vf}$ *on input a verification key* vk *a message* m *and signature* σ *outputs a bit* b *indicating if the signature is valid in which case* $b = 1$*. We require correctness that is for all security parameters* $\lambda \in \mathbb{N}$*, for all messages* $m \in \{0,1\}^*$ *for all* $(\mathsf{sk}, \mathsf{vk}) \in \mathrm{Supp}\big(\mathsf{S.KGen}(1^\lambda)\big)$ *it holds that*

$$\Pr[\mathsf{S.Vf}(\mathsf{vk}, m, \mathsf{S.Sign}(\mathsf{sk}, m)) = 1] = 1.$$

We say that a signature scheme is existentially unforgeable under chosen message attacks *if for all* PPT *adversaries* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\mathsf{Adv}^{\mathrm{eufcma}}_{\mathsf{S},\mathcal{A}}(\lambda) := \Pr\left[\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{S}}(\lambda)\right] \leq \mathsf{negl}(\lambda)\,,$$

where game $\mathsf{EUF\text{-}CMA}$ *is defined as follows:*

$\mathsf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{S}}(\lambda)$

$M \leftarrow \{\}$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{S.KGen}(1^\lambda)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}[\mathsf{sk}]}(\mathsf{vk})$
if $\mathsf{S.Vf}(\mathsf{vk}, m^*, \sigma^*) = 1 \wedge m^* \notin M$ **then**
  return 1
return 0

$\textsc{Sign}[\mathsf{sk}](m)$

$M \leftarrow M \cup \{m\}$
$\sigma \leftarrow\!\!\$\ \mathsf{Sign}(\mathsf{sk}, m)$
return σ

A.2 Complexity Theory

In the following we present basic definition from complexity theory such as the complexity classes P, NP, and $\mathsf{co\text{-}NP}$, the polynomial hierarchy and what it means for a language to be NP-complete. For a good introduction to complexity theory we refer to the books of Arora and Barak [AB09] and Goldreich [Gol08].

Definition A.2. *A language $\mathcal{L} \subseteq \{0,1\}^*$ is a set of strings. Language $\overline{\mathcal{L}} := \{0,1\}^* \setminus \mathcal{L}$ is called the complement of language $\mathcal{L}$. We say that a machine* decides *a language $\mathcal{L} \subseteq \{0,1\}^*$ if it computes its characteristic function $\chi_{\mathcal{L}} : \{0,1\}^* \to \{0,1\}$ defined as:*

$$\chi_{\mathcal{L}}(x) = 1 \iff x \in \mathcal{L}.$$

Complexity Classes

We can now define the complexity class P which is the set of languages that can be decided by a deterministic polynomial-time Turing machine.

Definition A.3 (The class P)**.** *A language $\mathcal{L} \subseteq \{0,1\}^*$ is in P (written as $\mathcal{L} \in \mathsf{P}$) if there exists a deterministic polynomial-time Turing machine M which decides $\mathcal{L}$.*

The class NP characterizes languages where membership can be efficiently verified, that is, for any x in the language there exists a polynomial size witness $w \in \{0,1\}^{\mathsf{poly}(\lambda)|x|]}$ such that membership in the language can be efficiently verified given (x, w).

Definition A.4 (The class NP)**.** *A language $\mathcal{L} \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $\mathsf{poly} : \mathbb{N} \to \mathbb{N}$ and a deterministic polynomial-time Turing machine M such that for any $x \in \{0,1\}^*$*

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x,w) = 1.$$

We call M the verifier for language $\mathcal{L}$ *and if $x \in \mathcal{L}$ and $w \in \{0,1\}^{\mathsf{poly}(|x|)}$ such that $M(x,w) = 1$ then we call w a* witness *for x (relative to language $\mathcal{L}$ and machine M).*

Given a complexity class C we define its complement class $\mathsf{co\text{-}C}$ as

$$\mathsf{co\text{-}C} := \left\{ \mathcal{L} : \overline{\mathcal{L}} \in \mathsf{C} \right\}.$$

This allows us to define the complexity class $\mathsf{co\text{-}NP}$:

Definition A.5 (The class $\mathsf{co\text{-}NP}$)**.** *A language $\mathcal{L} \subseteq \{0,1\}^*$ is in $\mathsf{co\text{-}NP}$ if its complement $\overline{\mathcal{L}}$ is in NP. That is,*

$$\mathsf{co\text{-}NP} := \left\{ \mathcal{L} : \overline{\mathcal{L}} \in \mathsf{NP} \right\}.$$

We can generalize the above definitions which leads to the definition of the *polynomial hierarchy* denoted by PH.

Definition A.6 (The class PH)**.** *We set* $\Sigma_0^p = \Pi_0^p = \mathsf{P}$*. For* $i \geq 1$*, a language* $\mathcal{L}$ *is in* Σ_i^p *if there exists a deterministic polynomial time Turing machine* M *and a polynomial* $\mathsf{poly} : \mathbb{N} \to \mathbb{N}$ *such that for any* $x \in \{0,1\}^*$

$$x \in \mathcal{L} \iff \exists w_1 \in \{0,1\}^{\mathsf{poly}(|x|)} \forall w_2 \in \{0,1\}^{\mathsf{poly}(|x|)} \cdots \mathcal{Q}_i w_i \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } M(x, w_1, \ldots, w_i) = 1.$$

Here, $\mathcal{Q}_i$ *denotes the quantifier* $\forall$ *or* $\exists$ *depending on whether* i *is even or odd.*
For $i \geq 1$*, we define* $\Pi_i^p = \mathsf{co}\text{-}\Sigma_i^p = \left\{ \mathcal{L} : \overline{\mathcal{L}} \in \Sigma_i^p \right\}$.

The polynomial hierarchy *is defined as the set* $\mathsf{PH} := \cup_i \Sigma_i^p$.

Note that we could define Π_i^p similarly to Σ_i^p by just switching the orders of quantifiers (i.e., start with a $\forall$ quantifier). We thus have a generalization of the previously defined classes:

- $\Sigma_0^p = \Pi_0^p = \mathsf{P}$
- $\Sigma_1^p = \mathsf{NP}$
- $\Pi_1^p = \mathsf{co}\text{-}\mathsf{NP}$

We say that the polynomial hierarchy *collapses* to the i-th level if $\mathsf{PH} = \Sigma_i^p$. As shown by Stockmeyer [Sto76] this occurs, for example, if for any $i \geq 1$ one can show that $\Sigma_i^p = \Pi_i^p$. We note that it is generally assumed that the polynomial hierarchy does not collapse.

Boolean Circuits

As for obfuscation we mostly consider boolean circuits we here define two circuit complexity classes that we encounter throughout this thesis: P/poly and NC.[1] The complexity class P/poly is the circuit analogue of P in that it captures all languages that can be decided by a sequence of polynomial-sized, deterministic circuits (in contrast to a polynomial-time Turing machine). Note that we only consider circuits with with AND, OR and NOT gates.

Definition A.7 (The class P/poly)**.** *A language* $\mathcal{L}$ *is in* P/poly *if there exists a sequence of polynomial sized circuits* $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ *that decide* $\mathcal{L}$*.*

The complexity class NC^d is a more restricted variant of P/poly. Here we consider circuits only of a specific *depth*, where by "depth" we denote the length of the longest directed path from an input node to the output node.

Definition A.8 (The class NC^d)**.** *For every* $d \in \mathbb{N}$*, we say that a language* $\mathcal{L}$ *is in* NC^d *if there exists a sequence of polynomial sized circuits* $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ *that decide* $\mathcal{L}$ *and where* C_λ *has depth* $\mathcal{O}\left(\log^d \lambda\right)$*. The class* NC *is defined as* $\mathsf{NC} := \cup_{i \geq 1} \mathsf{NC}^i$.

[1] Many general purpose obfuscator candidate constructions essentially first construct an obfuscator for the class NC^1 to then bootstrap from there.

Complete Problems

We next briefly recall what it means for a language $\mathcal{L} \in \{0,1\}^*$ to be complete for a complexity class C. Intuitively a language is complete for a class C if it is as hard to decide as any other language in class C and if it is in class C itself. We here consider only the case for NP.

Definition A.9. *Let $\mathcal{L} \subseteq \{0,1\}^*$ and $\mathcal{L}' \subseteq \{0,1\}^*$ be two languages. We say that $\mathcal{L}$ is polynomial-time Karp reducible to $\mathcal{L}'$ (denoted to by $\mathcal{L} \leq_p \mathcal{L}'$) if there exists a deterministic polynomial time Turing machine M such that for all $x \in \{0,1\}^*$:*

$$x \in \mathcal{L} \iff M(x) \in \mathcal{L}'.$$

We say that $\mathcal{L}'$ is NP-hard *if $\mathcal{L} \leq_p \mathcal{L}'$ for every $\mathcal{L} \in \mathsf{NP}$. We say that $\mathcal{L}'$ is* NP-complete *if $\mathcal{L}'$ is* NP-hard *and $\mathcal{L}' \in \mathsf{NP}$.*

A.3 Assumptions

Definition A.10 (Decisional Diffie–Hellman problem (DDH))**.** *Let $\mathcal{G}$ be a group generator that on input the security parameter 1^λ outputs a tuple $(\mathbb{G}, q, g)$ where $\mathbb{G}$ is a group description with order q and generator g. We say that the* DDH problem *is hard relative to $\mathcal{G}$ if for any* PPT *distinguisher* D *there exists a negligible function* negl *such that*

$$|\Pr[\mathsf{D}(\mathbb{G}, q, g, g^x, g^y, g^z) = 1] - \Pr[\mathsf{D}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = 1]| \leq \mathsf{negl}(\lambda)\,.$$

The probabilities are over the the coins of the group generator $\mathcal{G}$ that outputs $(\mathbb{G}, q, g)$, and then subsequently the values $x, y, z \in \mathbb{Z}_q$ are chosen uniformly at random.

Bibliography

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany. 32

[Abi15] Abigail. Just another perl hacker. `http://www.cpan.org/misc/japh` – Last visited 2015/05/25, 2015. 67, 69

[ARP03] Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Heidelberg, Germany. 156

[ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 36–54, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany. 50

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. `http://eprint.iacr.org/2013/689`. 91, 110

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. Cryptology ePrint Archive, Report 2015/173, 2015. `http://eprint.iacr.org/2015/173`. 77, 96, 104, 112

[AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14: 21st Conference on Computer and Communications Security*, pages 646–658, Scottsdale, AZ, USA, November 3–7, 2014. ACM Press. 47, 77, 96, 103, 111, 112

[App14] Benny Applebaum. Bootstrapping obfuscators via fast pseudorandom functions. In Sarkar and Iwata [SI14], pages 162–172. 220

[AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Dodis and Nielsen [DN15], pages 528–556. 77, 96, 104, 111, 112

[AIK04] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. In *45th Annual Symposium on Foundations of Computer Science*, pages 166–175, Rome, Italy, October 17–19, 2004. IEEE Computer Society Press. 218, 220, 221

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006. 218, 220, 221, 227, 228

[AIKW13] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In Canetti and Garay [CG13], pages 166–184. 220

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. 13, 298

[BMSZ15] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: The case of evasive circuits. Cryptology ePrint Archive, Report 2015/167, 2015. `http://eprint.iacr.org/2015/167`. 112

[Bae14] Paul Baecher. *Cryptographic Reductions: Classification and Applications to Ideal Models*. PhD thesis, TU Darmstadt, Oktober 2014. 33

[BBM13] Paul Baecher, Christina Brzuska, and Arno Mittelbach. Reset indifferentiability and its consequences. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 154–173, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 192, 201

[BFFS13] Paul Baecher, Pooya Farshim, Marc Fischlin, and Martijn Stam. Ideal-cipher (ir)reducibility for blockcipher-based hash functions. In Johansson and Nguyen [JN13], pages 426–443. 33

[BF11] Paul Baecher and Marc Fischlin. Random oracle reducibility. In Rogaway [Rog11], pages 21–38. 33

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science*, pages 106–115, Las Vegas, Nevada, USA, October 14–17, 2001. IEEE Computer Society Press. 256

[BGK+14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Nguyen and Oswald [NO14], pages 221–238. 77, 96, 103, 104, 105, 107, 111, 112

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6:1–6:48, May 2012. Preliminary version [BGI+01]. 6, 47, 70, 71, 73, 74, 75, 77, 78, 79, 80, 95, 140, 142, 274

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [Kil01], pages 1–18. 6, 47, 70, 302

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th Annual ACM Symposium on Theory of Computing*, pages 484–493, Montréal, Québec, Canada, May 19–21, 2002. ACM Press. 256

[Bar86] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc 1. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 1–5. ACM, 1986. 98

[BBO07] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 535–552, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany. 5, 9, 60, 62, 154, 183, 200

[BBP03] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid encryption problem. Cryptology ePrint Archive, Report 2003/077, 2003. `http://eprint.iacr.org/2003/077`. 3

[BBP04] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Cachin and Camenisch [CC04], pages 171–188. 3, 41, 148

[BBN+09] Mihir Bellare, Zvika Brakerski, Moni Naor, Thomas Ristenpart, Gil Segev, Hovav Shacham, and Scott Yilek. Hedged public-key encryption: How to protect against bad randomness. In Matsui [Mat09], pages 232–249. 156, 162, 163

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany. 5, 35, 190

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Krawczyk [Kra98], pages 26–45. 57

[BFOR08] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In Wagner [Wag08], pages 360–378. 60, 62, 63, 154

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Oswald and Fischlin [OF15b], pages 627–656. 157, 200, 205, 239, 281, 282

[BHK13:p] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Canetti and Garay [CG13], pages 398–415. 4, 5, 9, 18, 178, 179, 180, 181, 182, 183, 184, 186, 187, 188, 189, 192, 203, 204, 205, 212, 304

[BHK13:f] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., Aug 1, 2013. (Full version of [BHK13:p].) `http://eprint.iacr.org/2013/424/20130801:043135`.). 6, 179, 183, 184, 185, 189, 195, 196, 197, 198, 200, 204, 205, 209, 210

[BHK13:1] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., Oct 17, 2013. (Latest version prior to our second attack.) `http://eprint.iacr.org/2013/424`. 179, 181, 184, 204, 205, 207, 208, 209, 217, 218, 219, 230

[BHK13:2] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. Cryptology ePrint Archive, Report 2013/424., May 17, 2014. (Latest version at the time of writing.) `http://eprint.iacr.org/2013/424`. 174, 178, 179, 184, 187, 190, 205, 209, 210, 212, 213, 235, 238, 239, 256

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press. 182

[BHK15] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge decryption be disallowed? *Journal of Cryptology*, 28(1):29–48, January 2015. 57

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1431–1440, New York, NY, USA, 2015. ACM. 60

[BK11] Mihir Bellare and Sriram Keelveedhi. Authenticated and misuse-resistant encryption of key-dependent data. In Rogaway [Rog11], pages 610–629. 173

[BKR13] Mihir Bellare, Sriram Keelveedhi, and Thomas Ristenpart. Message-locked encryption and secure deduplication. In Johansson and Nguyen [JN13], pages 296–312. 173, 174, 183, 205

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Garay and Gennaro [GG14a], pages 1–19. 60

[BRT12] Mihir Bellare, Thomas Ristenpart, and Stefano Tessaro. Multi-instance security and its application to password-based cryptography. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 312–329, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany. 65

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press. 1, 2, 27, 28, 32, 35, 41, 45, 57, 58, 180, 239

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany. 2, 41, 58, 183, 188

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Maurer [Mau96], pages 399–416. 2, 41

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany. 20, 193

[BS15] Mihir Bellare and Igors Stepanovs. Point-function obfuscation: A framework and generic constructions. Cryptology ePrint Archive, Report 2015/703, 2015. `http://eprint.iacr.org/`. 132, 133

[BST14] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Poly-many hardcore bits for any one-way function and a framework for differing-inputs obfuscation. In Sarkar and Iwata [SI14], pages 102–121. 10, 18, 54, 78, 238

[BST15] Mihir Bellare, Igors Stepanovs, and Stefano Tessaro. Contention in cryptoland: Obfuscation, leakage and uce. Cryptology ePrint Archive, Report 2015/487, 2015. `http://eprint.iacr.org/`. 137, 142, 147, 150, 151, 152, 155, 208, 217, 218, 231

[BG81] C. H. Bennett and J. Gill. Relative to a random oracle A, $P^A \neq NP^A \neq coNP^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, 1981. 193, 194

[Bih03] Eli Biham, editor. *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, Warsaw, Poland, May 4–8, 2003. Springer, Heidelberg, Germany. 309, 313

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 520–537, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany. 305

[BC14] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. *Journal of Cryptology*, 27(2):317–357, April 2014. Preliminary version [BC10]. 75, 76, 95, 119, 120, 123, 134

[BCC+14] Nir Bitansky, Ran Canetti, Henry Cohn, Shafi Goldwasser, Yael Tauman Kalai, Omer Paneth, and Alon Rosen. The impossibility of obfuscation with auxiliary input or a universal simulator. In Garay and Gennaro [GG14b], pages 71–89. xxv, 11, 75, 178, 263, 266, 270, 271, 274

[BCKP14] Nir Bitansky, Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On virtual grey box obfuscation for general circuits. In Garay and Gennaro [GG14b], pages 108–125. 76, 95, 151

[BGT14] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. `http://eprint.iacr.org/2014/771`. 265, 267

[BP12] Nir Bitansky and Omer Paneth. Point obfuscation and 3-round zero-knowledge. In Cramer [Cra12], pages 190–208. 119, 122, 123, 127, 132, 142, 149, 150, 151, 152

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. Cryptology ePrint Archive, Report 2015/163, 2015. `http://eprint.iacr.org/2015/163`. 77, 96, 104, 112

[Bla06] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340, Graz, Austria, March 15–17, 2006. Springer, Heidelberg, Germany. 41

[BRS03] John Black, Phillip Rogaway, and Thomas Shrimpton. Encryption-scheme security in the presence of key-dependent messages. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 62–75, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Heidelberg, Germany. 3, 183, 238, 239

[Ble98] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Krawczyk [Kra98], pages 1–12. 58

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In STOC 1988 [STO88], pages 103–112. 108

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13(4):850–864, 1984. 52, 53

[BCFW09] Alexandra Boldyreva, David Cash, Marc Fischlin, and Bogdan Warinschi. Foundations of non-malleable hash and one-way functions. In Matsui [Mat09], pages 524–541. 4, 180

[BFO08] Alexandra Boldyreva, Serge Fehr, and Adam O'Neill. On notions of security for deterministic encryption, and efficient constructions without random oracles. In Wagner [Wag08], pages 335–359. 60, 62, 63, 154

[BF05] Alexandra Boldyreva and Marc Fischlin. Analysis of random oracle instantiation scenarios for OAEP and other practical schemes. In Shoup [Sho05], pages 412–429. 156, 169, 180

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany. 112, 189

[BF03] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. 188

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Kilian [Kil01], pages 213–229. 156

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Ishai [Ish11], pages 253–273. 7

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003. 104, 105

[BV98] Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany. 51

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 280–300, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany. 87, 88, 89

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/2014/930`. 112

[BS89] Ravi B Boppana and Michael Sipser. *The complexity of finite functions*. Laboratory for Computer Science, Massachusetts Institute of Technology, 1989. 99

[BDFP86] Allan Borodin, Danny Dolev, Faith E Fich, and Wolfgang Paul. Bounds for width two branching programs. *SIAM Journal on Computing*, 15(2):549–560, 1986. 98

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Lindell [Lin14], pages 52–73. 91, 92, 110, 133, 241, 244, 246, 250, 252, 253, 287, 288, 292

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519. 87, 88, 89

[BP13] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. Cryptology ePrint Archive, Report 2013/703, 2013. `http://eprint.iacr.org/2013/703`. 91

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, Massachusetts, USA, January 8–10, 2012. Association for Computing Machinery. 108

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Lindell [Lin14], pages 1–25. 47, 77, 82, 96, 103, 111, 112

[Bra90] Gilles Brassard, editor. *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany. 310, 318

[BFM13a] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Personal communication with Mihir Bellare, Viet Tung Hoang and Sriram Keelveedhi. Sep, 2013. 183, 184, 203, 217

[BFM13b] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Unpublished manuscript, Oct, 2013. 80

[BFM14a] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Indistinguishability obfuscation and UCEs: The case of computationally unpredictable sources. In Garay and Gennaro [GG14a], pages 188–205. 178, 179, 184, 209, 217

[BFM14b] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/867, 2014. `http://eprint.iacr.org/2014/867`. 171, 173

[BFM15] Christina Brzuska, Pooya Farshim, and Arno Mittelbach. Random-oracle uninstantiability from indistinguishability obfuscation. In Dodis and Nielsen [DN15], pages 428–455. 136, 153, 178, 179, 200

[BM14a] Christina Brzuska and Arno Mittelbach. Indistinguishability obfuscation versus multi-bit point obfuscation with auxiliary input. In Sarkar and Iwata [SI14], pages 142–161. 136, 137, 260

[BM14b] Christina Brzuska and Arno Mittelbach. Personal communication with Mihir Bellare. Oct, 2014. 207

[BM14c] Christina Brzuska and Arno Mittelbach. Using indistinguishability obfuscation via UCEs. In Sarkar and Iwata [SI14], pages 122–141. 178, 179, 210, 235

[BM15a] Christina Brzuska and Arno Mittelbach. Deterministic public-key encryption from indistinguishability obfuscation and point obfuscation. Unpublished Manuscript, 2015. 178, 281

[BM15b] Christina Brzuska and Arno Mittelbach. Universal computational extractors and the superfluous padding assumption for indistinguishability obfuscation. Cryptology ePrint Archive, Report 2015/581, 2015. `http://eprint.iacr.org/`. 178, 179, 235, 255, 263, 266

[CC04] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany. 303, 318

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany. 4, 8, 117, 119, 122, 123, 130, 131, 132, 150, 151, 169, 180

[CCR15] Ran Canetti, Yilei Chen, and Leonid Reyzin. On the correlation intractability of obfuscated pseudorandom functions. Cryptology ePrint Archive, Report 2015/334, 2015. `http://eprint.iacr.org/2015/334`. 238

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 489–508, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany. 118, 119, 121, 127, 142, 147

[CG13] Ran Canetti and Juan A. Garay, editors. *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 302, 303

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In STOC 1998 [STO98], pages 209–218. 2, 3, 8, 26, 27, 35, 40, 42, 136, 148, 156, 157, 295

[CGH03] Ran Canetti, Oded Goldreich, and Shai Halevi. On the random-oracle methodology as applied to length-restricted signature schemes. Cryptology ePrint Archive, Report 2003/150, 2003. `http://eprint.iacr.org/2003/150`. 41, 148

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Biham [Bih03], pages 255–271. 156

[CKP15] Ran Canetti, Yael Tauman Kalai, and Omer Paneth. On obfuscation with random oracles. In Dodis and Nielsen [DN15], pages 456–467. 75

[CKVW10] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In Micciancio [Mic10], pages 52–71. 116, 231

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega (log n) rounds. In *33rd Annual ACM Symposium on Theory of Computing*, pages 570–579, Crete, Greece, July 6–8, 2001. ACM Press. 256

[CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Dodis and Nielsen [DN15], pages 468–497. 71

[CMR98] Ran Canetti, Daniele Micciancio, and Omer Reingold. Perfectly one-way probabilistic hash functions (preliminary version). In STOC 1998 [STO98], pages 131–140. 4, 131, 180

[CV13] Ran Canetti and Vinod Vaikuntanathan. Obfuscating branching programs using black-box pseudo-free groups. Cryptology ePrint Archive, Report 2013/500, 2013. `http://eprint.iacr.org/2013/500`. 96, 103, 104, 112

[CHL+15] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. In Oswald and Fischlin [OF15a], pages 3–12. 112

[CT02] Christian S. Collberg and Clark Thomborson. Watermarking, tamper-proofing, and obfuscation-tools for software protection. *Software Engineering, IEEE Transactions on*, 28(8):735–746, 2002. 68

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Shoup [Sho05], pages 430–448. 191, 193, 194, 195

[CJNP02] Jean-Sébastien Coron, Marc Joye, David Naccache, and Pascal Paillier. Universal padding schemes for RSA. In Yung [Yun02], pages 226–241. 180

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 476–493, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany. 111, 112

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. `http://eprint.iacr.org/2014/975`. 112

[Cra12] Ronald Cramer, editor. *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany. 306, 312

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [Kra98], pages 13–25. 59

[Dag13] Özgür Dagdelen. *The Cryptographic Security of the German Electronic Identity Card*. PhD thesis, TU Darmstadt, Darmstadt, 2013. 2

[DFG+13] Özgür Dagdelen, Marc Fischlin, Tommaso Gagliardoni, Giorgia Azzurra Marson, Arno Mittelbach, and Cristina Onete. A cryptographic analysis of OPACITY - (extended abstract). In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 345–362, Egham, UK, September 9–13, 2013. Springer, Heidelberg, Germany. 2

[Dam90] Ivan Damgård. A design principle for hash functions. In Brassard [Bra90], pages 416–427. 190, 191

[DFF+14] JeanPaul Degabriele, Victoria Fehr, Marc Fischlin, Tommaso Gagliardoni, Felix Günther, GiorgiaAzzurra Marson, Arno Mittelbach, and KennethG. Paterson. Unpicking plaid. 8893:1–25, 2014. 58

[DGHM13] Gregory Demay, Peter Gaži, Martin Hirt, and Ueli Maurer. Resource-restricted indifferentiability. In Johansson and Nguyen [JN13], pages 664–683. 192, 201

[Den02] Alexander W. Dent. Adapting the weaknesses of the random oracle model to the generic group model. In Zheng [Zhe02], pages 100–109. 41, 105

[Dic05] LE Dickson. Goldbach's empirical theorem: Every integer is a sum of two primes. *LE Dickson. History of the Theory of Numbers*, 1:421–424, 2005. 39

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. 6, 47, 68

[DGG+15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Oswald and Fischlin [OF15a], pages 101–126. 183, 239

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Mitzenmacher [Mit09], pages 621–630. 133

[DN15] Yevgeniy Dodis and Jesper Buus Nielsen, editors. *TCC 2015: 12th Theory of Cryptography Conference, Part II*, volume 9015 of *Lecture Notes in Computer Science*, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany. 302, 308, 309, 313, 316

[DS05] Yevgeniy Dodis and Adam Smith. Correcting errors without leaking partial information. In Gabow and Fagin [GF05], pages 654–663. 131

[DAB+02] John R. Douceur, Atul Adya, William J. Bolosky, Dan Simon, and Marvin Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *International Conference on Distributed Computing Systems*, pages 617–624, 2002. 173, 183, 205

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany. 57

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985. 41

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany. 27, 32

[Fis99] Marc Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: Improvements and applications. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 432–445, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany. 131

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of schnorr signatures. In Johansson and Nguyen [JN13], pages 444–460. 32

[FLR+10] Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 303–320, Singapore, December 5–9, 2010. Springer, Heidelberg, Germany. 32

[FOC82] *23rd Annual Symposium on Foundations of Computer Science*, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press. 322

[FGK+13] David Mandell Freeman, Oded Goldreich, Eike Kiltz, Alon Rosen, and Gil Segev. More constructions of lossy and correlation-secure trapdoor functions. *Journal of Cryptology*, 26(1):39–74, January 2013. 51, 200, 205

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany. 9, 40, 59, 156, 164, 168, 180

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Kilian [Kil01], pages 260–274. 58

[FOPS04] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. *Journal of Cryptology*, 17(2):81–104, March 2004. 41, 58

[FOR12] Benjamin Fuller, Adam O'Neill, and Leonid Reyzin. A unified approach to deterministic encryption: New constructions and a connection to computational entropy. In Cramer [Cra12], pages 582–599. 52, 63, 154

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984. 98

[GF05] Harold N. Gabow and Ronald Fagin, editors. *37th Annual ACM Symposium on Theory of Computing*, Baltimore, Maryland, USA, May 22–24, 2005. ACM Press. 311, 321

[GMP+08] Sebastian Gajek, Mark Manulis, Olivier Pereira, Ahmad-Reza Sadeghi, and Jörg Schwenk. Universally composable security analysis of TLS. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008: 2nd International Conference on Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 313–327, Shanghai, China, October 31 – November 1, 2008. Springer, Heidelberg, Germany. 2

[GG14a] Juan A. Garay and Rosario Gennaro, editors. *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 304, 308, 313, 320

[GG14b] Juan A. Garay and Rosario Gennaro, editors. *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany. 305

[GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Johansson and Nguyen [JN13], pages 1–17. 104, 105, 111, 112

[GGH+13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49,

Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press. 6, 7, 47, 70, 77, 85, 90, 96, 101, 102, 103, 107, 108, 110, 111, 183

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Garay and Gennaro [GG14a], pages 518–535. 54, 77, 92, 94

[GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014. `http://eprint.iacr.org/2014/666`. 112

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press. 85, 271

[Gen03] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In Biham [Bih03], pages 272–293. 156

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Mitzenmacher [Mit09], pages 169–178. 107

[GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Dodis and Nielsen [DN15], pages 498–527. 112

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. `http://eprint.iacr.org/2014/929`. 112

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/2014/309`. 47, 77, 96, 103, 111

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Zheng [Zhe02], pages 548–566. 156

[Gol42] Christian Goldbach. LETTRE XLIII. goldbach à euler, Jun 1742. 39

[Gol90] Oded Goldreich. A note on computational indistinguishability. *Information Processing Letters*, pages 277–281, 1990. 79, 130

[Gol00] Oded Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, New York, NY, USA, 2000. 13, 80

[Gol06] Oded Goldreich. On post-modern cryptography. Cryptology ePrint Archive, Report 2006/461, 2006. `http://eprint.iacr.org/2006/461`. 27

[Gol08] Oded Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, New York, NY, USA, 1 edition, 2008. 298

[Gol13] Oded Goldreich. my choices [oded goldreich, started 2009]—candidate indistinguishability obfuscation and functional encryption for all circuits, 2013. 47, 77

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press. 33, 88, 89, 276

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *21st Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, USA, May 15–17, 1989. ACM Press. 53

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the Fiat-Shamir paradigm. In *44th Annual Symposium on Foundations of Computer Science*, pages 102–115, Cambridge, Massachusetts, USA, October 11–14, 2003. IEEE Computer Society Press. 3, 41, 148

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *46th Annual Symposium on Foundations of Computer Science*, pages 553–562, Pittsburgh, PA, USA, October 23–25, 2005. IEEE Computer Society Press. 71, 72, 75, 138

[GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Vadhan [Vad07], pages 194–213. 314

[GR14] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. *Journal of Cryptology*, 27(3):480–505, July 2014. Preliminary version [GR07]. 80, 83, 84, 264

[GIS+10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Micciancio [Mic10], pages 308–326. 220

[GOR11] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In Ishai [Ish11], pages 182–200. 46, 51

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309. 7

[GKMZ14] Matthew D. Green, Jonathan Katz, Alex J. Malozemoff, and Hong-Sheng Zhou. A unified approach to idealized model separations via indistinguishability obfuscation. Cryptology ePrint Archive, Report 2014/863, 2014. `http://eprint.iacr.org/2014/863`. 41, 148, 157

[Had00] Satoshi Hada. Zero-knowledge and code obfuscation. In Tatsuaki Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 443–457, Kyoto, Japan, December 3–7, 2000. Springer, Heidelberg, Germany. 6, 47, 70

[HK07] Shai Halevi and Hugo Krawczyk. Security under key-dependent inputs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 07: 14th Conference on Computer and Communications Security*, pages 466–475, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press. 3

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 33

[Hil01] David Hilbert. Mathematische probleme. *Archiv der Mathematik und Physik*, 1(3):44–63, 1901. 39

[HMLS07] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. In Vadhan [Vad07], pages 214–232. 131

[HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Nguyen and Oswald [NO14], pages 201–220. 238

[Hol15] Justin Holmgren. On necessary padding with io. Cryptology ePrint Archive, Report 2015/627, 2015. `http://eprint.iacr.org/`. 11, 178, 263, 266, 270, 273, 274

[Hor15] Máté Horváth. Survey on cryptographic obfuscation. Cryptology ePrint Archive, Report 2015/412, 2015. `http://eprint.iacr.org/`. 96, 110

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Structure in Complexity Theory Conference, 1995., Proceedings of Tenth Annual IEEE*, pages 134–147. IEEE, 1995. 79

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press. 79, 130

[Ish11] Yuval Ishai, editor. *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany. 307, 314

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Redondo Beach, California, USA, November 12–14, 2000. IEEE Computer Society Press. 220

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 244–256, 2002. 220

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Ladner and Dwork [LD08], pages 433–442. 218, 220, 221, 227

[IPS15] Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Dodis and Nielsen [DN15], pages 668–697. 110

[JSS12] Tibor Jager, Sebastian Schinzel, and Juraj Somorovsky. Bleichenbacher's attack strikes again: Breaking PKCS#1 v1.5 in XML encryption. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *ESORICS 2012: 17th European Symposium on Research in Computer Security*, volume 7459 of *Lecture Notes in Computer Science*, pages 752–769, Pisa, Italy, September 10–12, 2012. Springer, Heidelberg, Germany. 58

[JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany. 302, 304, 310, 311, 312, 320

[RFC 3447] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), February 2003. 2, 28, 58

[RFC 2313] B. Kaliski. PKCS #1: RSA Encryption Version 1.5. RFC 2313 (Informational), March 1998. Obsoleted by RFC 2437. 58

[RFC 2437] B. Kaliski and J. Staddon. PKCS #1: RSA Cryptography Specifications Version 2.0. RFC 2437 (Informational), October 1998. Obsoleted by RFC 3447. 58

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*. Chapman & Hall/CRC, 2007. 13

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 669–684, Berlin, Germany, November 4–8, 2013. ACM Press. 87, 88, 89

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In STOC 1988 [STO88], pages 20–31. 103

[Kil01] Joe Kilian, editor. *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany. 303, 307, 312, 321

[KM04] Neal Koblitz and Alfred Menezes. Another look at "provable security". Cryptology ePrint Archive, Report 2004/152, 2004. `http://eprint.iacr.org/2004/152`. 42, 148, 149

[KM06] Neal Koblitz and Alfred Menezes. Another look at generic groups. Cryptology ePrint Archive, Report 2006/230, 2006. `http://eprint.iacr.org/2006/230`. 27, 42, 148

[KM15] Neal Koblitz and Alfred Menezes. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Report 2015/140, 2015. `http://eprint.iacr.org/2015/140`. 3, 42, 149, 180

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, January 2007. 3

[KMN+14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th Annual Symposium on Foundations of Computer Science*, pages 374–383, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press. Incorporates an earlier manuscript of Moran and Rosen [MR13]. 79, 80

[KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/2014/925`. 110

[Kra98] Hugo Krawczyk, editor. *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany. 303, 306, 310

[Kra14] Hugo Krawczyk, editor. *PKC 2014: 17th International Conference on Theory and Practice of Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany. 307, 318

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. IETF Internet Request for Comments 2104, February 1997. 5, 35, 190

[LD08] Richard E. Ladner and Cynthia Dwork, editors. *40th Annual ACM Symposium on Theory of Computing*, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press. 316, 320

[Lee59] Chang-Yeong Lee. Representation of switching circuits by binary-decision programs. *Bell System Technical Journal*, 38(4):985–999, 1959. 97

[Lin14] Yehuda Lindell, editor. *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany. 307, 318

[LR86] Michael Luby and Charles Rackoff. How to construct pseudo-random permutations from pseudo-random functions (abstract). In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, page 447, Santa Barbara, CA, USA, August 18–22, 1986. Springer, Heidelberg, Germany. 283

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Cachin and Camenisch [CC04], pages 20–39. 8, 65, 117, 119, 121, 122, 123, 131, 136, 137, 138, 148, 149, 153, 239, 256

[MMN15] Mohammad Mahmoody, Ameer Mohammed, and Soheil Nematihaji. More on impossibility of virtual black-box obfuscation in idealized models. Cryptology ePrint Archive, Report 2015/632, 2015. `http://eprint.iacr.org/`. 75

[MH14a] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via point obfuscation. In Lindell [Lin14], pages 95–120. 120, 125, 127, 128, 134, 142, 149, 150

[MH14b] Takahiro Matsuda and Goichiro Hanaoka. Chosen ciphertext security via UCE. In Krawczyk [Kra14], pages 56–76. 183, 188, 209, 239

[Mat09] Mitsuru Matsui, editor. *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany. 303, 306

[Mau96] Ueli M. Maurer, editor. *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany. 305, 320

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany. 23, 36, 191, 192

[Mer90] Ralph C. Merkle. One way hash functions and DES. In Brassard [Bra90], pages 428–446. 190, 191

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. 40

[Mic10] Daniele Micciancio, editor. *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany. 309, 314

[MSW14] Eric Miles, Amit Sahai, and Mor Weiss. Protecting obfuscation against arithmetic attacks. Cryptology ePrint Archive, Report 2014/878, 2014. `http://eprint.iacr.org/2014/878`. 111, 112

[Mit14] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Nguyen and Oswald [NO14], pages 603–621. 9, 178, 179, 192, 193, 194, 197, 201

[Mit09] Michael Mitzenmacher, editor. *41st Annual ACM Symposium on Theory of Computing*, Bethesda, Maryland, USA, May 31 – June 2, 2009. ACM Press. 311, 313

[MR13] Tal Moran and Alon Rosen. There is no indistinguishability obfuscation in pessiland. Cryptology ePrint Archive, Report 2013/643, 2013. `http://eprint.iacr.org/2013/643`. 317

[MSW08] Paul Morrissey, Nigel P. Smart, and Bogdan Warinschi. A modular security analysis of the TLS handshake protocol. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 55–73, Melbourne, Australia, December 7–11, 2008. Springer, Heidelberg, Germany. 2

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany. 111, 133

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press. 57, 59, 90, 108

[NO14] Phong Q. Nguyen and Elisabeth Oswald, editors. *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany. 302, 315, 318

[Nie02] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Yung [Yun02], pages 111–126. 8, 41, 148

[SHA-3] National Institute of Standards and Technology (NIST). NIST SHA-3 Competition. `http://csrc.nist.gov/groups/ST/hash/sha-3/index.html`. 31

[NIST07] National Institute of Standards and Technology (NIST). Federal Register / Vol. 72, No. 14 / Tuesday, January 23, 2007 / Notices. `http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Jan07.pdf`, 2007. 1

[FIPS 180-4] National Institute of Standards and Technology (NIST). FIPS 180-4, Secure Hash Standard (SHS). `http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf`, 2012. 31

[FIPS 186-4] National Institute of Standards and Technology (NIST). FIPS 186-4. Digital Signature Standard (DSS). `http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`, 2013. 2, 28, 41

[Oka93] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In Ernest F. Brickell, editor, *Advances in Cryptology – CRYPTO'92*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53, Santa Barbara, CA, USA, August 16–20, 1993. Springer, Heidelberg, Germany. 32

[OF15a] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 309, 311

[OF15b] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany. 303, 322

[Pas15] Rafael Pass and abhi shelat. Impossibility of VBB obfuscation with ideal constant-degree graded encodings. Cryptology ePrint Archive, Report 2015/383, 2015. `http://eprint.iacr.org/2015/383`. 75

[PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Garay and Gennaro [GG14a], pages 500–517. 77, 96, 103, 111

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Ladner and Dwork [LD08], pages 187–196. 200, 205

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Maurer [Mau96], pages 387–398. 41

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000. 32

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Santa Barbara, CA, USA, August 11–15, 1992. Springer, Heidelberg, Germany. 57

[RSV13] Ananth Raghunathan, Gil Segev, and Salil P. Vadhan. Deterministic public-key encryption for adaptively chosen plaintext distributions. In Johansson and Nguyen [JN13], pages 93–110. 63, 157, 158, 167

[Rey03] Leonid Reyzin. Random oracles and full-domain-hash (lectures 19-20), 2003. 1, 4, 42

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany. 20, 23, 164, 183, 192, 201, 239

[RAD78] Ronald L. Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978. 107

[RSA78] Ronald L Rivest, Adi Shamir, and Len Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 41

[Rog11] Phillip Rogaway, editor. *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany. 302, 304

[RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM Journal on Computing*, 39(7):3058–3088, 2010. 51

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th Annual ACM Symposium on*

Theory of Computing, pages 475–484, New York, NY, USA, May 31 – June 3, 2014. ACM Press. 70, 85, 87, 89, 141, 220, 236, 240, 271

[SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany. 7, 156

[SI14] Palash Sarkar and Tetsu Iwata, editors. *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany. 301, 305, 308

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949. 27, 29

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany. 105

[Sho98] Victor Shoup. *Why chosen ciphertext security matters.* IBM TJ Watson Research Center, 1998. 57

[Sho01] Victor Shoup. OAEP reconsidered. In Kilian [Kil01], pages 239–259. 58

[Sho02] Victor Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4):223–249, 2002. 58

[Sho05] Victor Shoup, editor. *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany. 306, 310

[STO88] *20th Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, USA, May 2–4, 1988. ACM Press. 306, 316

[STO98] *30th Annual ACM Symposium on Theory of Computing*, Dallas, Texas, USA, May 23–26, 1998. ACM Press. 309

[Sto76] Larry J Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976. 81, 299

[Vad07] Salil P. Vadhan, editor. *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany. 314, 315

[Wag08] David Wagner, editor. *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany. 303, 306

[Wee05] Hoeteck Wee. On obfuscating point functions. In Gabow and Fagin [GF05], pages 523–532. 8, 117, 123, 131, 132, 152

[Wee14] Hoeteck Wee. Functional encryption and its impact on cryptography. In Michel Abdalla and Roberto De Prisco, editors, *SCN 14: 9th International Conference on Security in Communication Networks*, volume 8642 of *Lecture Notes in Computer Science*, pages 318–323, Amalfi, Italy, September 3–5, 2014. Springer, Heidelberg, Germany. 7

[Wic13] Daniel Wichs. Barriers in cryptography with weak, correlated and leaky sources. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 111–126, Berkeley, CA, USA, January 9–12, 2013. Association for Computing Machinery. 23, 50, 51, 62, 155, 239, 240, 255

[Wil11] Stephen C. Williams. Analysis of the SSH key exchange protocol. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 356–374, Oxford, UK, December 12–15, 2011. Springer, Heidelberg, Germany. 2

[Win83] Robert S. Winternitz. Producing a one-way hash function from DES. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 203–207, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA. 3, 195

[Yao83] Andrew C Yao. Lower bounds by probabilistic arguments. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 420–428. IEEE, 1983. 98

[Yao82a] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In FOCS 1982 [FOC82], pages 160–164. 68, 182

[Yao82b] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In FOCS 1982 [FOC82], pages 80–91. 56, 80

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press. 227

[Yun02] Moti Yung, editor. *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Heidelberg, Germany. 310, 319

[Zhe02] Yuliang Zheng, editor. *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany. 310, 313

[Zim15] Joe Zimmerman. How to obfuscate programs directly. In Oswald and Fischlin [OF15b], pages 439–467. 47, 77, 96, 104, 111, 112