Attacks on the Fiat-Shamir Paradigm and Program Obfuscation

by

Yael Tauman Kalai

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2006

Author .....
Department of Electrical Engineering and Computer Science
June 15, 2006

Certified by .....
Shafi Goldwasser
RSA Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Attacks on the Fiat-Shamir Paradigm and Program Obfuscation

by

Yael Tauman Kalai

Submitted to the Department of Electrical Engineering and Computer Science
on June 15, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

The goal of cryptography is to construct *secure* and *efficient* protocols for various tasks. Unfortunately, it is often the case that protocols that are provably secure are not efficient enough for practical use. As a result, most protocols used in practice are *heuristics* that lack a proof of security. These heuristics are typically very efficient and are believed to be secure, though no proof of security has been provided. In this thesis we study the security of two types of such popular heuristics: (1) the Fiat-Shamir paradigm for constructing digital signature schemes, and (2) heuristics for obfuscation. We show that, in some sense, both of these types of heuristics are insecure.

This thesis consists of two parts:

1. **The insecurity of the Fiat-Shamir paradigm.** The Fiat-Shamir paradigm provides a general method for transforming any 3-round identification scheme, in which the verifier's message is random (and consists of his random coin tosses), into a digital signature scheme. The idea of the transformation is to replace the random message of the verifier in the identification scheme, with the value of some deterministic hash function evaluated on the first-round message (sent by the prover) and on the message to be signed. The Fiat-Shamir methodology for producing digital signature schemes quickly gained popularity both in theory and in practice, as it yields efficient and easy to implement digital signature schemes. The most important question however remained open: *are the digital signature schemes produced by the Fiat-Shamir methodology secure*?

   In this thesis, we answer this question negatively. We show that there exist secure 3-round public-coin identification schemes for which the Fiat-Shamir transformation yields *insecure* digital signature schemes for *any* hash function used by the transformation. This is in contrast to the work of Pointcheval and Stern, who proved that the Fiat-Shamir methodology always produces digital

signature schemes that are secure against chosen message attacks in the "Random Oracle Model" – when the hash function is modelled by a random oracle.

2. **The impossibility of obfuscation.** The goal of code obfuscation is to make a program completely "unintelligible" while preserving its functionality. Obfuscation has been used for many years in attempts to prevent reverse engineering, e.g., for copy protection, licensing schemes, and games. As a result, many heuristics for obfuscation have emerged, and the important question that remained is: *are these heuristics for obfuscation secure?*

   In this thesis, we show that there are many "natural" classes of functions for which obfuscation is not at all possible. This impossibility result holds in an augmentation of the formal obfuscation model of Barak *et al.* (2001) that includes auxiliary input.

In both of these parts, among other things, we make usage of Barak's technique for taking advantage of non black-box access to a program, this time in the context of digital signature schemes and in the context of obfuscation.

Thesis Supervisor: Shafi Goldwasser
Title: RSA Professor of Electrical Engineering and Computer Science

Acknowledgments

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In their famous paper, Diffie and Hellman [8] introduced the concept of public-key cryptography, thus setting the foundations for modern cryptography. Since then there has been a major effort to shift cryptography to provably secure grounds. The goal was to formally define security of different cryptographic primitives, and to construct protocols that are *provably secure* according to these definitions. For example, Goldwasser and Micali [18] introduced the notion of *semantic security* for encryption schemes, and presented a construction of an encryption scheme that is semantic secure. Golwasser, Micali and Rivest [19] introduced several definitions for secure digital signature schemes, the strongest which is *existential security against adaptive chosen message attacks*, and presented a construction of a digital signature scheme that is secure with respect to this strongest security definition.

Unfortunately, it is often the case that *provably secure* protocols are not efficient enough for practical use. As a result, most protocols used in practice are *heuristics* that lack a proof of security. These heuristics are typically very efficient and are believed to be secure, though no proof of security has been provided. In this thesis we study the security of two types of such popular heuristics: (1) the Fiat-Shamir paradigm for constructing digital signature schemes, and (2) heuristics for obfuscation. We show that, in some sense, both of these types of heuristics are insecure.

1.1 The Fiat-Shamir Heuristic

Fiat and Shamir [12] proposed a general paradigm for designing digital signature schemes. Their starting observation was that designing (provably) secure identification schemes (in which a sender *interactively* identifies himself to a receiver) can be done with greater ease and efficiency than seems to be the case for secure digital signature schemes (in which a signer *non-interactively* produces a digital signature for some message, to be verified valid by a verifier). Indeed, to this day, all of the provably secure signature schemes considered in the literature [19, 31, 34, 16, 7] are quite complicated and are not sufficiently efficient for many practical purposes.

Building on this observation, they proposed a two-step heuristic for designing secure digital signatures.

1. First, design a secure 3-round public-coin identification scheme. Namely, a secure 3-round identification scheme $(\alpha; \beta; \gamma)$ where α is the prover's first message, β is a random message sent by the verifier (consisting of the verifier's random coin tosses), and γ is the prover's response.

2. Second, choose a function ensemble $\mathcal{H}$, and obtain a digital signature scheme as follows. Let the signer choose at random a public key PK of the identification scheme designed in step 1, and append to it a randomly chosen function $h \in_R \mathcal{H}$. Namely, the verification-key of the signer is (PK, h). To sign a message M, the legal signer produces an accepting transcript $(\alpha; \beta; \gamma)$ of the interactive identification scheme (with respect to the public-key PK), where $\beta = h(\alpha, \mathsf{M})$.

The completeness of the identification scheme implies that the legal signer, who knows the secret-key corresponding to PK, can easily produce an accepting transcript for any M. The intuition for why this signature scheme is secure is that when h is a "sufficiently complicated" function (chosen by the real signer), then for any practical purpose it looks like a truly random function. Now, if h was a truly random function (say given to all parties via oracle access), then it should be hard for a forger, who takes as input a pair (PK, h), to find an accepting transcript $(\alpha; \beta; \gamma)$, such that

$\beta = h(\alpha, \mathsf{M})$. Loosely speaking, this is the case since interacting with a truly random function is essentially the same as interacting with a verifier that sends a truly random message β.

The complexity of a digital signature scheme resulting from the above paradigm is equivalent to the complexity of the starting identification scheme and the cost of evaluating the public function h. Current proposals for a public (keyless) function h are very efficient (e.g. [28]).

Due to the efficiency and the ease of design, the Fiat-Shamir paradigm quickly gained much popularity both in theory and in practice. Several digital signature schemes, including [35, 21, 32], were designed following this paradigm. The paradigm has also been used to achieve forward secure digital signature schemes [1] and to achieve better exact security [29]. Both of the above applications actually use a variation of the Fiat-Shamir paradigm. Still, they share the same basic structure: start with some secure 3-round public-coin identification scheme and transform it into a digital signature scheme, eliminating the random move of the verifier by an application of a fixed function h to different quantities determined by the protocol and to the message to be signed. The Fiat-Shamir paradigm, was also taken outside of the context of identification schemes and digital signature schemes, as it provides a general way of eliminating interaction from protocols by replacing the verifier with a function ensemble. In particular, it was used by Micali in the context of CS proofs [27]. The main question regarding any of these proposals is:

What can be proven about the security of the resulting schemes?

1.2 Program Obfuscation

The goal of program obfuscation is to make a program completely "unintelligible" while preserving its functionality. Obfuscation has been used for many years in attempts to prevent reverse engineering, e.g., for copy protection, licensing schemes, and games. The problem of program obfuscation, which practitioners have been engaged

in for many years, has only recently received attention in the theoretical community. This was initiated by the work of Barak *et al.* [4], who formulated the notion of program obfuscation. They formalized the intuition that a program should be "unintelligible" via the "virtual black box" property, which asserts that any predicate that can be computed (in polynomial time) from the obfuscated program can also be computed (in polynomial time) from the input-output behavior of the program (i.e., given black-box access to the program).

Barak *et al.* [4] showed that there exists a (contrived) class of functions $\mathcal{F}$ that is not obfuscatable; meaning that every class of polynomial time programs $\mathcal{P} = \{P_f : f \in \mathcal{F}\}$, where P_f computes the function f, is not obfuscatable. In contrast, Canetti and Wee [5, 36] showed how to obfuscate the particular class of point functions under various complexity assumptions. The class of point functions consists of all Boolean functions of the form $I_x(y) = 1$ if and only if $x = y$ (one may think of x as a password and the obfuscation of I_x as a public program that checks whether y is a valid password or not). The question that remained is:

Are most functions of interest obfuscatable?

1.3 Our Results

In this thesis we investigate the above two questions. Our results are mainly negative.

1.3.1 On the Insecurity of the Fiat-Shamir Paradigm

We prove that the Fiat-Shamir paradigm for designing digital signature schemes can lead to universally forgeable digital signatures. We do so by demonstrating the existence of a *secure* 3-round public-coin identification scheme for which the corresponding signature scheme, obtained by applying the Fiat-Shamir paradigm, is *insecure* with respect to any function ensemble implementing the public function.

Our result relies on the existence of one-way functions. Note, however, that if one-way functions do not exist then secure signature schemes do not exist and thus

the Fiat-Shamir paradigm always fails to produce secure signature schemes, as none exist. In this sense, our result is unconditional. The problems we demonstrate for the Fiat-Shamir paradigm apply to all other variations of the Fiat-Shamir paradigm proposed in the literature [29, 1].

We stress that our result does not imply that a particular identification scheme, such as [12, 35], cannot be proven to yield secure signature schemes with respect to some tailor-made function ensemble $\mathcal{H}$, under the Fiat-Shamir paradigm. What it does imply is that any proof of security would have to involve the particulars of the identification scheme and the $\mathcal{H}$ in question.

Our first idea is to make use of Barak's technique [2] of taking advantage of non black-box access to the program of the verifier. Intuitively, the idea is to take any secure 3-round public-coin identification scheme (which is not necessarily zero-knowledge) and extend its verdict function so that the verifier also accepts views which convince him that the prover knew in advance a (deterministic) function that computes the verifier's message. Since the verifier chooses its message at random, there is no way that the prover can guess in advance a (deterministic) function that computes the verifier's message, except with negligible probability, and therefore the scheme remains secure. However, when the identification scheme is converted into a signature scheme, by applying the Fiat-Shamir paradigm, the "verifier's message" is computed by a public (deterministic) function, chosen at random from some function ensemble, and is known in advance to everyone. A forger, who will now know in advance this function, will be able to generate an accepting view, which corresponds to a legitimate signature. This makes the signature scheme insecure regardless of which function ensemble is used to compute the "verifier's message" in the identification scheme.

The main technical challenge with implementing this approach is the following: How can the prover convince the verifier that he knew in advance a function that computes the verifier's message in a 3-round protocol? (Note that the size of the messages of the identification scheme should be a priori bounded by some fixed polynomial, whereas the size of this function is not a priori bounded by any fixed polynomial.)

Non-interactive CS-proofs of Micali [27] can be used to overcome this challenge. However, non-interactive CS-proofs (which themselves use a Fiat-Shamir type step to eliminate interaction) are only known to hold in the Random Oracle Model. Thus, we *first* get the (somewhat odd-looking) conditional result that if CS-proofs are realizable in the real world by some function ensemble, then there exists a secure identification scheme for which the Fiat-Shamir paradigm fails to produce a secure digital signature scheme for all function ensembles. Next, more generally, we show that the Fiat-Shamir paradigm is insecure regardless of whether or not CS-proofs are realizable in the real world. This part of the proof contains the bulk of difficulty and technical complication. It entails showing different extensions of secure 3-round public-coin identification schemes, which become insecure as digital signature schemes when the Fiat-Shamir paradigm is applied to them. All in all, we construct three identification schemes $\mathsf{ID}^1, \mathsf{ID}^2$ and ID^3, and prove that at least one of them demonstrates the insecurity of the Fiat-Shamir paradigm.

The Insecurity of Modifications of the Fiat-Shamir Paradigm. Two modifications of the Fiat-Shamir paradigm were considered in the literature: One due to Micali and Reyzin [29] and the other due to Abdalla, An, Bellare and Nampremre [1].

Micali and Reyzin [29] presented a method for constructing Fiat-Shamir-like signature schemes that yield better "exact security" than the original Fiat-Shamir method. In their method, they convert any identification scheme $(\alpha; \beta; \gamma)$ into a signature scheme, in which the signer first chooses β and only then produces α by computing $\alpha = h(\beta, \mathsf{M})$, where M is the message to be signed and h is the function used to eliminate interaction.[1]

Abdalla et. al. [1] defined a randomized generalization of the Fiat-Shamir paradigm, and showed that signature schemes, obtained from the generalized Fiat-Shamir paradigm, are secure (resp. forward secure) in the Random Oracle Model if and only if the underlying identification scheme is secure (resp. forward secure) against impersonation

[1]Note that this method can be applied only to identification schemes in which the sender can compute γ only given $(\mathsf{SK}, \mathsf{PK}, \alpha, \beta)$, and does not need any additional information on α.

under passive attacks. Their randomized method transforms any 3-round public-coin identification scheme $(\alpha; \beta; \gamma)$ into a signature scheme by replacing the random β with $h(\alpha, \mathsf{M}, r)$, where M is the message to be signed, h is the function used to eliminate interaction, and r is randomness chosen by the signer.

Using similar ideas to the ones presented in this thesis, one can prove the insecurity of these Fiat-Shamir modifications as well.

1.3.2 On the Impossibility of Obfuscation

We show that there are many "natural" classes of functions that are not obfuscatable under a *multiple obfuscation attack*. More precisely:

1. We first argue that the definition of obfuscation due to Barak *et al*. is not robust against *multiple obfuscation attacks.*

2. We then define *obfuscation w.r.t. auxiliary input*, which is robust against such attacks.

3. Finally, we argue that many "natural" classes of functions are not obfuscatable w.r.t. auxiliary input.

- **Multiple obfuscation attacks.** Let us first illustrate, via an example, that the definition of obfuscation due to Barak *et al*. is not robust against *multiple obfuscation attacks*. A primary usage of obfuscation, pointed out in [4], is to delegate cryptographic ability. Consider the task of decryption where $\mathsf{DEC}_{\mathsf{SK}}(\mathsf{C})$ stands for the decryption algorithm with secret key SK applied to the ciphertext C. Say Alice wants to delegate to her assistant Bob the ability to decrypt all documents which pertain to travel matters. This is easily achieved using an obfuscator $\mathcal{O}$ as follows. Define the function $\mathsf{DEC}^1_{\mathsf{SK}}(\mathsf{C}) \triangleq$ "compute $\mathsf{M} = \mathsf{DEC}_{\mathsf{SK}}(\mathsf{C})$; output M if and only if M starts with 'subject:travel'," and give Bob the obfuscated program $\mathcal{O}(\mathsf{DEC}^1_{\mathsf{SK}})$. Say that next month, Alice wants to delegate to Bob the ability to decrypt all documents which pertain to recruiting matters. This is achieved in the same manner:

let $\mathsf{DEC}^2_{\mathsf{SK}}(\mathsf{C}) \triangleq$ "compute $\mathsf{M} = \mathsf{DEC}_{\mathsf{SK}}(\mathsf{C})$; output M if and only if M starts with 'subject:recruiting'," and give Bob the obfuscated program $\mathcal{O}(\mathsf{DEC}^2_{\mathsf{SK}})$.

Thus, Bob is given obfuscations of two functions $\mathcal{O}(\mathsf{DEC}^1_{\mathsf{SK}})$ and $\mathcal{O}(\mathsf{DEC}^2_{\mathsf{SK}})$. We should obviously require that Bob cannot gain (in polynomial time) too much knowledge from $\mathcal{O}(\mathsf{DEC}^1_{\mathsf{SK}})$ *and* $\mathcal{O}(\mathsf{DEC}^2_{\mathsf{SK}})$. In particular, we should require that Bob cannot learn the corresponding secret-key SK from these two obfuscations. This is not guaranteed by the original definition of Barak *et al.* [4], and led us to the conclusion that we should consider *obfuscation with respect to auxiliary input*.

In the example above, Bob was given an obfuscation of two *dependent* functions $\mathsf{DEC}^1_{\mathsf{SK}}$ and $\mathsf{DEC}^2_{\mathsf{SK}}$ (these functions are dependent in the sense that they both use the same secret-key). Quite surprisingly, we show that even if Bob is given obfuscations of two *independent* functions, he can still carry out a *multiple obfuscation attack*.

For example, consider any secure digital signature scheme $\mathsf{SIG} = (\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$. Say a signer Alice with secret signing key SK_1 and a signer Carol with an *independent* signing key SK_2 share an assistant Bob. Alice delegates to Bob the ability to sign on her behalf the documents which pertain to personal matters, by giving Bob the obfuscated program $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$, where $\mathsf{SIGN}^1_{\mathsf{SK}_1}(\mathsf{M}) \triangleq$ "output $\mathsf{SIGN}_{\mathsf{SK}_1}(\mathsf{M})$ if and only if M starts with 'subject:personal'." Carol delegates to Bob the ability to sign on her behalf all documents which pertain to student affairs matters, by giving Bob the obfuscated program $\mathcal{O}(\mathsf{SIGN}^2_{\mathsf{SK}_2})$, where $\mathsf{SIGN}^2_{\mathsf{SK}_2}(\mathsf{M}) \triangleq$ "output $\mathsf{SIGN}_{\mathsf{SK}_2}(\mathsf{M})$ if and only if M starts with 'subject:student affairs'." Thus, Bob is given two obfuscations $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$ and $\mathcal{O}(\mathsf{SIGN}^2_{\mathsf{SK}_2})$.

We show that even in this case, where the two obfuscations are *independent*, there is some predicate s that can be learned from the two obfuscations that could not be learned if one of these obfuscations was replaced with black-box access to the function. In other words, even though each obfuscation by itself is essentially equivalent to a black-box, and even though these functions are completely inde-

pendent of each other, together they are not a valid obfuscation, in the sense that they reveal a predicate s that could not have been computed if one of these obfuscations was replaced with black-box access to the function. At first glance this may seem counterintuitive, since given only $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$, Bob himself could generate $\mathcal{O}(\mathsf{SIGN}^2_{\mathsf{SK}_2})$ (as it is independent of $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$), and thus learn the predicate s. However, this is not a contradiction, since s is actually a function of both SK_1 and SK_2, and it is efficiently computable from $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$ and SK_2, but it is not efficiently computable from $\mathcal{O}(\mathsf{SIGN}^1_{\mathsf{SK}_1})$ and $\mathcal{O}(\mathsf{SIGN}^2_{\mathsf{SK}_2})$. This led us to the conclusion that we should consider obfuscation w.r.t. auxiliary input, even if the auxiliary input is *independent* of the obfuscated function.

The idea of requiring security to hold even when an auxiliary input is available to the adversary is not new, and has been present since the early work on auxiliary-input zero-knowledge protocols [20]. In the context of zero-knowledge, the requirement is that for every $x \in L$ and for every auxiliary input z, whatever can be learned by a polynomial time verifier that is given (x, z) and interacts with a prover on input x, can also be learned by a polynomial time simulator that is given only (x, z). Intuitively, one may think of z as the *history* observed by the verifier in previous executions. Without this requirement, it is impossible to show secure (even sequential) composition of zero-knowledge protocols. Thus, by now, the terms zero-knowledge [19] and auxiliary-input zero-knowledge [20] have become one and the same. In the context of obfuscation, we incorporate auxiliary-input in a very similar manner into the definition.

- **Obfuscation w.r.t auxiliary input.** When considering *obfuscation w.r.t. auxiliary input*, we modify the "virtual black box" property, to require that for every auxiliary input z any predicate that can be learned by a polynomial time non-uniform adversary that is given z and an obfuscated program, can also be learned by a polynomial time non-uniform simulator that is given z and input/output access to the program. Notice that, as obfuscation w.r.t. auxiliary input is harder to satisfy than obfuscation without auxiliary input, the result of [4] already implies

the existence of classes of functions that cannot be obfuscated w.r.t. auxiliary input. Our emphasis is to show that this is actually true for wide and natural classes of functions.

We distinguish between two types of obfuscation w.r.t. auxiliary input: *obfuscation w.r.t. dependent auxiliary input* and *obfuscation w.r.t. independent auxiliary input*.[2]

Dependent auxiliary input. Let $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be a class of functions. We say that $\mathcal{O}$ is an *obfuscator w.r.t. dependent auxiliary input* for the class $\mathcal{F}$ if for every function $f \in \mathcal{F}$ the virtual black-box property holds even when the adversary (and the simulator) are given an additional auxiliary input z (this should hold for any z, including one that possibly depends on f).[3]

Independent auxiliary input. We say that $\mathcal{O}$ is an *obfuscator w.r.t. independent auxiliary input* for the class $\mathcal{F}$ if for every function $f \in \mathcal{F}$ the virtual black-box property holds even when the adversary (and the simulator) are given an additional auxiliary input z which is *independent* of f. To capture the independence of the auxiliary input z from the obfuscated function f, we fix z before f is chosen from the class $\mathcal{F}$. Formally, we require that for all auxiliary inputs z given to the adversary, the black-box property should hold for a *randomly* chosen function in $\mathcal{F}$.

At first it may seem that fixing the independent auxiliary input z to the adversary before choosing the function to be obfuscated, is equivalent to hard-wiring z to the adversary, and thus that an impossibility result for obfuscation w.r.t. independent auxiliary input implies an impossibility result for obfuscation w.r.t. [4]'s original definition. However, as was mentioned above (in the example of a multiple obfuscation attack w.r.t. independent auxiliary input) this intuition is misleading. Let us illustrate this via an example. Consider a z that satisfies the following three

[2]This distinction was not done in the context of zero-knowledge.

[3]This definition follows the lines of the definition of auxiliary-input zero-knowledge, which also allows the auxiliary input z to depend on the statement x being proven.

requirements: (1) given z and an obfuscation of $f \in_R \mathcal{F}$ it is easy to compute some predicate $\pi(f, z)$; (2) given z and black-box access to $f \in_R \mathcal{F}$ it is hard to compute the predicate $\pi(f, z)$; (3) given z and black-box access to $f \in_R \mathcal{F}$ *and* the value of some hard computation on z (say $g(z)$, for a hard function g) it is easy to compute the predicate $\pi(f, z)$. Then, certainly, requirements (1) and (2) imply that it is impossible to obfuscate f w.r.t. auxiliary input. In contrast, no such impossibility is implied for the [4]'s definition, since by requirements (1) and (3), if z is hard-wired to the adversary which is given z and an obfuscation of f, then $g(z)$ can be hard-wired to the simulator which is only given z and black-box access to f. This will enable the simulator to compute $\pi(f, z)$. We note that both in our impossibility results and in our examples we use an auxiliary input of this form.

Whereas requiring the black box property to hold when an auxiliary input is available, is a strengthening of the requirement made by the original definition [4], the fact that we require the black box property to hold for a *random* function in the class rather than for *every* function, is a weakening of the requirement in [4].[4] We emphasize that weakening the definition of obfuscation strengthens any impossibility result on obfuscation, which is the focus of this work. However, we do believe that this weakening is meaningful and sufficient for many positive applications of obfuscation, where the particular obfuscated function is chosen at random from some class of functions. This is how obfuscation is used in most of the examples of [4], where generally speaking a class of functions $\mathcal{F}$ corresponds to a class of cryptographic algorithms (e.g. a class of RSA decryption functions where each decryption function is w.r.t. a different secret key, a class of digital signature functions where each signing function is w.r.t. a different signing key, or a class of pseudo random functions, where each pseudo random function is w.r.t. a different seed) and a random choice of $f \in \mathcal{F}$ corresponds to choosing a particular key for the cryptographic algorithm at hand.

We give separate impossibility results for obfuscation w.r.t. independent auxiliary

[4]We note that the negative results of [4] hold with respect to this weakening.

input and obfuscation w.r.t. dependent auxiliary input. We stress that our impossibility results are unconditional and do not require any intractability assumptions such as the existence of one-way functions.

- **Impossibility of obfuscation w.r.t. auxiliary input**

Independent auxiliary input. Our first result considers obfuscation w.r.t. independent auxiliary input. Loosely speaking, we show that many natural *filter functions* (defined below) cannot be obfuscated w.r.t. independent auxiliary input.

Filter functions. Loosely speaking, each filter function is associated with a function f and an $\mathcal{NP}$ language L, and is denoted by f^L. The filter function f^L on input (x, w) checks whether $(x, w) \in \mathcal{R}_L$ (where $\mathcal{R}_L$ is the $\mathcal{NP}$ relation that corresponds to L), and outputs $f(x)$ if and only if w is a valid witness. Thus, f^L gives the value of $f(x)$ only to whoever knows a witness corresponding to x.

Formally, each class of filter functions is associated with a class of functions $\mathcal{F}$ and an $\mathcal{NP}$ language L. The class of *filter functions* $\mathcal{F}^L \triangleq \{f^L : f \in \mathcal{F}\}$ is the class of functions where each function $f^L \in \mathcal{F}^L$ is defined as follows: $f^L(x, w) = f(x)$ for every input $(x, w) \in \mathcal{R}_L$, and $f^L(x, w) = \perp$ for every input $(x, w) \notin \mathcal{R}_L$. For example, one may think of $\mathcal{F} = \{\mathsf{SIGN}_{\mathsf{SK}}\}$ as any class of signing functions, L as the set $\{(\mathsf{N}, y) : y \in QR_{\mathsf{N}}\}$ (the set of quadratic residues mod N), and $\mathsf{SIGN}^L_{\mathsf{SK}}$ as computing $\mathsf{SIGN}_{\mathsf{SK}}(\mathsf{N}, y)$ only for those users who supply the pair (N, y) together with a square-root of y modulo N. An analogy may be taken from the setting of certification authority: a user's identity corresponds to a pair (N, y) for which only the legal user knows a square root of y mod N, and when he presents this square-root to the trusted center, he gets from the authority a signed certificate of (N, y).

Our result on the impossibility of obfuscation w.r.t. independent auxiliary input is the following:

Result 1 (Informal): Let L be *any* $\mathcal{NP}$-complete language, and let $\mathcal{F}$ be any *any* class of pseudo-random functions, secure secret-key encryption algorithms, or randomized digital signature algorithms (where the coins used by the algorithms are replaced by a pseudo random function). Then the class of filter functions $\mathcal{F}^L$ cannot be obfuscated w.r.t. independent auxiliary input.

To express this result in its full generality, we need to introduce the concept of functions with *super-polynomial pseudo entropy*.[5]

Functions with super-polynomial pseudo entropy. Pseudo entropy can be thought of as a relaxation of pseudo-randomness. A class of functions $\mathcal{F}$ is pseudo-random if it is hard to distinguish between having oracle access to $f \in_R \mathcal{F}$ and having oracle access to a totally random function. The pseudo-randomness requirement is very strong: f needs to look *truly random* on *every* (polynomial time computable) element in the domain. Pseudo entropy requires the pseudo-randomness to hold only on a (small) subset of the domain. Moreover, the function need not look truly random on this subset; rather we require the function values on this subset to look as if they have high min entropy.

Specifically, we say that a class of functions $\mathcal{F}$ has *pseudo entropy at least* $p(\cdot)$ if for every $n \in \mathbb{N}$ there exists a polynomial size subset $I_n \subseteq \{0,1\}^n$ such that for a randomly chosen $f \in_R \mathcal{F}_n$, the random variable $\{f(x)\}_{x \in I_n}$ cannot be distinguished from a random variable which has statistical min-entropy $p(n)$.[6] We say that $\mathcal{F}$ has *super-polynomial pseudo entropy* if it has pseudo entropy at least $p(\cdot)$, for every polynomial $p(\cdot)$.

We claim that there are many natural classes of functions with super-polynomial pseudo entropy, and in particular we show (Claim 26) that *every* class of pseudo random functions [15], *every* class of secure secret-key encryption algorithms [18], and *every* class of randomized digital signature algorithm [19] (where the coins

[5]The term "pseudo entropy" was introduced by [23] in the context of random variables. We extend the use of this term to the context of classes of functions.

[6]We refer the reader to Section 4.3 for the precise definition.

used by the algorithms are replaced by a pseudo random function), has super-polynomial pseudo entropy.

Result 1 can now be stated more generally.

Result 1 (General): The class of filter functions $\mathcal{F}^L$ cannot be obfuscated (in the presence of independent auxiliary inputs), where L is *any* $\mathcal{NP}$-complete language and $\mathcal{F}$ is any class of functions that satisfies the following two properties:

1. $\mathcal{F}$ is *strongly unpredictable*. Namely, for every x and for a random $f \in_R \mathcal{F}$ it is hard to predict $f(x)$, given oracle access to f everywhere except x .
2. $\mathcal{F}$ has *super-polynomial pseudo entropy* on inputs in L (i.e., where the polynomial size subsets I_n are contained in $L \cap \{0,1\}^n$).[7,8]

Impossibility of Obfuscation w.r.t. Dependent Auxiliary Input. We next consider obfuscation w.r.t. dependent auxiliary input. We show that if *point-filter functions* (defined below) can be obfuscated then *every* class of functions with super-polynomial pseudo entropy cannot be obfuscated (in particular, every class of pseudo random functions, every class of secure secret-key encryption algorithms, and every class of randomized signing algorithms cannot be obfuscated).[9] Using a separate proof we show that this condition also implies that the class of decryption algorithms corresponding to *any* secure encryption scheme cannot be obfuscated.

Point-filter functions. Loosely speaking, each point-filter function is associated with an $\mathcal{NP}$ language L, a point x, and a secret bit b. On input w, it outputs its secret bit b if and only if w is a valid witness of x (with respect to the language

[7] We note that every class of pseudo-random functions, secure secret-key encryption algorithms and secure probabilistic digital signature algorithms (that use pseudo-random functions to replace their randomness), satisfies these two properties.

[8] Although it may seem that strong unpredictability implies super-polynomial pseudo entropy, and thus that item 2 is superfluous, this is not the case. For example an unpredictable function may be verifiable (i.e., hard to compute but easy to verify), and thus have pseudo entropy 0.

[9] We stress that [4] presents particular (contrived) examples of classes of pseudo-random functions, secret-key encryption algorithms and digital signature algorithms which are not obfuscatable.

L). We stress that the language L and the point x are public, whereas the output bit b is secret. Intuitively, the reason that it may be hard to compute the secret bit b is that it may be hard to find a valid witness w for x.[10]

More precisely, every class of point-filter functions is associated with some language $L \in \mathcal{NP}$ and is of the form $\Delta^L = \{\Delta^L_n\}_{n\in\mathbb{N}}$. Every function $\delta_{x,b} \in \Delta^L_n$ is associated with a public string $x \in \{0,1\}^n$ and a secret bit $b \in \{0,1\}$. The function $\delta_{x,b}$ reveals its secret bit b only on inputs w such that $(x,w) \in \mathcal{R}_L$ (where $\mathcal{R}_L$ is the $\mathcal{NP}$ relation corresponding to the language L). In order to emphasize that x is public, we append x to each output. Formally, $\delta_{x,b}(w) = (x,b)$ if $(x,w) \in \mathcal{R}_L$, and $\delta_{x,b}(w) = x$ otherwise. Examples of point-filter classes are $\Delta^{\mathsf{SIG}} = \{\delta_{\mathsf{VK},b}\}$, where $\delta_{\mathsf{VK},b}$ reveals its secret bit b only on inputs which are valid signatures w.r.t. the verification key VK, and the class $\Delta^{\mathsf{SAT}} = \{\delta_{\phi,b}\}$, where $\delta_{\phi,b}$ reveals its secret bit b only on inputs which satisfy the formula ϕ.

We can now state the result, on the impossibility of obfuscation w.r.t. dependent auxiliary input.

Result 2: *Every* class of functions with super-polynomial pseudo entropy cannot be obfuscated w.r.t. dependent auxiliary input, or for *every* $\mathcal{NP}$-complete language L, the class of point-filter functions Δ^L cannot be obfuscated w.r.t. dependent auxiliary input.

Thus, we exhibit two classes of functions and show that at least one of them cannot be obfuscated. An alternative, and interesting, conditional formulation of this result, is that if one could obfuscate a single point-filter class Δ^L for some $\mathcal{NP}$-complete language L, then every class of functions with super-polynomial pseudo

[10]Note the contrast between point-filter functions and point functions of [5, 36]. While point functions are zero everywhere except for one point, a point-filter function can be non-zero on exponentially many inputs (x may have exponentially many witnesses). Moreover, in a point-filter function, x is public and may yield information about the points w which yield the secret value b, whereas there is absolutely no information about which point yields a non-zero value in the point function case.

entropy could not be obfuscated.

The question of whether there exists an $\mathcal{NP}$-complete language L for which the point-filter class Δ^L is obfuscatable is thus a worthy future direction to pursue. We show that this question is related to a beautiful fundamental question on the existence of a *hard core predicate for $\mathcal{NP}$ languages* (see Section 4.5.1 for details).

1.4 Related Work

There are several works in the literature on the security (or insecurity) of the Fiat-Shamir paradigm and on the possibility (or impossibility) of program obfuscation.

1.4.1 Prior Work on the Security and Insecurity of the Fiat-Shamir Paradigm

There are two main papers addressing the security of the Fiat-Shamir paradigm. The first is a positive result due to Pointcheval and Stern [33], showing that the Fiat-Shamir paradigm is secure in the Random Oracle Model. The second is the work of Dwork, Naor, Reingold and Stockmeyer [9], showing that the security of the Fiat-Shamir paradigm is closely related to two previously studied problems: *the selective decommitment problem*,[11] and *the existence of* 3*-round public-coin weak zero-knowledge arguments for non* BPP *languages*. We note that our negative results, regarding the security of the Fiat-Shamir paradigm, have implications on these related problems.

In particular, the result of [9], that the existence of 3-round public-coin zero-knowledge protocols for non BPP languages implies the insecurity of the Fiat-Shamir paradigm, is worth elaborating on. It follows from the following simple observation. Suppose there exists a 3-round public-coin zero-knowledge argument for some hard

[11] In the selective decommitment problem, an adversary is given commitments to a collection of messages, and the adversary can ask for some subset of the commitments to be opened. The question is whether seeing the decommitments to these open plaintexts allows the adversary to learn something unexpected about the plaintexts that are still hidden.

language. View this zero-knowledge argument as a secure identification protocol.[12] The fact that the identification protocol is zero-knowledge (and not only honest verifier zero-knowledge) means that for *every verifier* there exists a simulator that can generate views which are computationally indistinguishable from the ones produced during the run of the identification protocol. As the Fiat-Shamir paradigm (applied to this identification protocol) essentially fixes a public program for the verifier of the zero-knowledge argument, a forger can now simply run the simulator for this fixed verifier to produce a view of the identification protocol, i.e. a valid digital signature.

This simple argument extends to any k-round public-coin zero-knowledge argument. Namely, if such a k-round public-coin zero-knowledge argument exists, it can be viewed as an identification protocol. Now, extend the original Fiat-Shamir paradigm to an *Extended-Fiat-Shamir paradigm* which replaces each message of the verifier (one round at a time) by applying a fixed public function to previous messages in the protocol. Then the same argument as above says, that the simulator for the k-round zero-knowledge protocol can be used to produce forgeries in the signature scheme resulting from the Extended-Fiat-Shamir paradigm.

Barak [2] has shown that under the assumption that collision-resistant function ensembles exist, every language in $\mathcal{NP}$ has a k-round (for some constant $k > 3$) public-coin zero-knowledge argument. Thus, it follows from [9] and [2] that the k-round Extended-Fiat-Shamir paradigm is insecure.

However, the Fiat-Shamir paradigm was defined, and has always been used, only for 3-round identification schemes. Barak's work does not apply to this case, and it is not known whether there exist 3-round public-coin ZK protocols for non BPP languages. Moreover, whereas all that can be deduced from [9, 2] is that the Fiat-Shamir paradigm (extended or otherwise) fails on zero-knowledge identification schemes (indeed it is the simulator for the zero-knowledge system which will produce forgeries), it leaves open the possibility that the (extended and ordinary) Fiat-Shamir paradigm works when the starting identification schemes are secure with respect to a less strict

[12] It is not necessarily a proof of knowledge but it is certainly a proof of ability of proving membership in L, which is hard for polynomial-time impersonating algorithms.

security requirement and are not zero-knowledge.

1.4.2 Prior work on the Possibility and Impossibility of Obfuscation

As was mentioned in Section 1.2, the theoretical investigation of obfuscation was initiated in the work of Barak *et al.* [4], who formalized the notion of obfuscation and proved that there are (contrived) classes of functions that are not obfuscatable. However, even prior to this work, Hada [22] considered the question of whether pseudo random functions can be obfuscated. He also considered a "virtual black box" type definition. However, he did not restrict the output to be a Boolean predicate. Instead, he defined the notion of obfuscation with respect to a given adversary. Namely, an obfuscator which is designed to work against a specific probabilistic polynomial-time adversary $\mathcal{A}$ (who may be given an auxiliary input). He gave a negative result for obfuscating *any* class of pseudo-random functions against the following adversary $\mathcal{A}$: $\mathcal{A}$ fixes a language $L \in \mathcal{NP}$, fixes a zero-knowledge proof (P, V) for L, and fixes a sequence $\{x_n\}_{n\in\mathbb{N}}$ such that $x_n \notin L$. Given any program computing a function f, it outputs an accepting view of $(\mathsf{P}, \mathsf{V}^*)(x_n)$, where V^* computes its messages by applying f to all previous messages. This can be done using the simulator of the zero-knowledge proof.[13] On the other hand, notice that the fact that $x_n \notin L$ together with the fact that f is a pseudo random function, implies that with black-box access to f it is computationally hard to output an accepting view of $(\mathsf{P}, \mathsf{V}^*)(x_n)$.

There are also several positive results that appeared in the literature. As was mentioned in Section 1.2, Canetti and Wee [5, 36] proved that point functions are obfuscatable (under complexity theoretic assumptions). Other positive results include [6], [10] and [25]. The work of [6] generalizes the work of [5], in the sense that it relies on weaker and more general assumptions. However, their work yields an obfuscator for the class of point functions, which is weaker in two aspects: (1) the obfuscator is *not* w.r.t. auxiliary input, whereas the work of [5] yields an obfuscator w.r.t auxiliary

[13] The result of Hada was conditioned on the existence of a constant-round public-coin zero knowledge protocol for a non-trivial language, which was later shown to exist by Barak [2].

input. (2) The "virtual black box" property holds only w.r.t. distributions with high min-entropy (i.e., where the point function I_x, to be obfuscated, is chosen according to some high min-entropy distribution over the class of all point functions). The work of [10] generalizes the work of [6]; it shows how to obfuscate (in the sense of [6]) proximity functions (and not only point functions), where a proximity function is of the form $I_{x,\tau}(y) = 1$ if and only if the Hamming distance between x and y is at most τ. The work of [25] generalizes the work of [5, 36] in the sense that it shows that many access control functions (not only point functions) can be obfuscated. However, their result is in the Random Oracle Model, which assumes black box access to a truly random function.

Chapter 2

Preliminaries

Notations: We use [19]'s notations and conventions for probabilistic algorithms. If A is a probabilistic algorithm then for any input x we let $A(x)$ refer to the probability space which assigns to any string σ the probability that $A(x)$ outputs σ. If S is a probability space then $x \leftarrow S$ denotes that x is randomly chosen according to S. If S is a finite set then $x \in_R S$ denotes that x is randomly chosen in the set S. For any probabilistic interactive Turing machines A and B, we let $(A, B)(x)$ refer to the transcript of their interaction on input x. We assume that at the end of the interaction B will always either accept or reject. We refer to this decision function of B as the *verdict function* of B. We abuse notion by saying that $(A, B)(x) = 1$ if B accepts, and we denote by $\mathsf{ACC}(B(x))$ the set of all transcripts that $B(x)$ accepts. We denote by $A|_\alpha$, machine A, restricted to sending α as its first message. More generally, we denote by $A|_{\alpha_1;\ldots;\alpha_t}$, machine A, restricted to sending α_i as its i'th message, for $i = 1, \ldots, t$. We adopt the standard way of modeling an efficient adversary as a non-uniform probabilistic polynomial-time Turing machine (or equivalently, as a polynomial size circuit family). Similarly, computational indistinguishability refers to indistinguishability by non-uniform probabilistic polynomial-time adversaries. For any binary relation $\mathcal{R}$ we denote by $L_{\mathcal{R}} \stackrel{\text{def}}{=} \{x : \exists w \text{ s.t. } (x, w) \in \mathcal{R}\}$ the language corresponding to $\mathcal{R}$.

Definition 1 (Negligible): *We say that a function $g(\cdot)$ is* negligible *if for every poly-*

nomial $p(\cdot)$ there exists $n_0 \in \mathbb{N}$ such that for every $n \geq n_0$ it holds that $g(n) < \frac{1}{p(n)}$.

For any function $g(\cdot)$, we let $g(n) = \mathsf{negl}(n)$ denote the fact that $g(\cdot)$ is a negligible function.

Definition 2 (Non-negligible): *We say that a function $g(\cdot)$ is* non-negligible *if it is not negligible. That is, we say that $g(\cdot)$ is non-negligible if there exists a polynomial $p(\cdot)$ such that for infinitely many n's it holds that $g(n) \geq \frac{1}{p(n)}$.*

For any function $g(\cdot)$, we let $g(n) = \mathsf{non\text{-}negl}(n)$ denote the fact that $g(\cdot)$ is a non-negligible function.

Definition 3 (one-way function): *We say that a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ is* one-way *if for every polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$,*

$$\Pr[C_n(y) = x \ : \ f(x) = y] = \mathsf{negl}(n)$$

(where the probability is over $y \leftarrow f(U_n)$).

Definition 4 (hash-function ensemble): *A* hash-function ensemble *is a family of polynomial size functions $\mathcal{F} = \{\mathcal{F}\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$, every $f \in \mathcal{F}_n$ is a function from $\{0,1\}^*$ to $\{0,1\}^n$.*

Definition 5 (collision resistant hash-function ensemble): *We say that a hash-function ensemble $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ is* collision resistant *if for every polynomial-size circuit family $C = \{C_n\}_{n\in\mathbb{N}}$,*

$$\Pr[C_n(f_n) = (x_1, x_2) \ : \ f_n(x_1) = f_n(x_2) \ \wedge \ x_1 \neq x_2] = \mathsf{negl}(n)$$

(where the probability is over $f_n \in_R \mathcal{F}_n$).[1]

[1] We abuse notation by letting f also denote the description of the function f.

Chapter 3

The Insecurity of the Fiat-Shamir Paradigm

3.1 Definitions

Let us start by giving the standard definitions (see [19, 11, 13]) of identification schemes (Section 3.1.1), signature schemes (Section 3.1.2), and the Fiat-Shamir paradigm (Section 3.1.3).

3.1.1 Identification Schemes

Definition 6 (Identification Scheme): *An* identification scheme (*or* ID scheme*, for short*) *is identified with a triplet* $(\mathsf{G}, \mathsf{S}, \mathsf{R})$*, where* G *is a key generation algorithm and* S *is the sender who wishes to prove his identity to the receiver* R*.*

- G *is a probabilistic polynomial-time Turing machine that, on input a security parameter* 1^n*, outputs a pair* $(\mathsf{SK}, \mathsf{PK})$*, such that the sizes of* SK *and* PK *are polynomially related to* n*.* SK *is referred to as the secret-key and* PK *is referred to as the public-key.*

- (S, R) *is a pair of probabilistic polynomial-time interactive Turing machines that take a public-key* PK *as common input. The sender* S *also takes as input a corresponding secret-key* SK*. Intuitively,* R *outputs* 1 *if and only if he is convinced*

that he is interacting with the sender who's identity is PK*. Formally, it is required that for any pair* $(\mathsf{SK}, \mathsf{PK})$ *in the support of* $\mathsf{G}(1^n)$*,*

$$\Pr[(\mathsf{S}(\mathsf{SK}), \mathsf{R})(\mathsf{PK}) = 1] = 1$$

(*where the probability is over the random coin tosses of* S *and* R)*.*

Recall that the Fiat-Shamir paradigm was defined as a method for converting 3-round public-coin ID schemes into signature schemes. We refer to such ID schemes as *canonical* ID schemes.

Definition 7 (Canonical ID Scheme): *A* canonical *ID scheme is a* 3*-round ID scheme with a transcript of the form* $(\alpha; \beta; \gamma)$*, where* α *is sent by the sender* S*,* β *is sent by the receiver* R *and consists of* R*'s random coins,*[1] *and* γ *is sent by the sender* S*.*

For a sender S, with keys $(\mathsf{SK}, \mathsf{PK})$ and randomness r, we denote $\alpha = \mathsf{S}_{(\mathsf{SK},\mathsf{PK})}(r)$ and $\gamma = \mathsf{S}_{(\mathsf{SK},\mathsf{PK})}(\alpha, \beta, r)$.

Security of ID Schemes. As with any cryptographic primitive, the notion of security considers adversarial goals (i.e., what it has to do to win) and adversarial capability (i.e., what attacks it is allowed). Naturally, for an ID scheme, the adversary's goal is impersonation: it wins if it can interact with the receiver (in the role of a sender), and convince him to accept. As for the adversary's capabilities, the adversary is modeled as a (probabilistic) polynomial-size circuit family. There are two natural attacks to consider: passive and active. Passive attacks correspond to eavesdropping, meaning the adversary is in possession of transcripts of conversations between the real sender and the receiver. Active attacks means that it gets to play the role of a receiver, interacting with the real sender in an effort to extract information. We will not give formal definitions of secure ID schemes, as they are not needed for the understanding this work.

[1] For the simplicity of notations (and without loss of generality), throughout this paper we assume that the random string β sent by the receiver is of length n, where n is the security parameter.

We note that assuming the existence of one-way functions, there exist ID schemes which are secure against active attacks.[2] Throughout this manuscript, security of an ID scheme should be interpreted as security against active attacks.

3.1.2 Signature Schemes

Definition 8 (Signature scheme): *A* signature scheme *is identified with a triplet* $(\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ *of probabilistic polynomial-time Turing machines, where*

- GEN *is the key generation algorithm that takes as input a security parameter* 1^n *and outputs a pair* $(\mathsf{SK}, \mathsf{VK})$ *such that the sizes of* SK, VK *are polynomially related to* n*.* SK *is referred to as the signing-key and* VK *is referred to as the verification-key.*

- SIGN *is the signing algorithm which takes as input a pair* $(\mathsf{SK}, \mathsf{VK})$ *and a message* M *to be signed, and outputs a signature of* m *with respect to* $(\mathsf{SK}, \mathsf{VK})$*.*

- VERIFY *is the verification algorithm which takes as input a verification-key* VK*, a message* M *and a string* σ *(supposedly a signature of* M *with respect to* VK*), and outputs* 0 *or* 1*. Intuitively, it outputs* 1 *if* σ *is a valid signature of* M *with respect to* VK*, and it outputs* 0 *otherwise.*

Formally, it is required that for any pair $(\mathsf{SK}, \mathsf{VK})$ *in the support of* $\mathsf{GEN}(1^n)$ *and for any message* $\mathsf{M} \in \{0,1\}^*$*,*[3]

$$\Pr[\mathsf{VERIFY}(\mathsf{VK}, \mathsf{M}, \mathsf{SIGN}((\mathsf{SK}, \mathsf{VK}), \mathsf{M})) = 1] = 1$$

(*where the probability is over the random coin tosses of* SIGN *and* VERIFY*).*

[2] This is the case since the existence of one-way functions implies the existence of secure signature schemes [31, 34], which in turn implies the existence of ID schemes which are secure against active attacks (see Section 3.3).

[3] Notice that it is (implicitly) assumed that the message space is $\{0,1\}^*$. This assumption is only for the sake of simplicity.

Security of Signature Schemes. Several types of security requirements were considered in the literature, the strongest which is *existential security against adaptive chosen message attacks*. Throughout this manuscript we say that a signature scheme is secure if it is secure with respect to this strongest security requirement.

Definition 9 (Security against adaptive chosen message attacks): *We say that a signature scheme* $\mathsf{SIG} = (\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ *is* secure *if for every polynomial-size circuit family* $C = \{C_n\}_{n\in\mathbb{N}}$ *with oracle access to* SIGN*, the probability that on input a uniformly chosen verification-key* VK*, where* $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{GEN}(1^n)$*,* C_n *outputs a pair* $(\mathsf{M}, \mathsf{SIG}_\mathsf{M})$ *such that* $\mathsf{VERIFY}(\mathsf{VK}, \mathsf{M}, \mathsf{SIG}_\mathsf{M}) = 1$ *and such that* M *was not sent by* C_n *as an oracle query to* SIGN*, is negligible* (*where the probability is over* VK *and over the randomness of the oracle* SIGN)*.*

3.1.3 The Fiat-Shamir Paradigm

Definition 10 (The Fiat-Shamir Paradigm): *Given any canonical ID scheme* $\mathsf{ID} = (\mathsf{G}, \mathsf{S}, \mathsf{R})$ *and any hash-function ensemble* $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$*, the* Fiat-Shamir paradigm *transforms* ID *and* $\mathcal{H}$ *into a signature scheme* $\mathsf{SIG}_\mathcal{H} = (\mathsf{GEN}_\mathcal{H}, \mathsf{SIGN}_\mathcal{H}, \mathsf{VERIFY}_\mathcal{H})$*, defined as follows.*

- *The key generation algorithm* $\mathsf{GEN}_\mathcal{H}$*, on input* 1^n*, emulates algorithm* $\mathsf{G}(1^n)$ *to generate a pair* $(\mathsf{SK}, \mathsf{PK})$ *of secret key and public key. It then chooses at random a function* $h \in_R \mathcal{H}_n$*, and outputs* SK *as the signing key and* $\mathsf{VK} = (\mathsf{PK}, h)$ *as the verification key.*

- *The signing algorithm* $\mathsf{SIGN}_\mathcal{H}$*, on input a signing key* SK*, a corresponding verification key* $\mathsf{VK} = (\mathsf{PK}, h)$*, and a message* M*, emulates the sender* S *with respect to* $(\mathsf{SK}, \mathsf{PK})$ *to produce* $(\alpha; \beta; \gamma)$*, where* $\beta = h(\alpha, \mathsf{M})$*. That is,* $\mathsf{SIGN}_\mathcal{H}(\mathsf{SK}, \mathsf{VK}, \mathsf{M})$ *operates as follows.*

  1. *Tosses coins* r *(for* S*) and computes* $\alpha = \mathsf{S}_{(\mathsf{SK},\mathsf{PK})}(r)$*.*
  2. *Computes* $\beta = h(\alpha, \mathsf{M})$*.*
  3. *Computes* $\gamma = \mathsf{S}_{(\mathsf{SK},\mathsf{PK})}(\alpha, \beta, r)$

4. *Outputs* $(\alpha;\beta;\gamma)$ *as a signature of* M.

- *The verification algorithm* $\mathsf{VERIFY}_{\mathcal{H}}$*, on input a verification-key* $\mathsf{VK} = (\mathsf{PK}, h)$*, a message* M *and a triplet* $(\alpha;\beta;\gamma)$ (*which is supposedly a signature of* M)*, accepts if and only if* $\beta = h(\alpha, \mathsf{M})$ *and* $(\alpha;\beta;\gamma) \in \mathsf{ACC}(\mathsf{R}(\mathsf{PK}))$.

Throughout this paper, the Fiat-Shamir paradigm is referred to as the FS paradigm. We denote by $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ the signature scheme obtained by applying the FS paradigm to ID and $\mathcal{H}$. $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ is referred to as a FS signature scheme corresponding to ID. We say that the FS paradigm is *secure* if for every secure canonical ID scheme ID, there exists a hash-function ensemble $\mathcal{H}$ such that $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ is secure. Otherwise, we say that the FS paradigm is *insecure*. We denote by (FS) the case that the FS paradigm is secure and we denote by $\neg(\mathsf{FS})$ the case that the FS paradigm is insecure.

3.2 Road Map

In the remaining of Chapter 3 we focus on proving the following two theorems.

Theorem 11 *If collision resistant hash-function ensembles do not exist and one-way functions do exist then the FS paradigm is insecure.*

Theorem 12 *If collision resistant hash-function ensembles exist then the FS paradigm is insecure.*

From the above two theorems and from the fact that the existence of a collision resistant hash-function ensemble implies the existence of a one-way function, we obtain the following corollary.

Corollary 13 *If one-way functions exist then the FS paradigm is insecure.*

It is well known that if one-way functions do not exist then neither do secure signature schemes. Thus, in a sense our result is unconditional since we get that the FS paradigm is either insecure or useless (i.e., never produces secure signatures, as none exist).

We note that the proof of Theorem 11 is relatively simple and that the main result in Chapter 3 is the proof of Theorem 12. We begin by proving Theorem 11 (Section 3.3), and we then prove Theorem 12 (Sections 3.4-3.7).

3.3 Proof of Theorem 11

In this subsection, we assume that collision resistant hash-function ensembles do not exist and that one-way functions do exist. That is, we assume that for every hash-function ensemble $\mathcal{H} = \{\mathcal{H}_n\}$ there exist infinitely many n's such that on input a random $h \in \mathcal{H}_n$, it is easy to find $\mathsf{M}_1 \neq \mathsf{M}_2$ such that $h(\mathsf{M}_1) = h(\mathsf{M}_2)$. More formally, we assume that for every hash-function ensemble $\mathcal{H} = \{\mathcal{H}_n\}$ there exists a polynomial-size circuit family $C = \{C_n\}$ and a polynomial $p(\cdot)$, such for infinitely many n's,

$$\Pr[C_n(h) = (\mathsf{M}_1, \mathsf{M}_2) \ : \ h(\mathsf{M}_1) = h(\mathsf{M}_2) \ \wedge \ \mathsf{M}_1 \neq \mathsf{M}_2] \geq \frac{1}{p(n)}$$

(where the probability is over $h \in_R \mathcal{H}_n$). For every $\mathcal{H}$, we denote the set of all such n's by $S_{\mathcal{H}}$.

Our goal is to construct a secure canonical ID scheme ID such that for every $\mathcal{H}$, the corresponding signature scheme $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}) = (\mathsf{GEN}_{\mathcal{H}}, \mathsf{SIGN}_{\mathcal{H}}, \mathsf{VERIFY}_{\mathcal{H}})$ is insecure. More specifically, we demonstrate the insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ by constructing a forger that for every $n \in \mathsf{S}_{\mathcal{H}}$ succeeds in forging signatures, with respect to $\mathsf{VK} = (\mathsf{PK}, h)$ generated by $\mathsf{GEN}_{\mathcal{H}}(1^n)$, with non-negligible probability.

Intuitively, ID is defined as follows. Fix any secure signature scheme $\mathsf{SIG} = (\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ (the existence of secure signature schemes follows from the existence of one-way functions [31, 34]). The sender will identify himself by signing a random message sent by the receiver.[4] The security of ID will follow from the security of SIG. The insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ will follow from the assumption that collision

[4] In some sense this is the inversion of the Fiat-Shamir paradigm, which starts with an ID scheme and converts it into a signature scheme. Here, we start with a signature scheme and use it to construct an ID scheme.

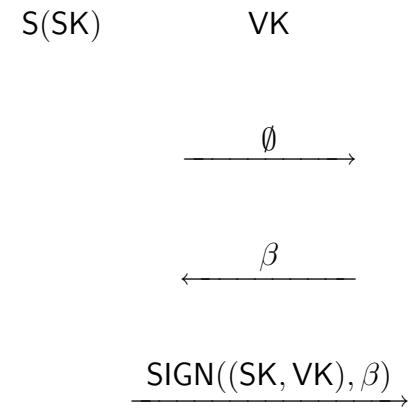
resistant hash-function ensembles do not exist.

In what follows we give a formal proof of Theorem 1.

Proof: Let $\mathsf{SIG} = (\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ be any secure signature scheme. Consider the following ID scheme, $\mathsf{ID} = (\mathsf{G}, \mathsf{S}, \mathsf{R})$.

- G: On input 1^n, emulate $\mathsf{GEN}(1^n)$ to obtain a pair $(\mathsf{SK}, \mathsf{VK})$, and output SK as the secret-key and VK as the public-key.

- S and R are interactive probabilistic polynomial-time Turing machines, that for any $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{G}(1^n)$, the interaction of $(\mathsf{S}(\mathsf{SK}), \mathsf{R})(\mathsf{VK})$ is as follows.



  $\mathsf{R}(\mathsf{VK})$ accepts a transcript $(\alpha; \beta; \gamma)$ if and only if $\alpha = \emptyset$ and $\mathsf{VERIFY}(\mathsf{VK}, \beta, \gamma) = 1$ (i.e., γ is a valid signature of β, with respect to the verification-key VK).

Claim 14 ID *is secure, assuming* SIG *is a secure signature scheme.*

Proof: Follows immediately from the definition of a secure signature scheme (which corresponds to security against adaptive chosen message attacks). ■

Claim 15 *The signature scheme* $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}) = (\mathsf{GEN}_{\mathcal{H}}, \mathsf{SIGN}_{\mathcal{H}}, \mathsf{VERIFY}_{\mathcal{H}})$ *is insecure for every hash-function ensemble* $\mathcal{H}$*, assuming collision resistant hash-function ensembles do not exist.*

Proof: Fix any hash-function ensemble $\mathcal{H}$. Our assumption that collision resistant hash-function ensembles do not exist implies that there exists a circuit family $C = \{C_n\}$, a polynomial $p(\cdot)$, and an infinite set $S_{\mathcal{H}}$, such that for every $n \in S_{\mathcal{H}}$,

$$\Pr[C_n(h) = (\mathsf{M}_1, \mathsf{M}_2) \ : \ h(\mathsf{M}_1) = h(\mathsf{M}_2) \ \wedge \ \mathsf{M}_1 \neq \mathsf{M}_2] \geq \frac{1}{p(n)}$$

(where the probability is over $h \in_R \mathcal{H}_n$). The forger will use this circuit family $C = \{C_n\}$. Given a verification-key $(\mathsf{VK}, h) \leftarrow \mathsf{GEN}_{\mathcal{H}}(1^n)$, where $n \in S_{\mathcal{H}}$, and given a signing oracle, the forger will forge a signature of some new message M, as follows.

1. Compute $(\mathsf{M}_1, \mathsf{M}_2) = C_n(h)$. From our assumption, with probability at least $\frac{1}{p(n)}$ it holds that $h(\mathsf{M}_1) = h(\mathsf{M}_2)$ and $\mathsf{M}_1 \neq \mathsf{M}_2$.
2. Query the signing oracle with the message M_1, to obtain a signature $(\alpha; \beta; \gamma)$.
3. Output $(\alpha; \beta; \gamma)$ as a signature of M_2.

Notice that $(\alpha; \beta; \gamma)$ is a valid signature of M_2 if it is a valid signature of M_1 and $h(\mathsf{M}_1) = h(\mathsf{M}_2)$. Since both of these conditions are satisfied with non-negligible probability, the forger succeeds in forging a signature of M_2 with non-negligible probability.

■

■

Throughout the remaining of Chapter 3 we assume the existence of a collision resistant hash-function ensemble, which we denote by $\mathcal{F}$. Actually, we restrict our attention to a collision resistant hash-function ensemble from $\{0,1\}^{2n}$ to $\{0,1\}^n$.

3.4 Overview of the Proof of Theorem 12

Recall that our goal is to construct a *secure* canonical ID scheme such that for any hash-function ensemble $\mathcal{H}$, $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ is an *insecure* signature scheme. Our idea towards achieving this goal is the following.

Take any secure canonical ID scheme and extend its verdict function in such a way that the ID scheme remains secure, yet all the corresponding FS signature schemes

become insecure no matter which hash-function ensemble is used. To carry out this idea we need to exploit the difference between the ID scheme and the corresponding FS signature schemes. As we see it, the main difference between the two is that in the ID scheme the receiver's message is totally random, whereas in the corresponding FS signature schemes everyone knows in advance a (deterministic) public function h that computes the "receiver's message."

Thus, our idea is to take any secure ID scheme and to extend its verdict function so as to also accept transcripts which convince the receiver that the sender knew in advance a (deterministic) function that computes the receiver's message. Since the receiver chooses its message at random (by definition of a canonical ID scheme), there is no way that a sender could have guessed in advance such a function, except with negligible probability, and therefore the ID scheme remains secure. However, when the ID scheme is converted into a signature scheme via the FS paradigm, the receiver is replaced with a succinct (deterministic) public function, and thus everyone knows in advance a function that computes the "receiver's message." Thus, a forger can easily convince the verifier of knowledge of this function, which corresponds to a legitimate signature. Hence, all the corresponding FS signature schemes are insecure (no matter which hash-function ensemble is used).

The main problem with this approach is the following: How can the sender convince the receiver that he knew in advance a function that computes the receiver's message? The first idea that comes to mind is for the sender to simply send to the receiver (in the first round) a polynomial-size circuit that computes the receiver's message. The problem with this idea is that we must first fix the ID scheme (in particular, fix a polynomial bound on the size of its messages) and only then show that for *any* hash-function ensemble $\mathcal{H}$ replacing the receiver, $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ is insecure. In other words, we need to find a protocol of a-priori bounded size, in which the sender will be able to convince the receiver of knowledge of *any* polynomial-size circuit corresponding to *any* $\mathcal{H}$. Thus, the sender cannot simply send the verifier his circuit in hand (which may be too big). We overcome this problem by having the sender send

a size-reducing commitment to his circuit, rather than the circuit itself.[5]

To summarize, our idea for proving the insecurity of the FS paradigm is as follows: Start with any secure ID scheme and extend its verdict function so as to also accept views in which the sender first sends message a (supposedly $a = \mathsf{COM}(C)$ is a size-reducing commitment to a circuit C), the receiver replies with b, and only then the sender proves to the receiver that he knows a circuit C, such that both $\mathsf{COM}(C) = a$ and $C(a) = b$. More precisely, after receiving the message b, the sender proves that he knows a circuit C, which is a witness to (a, b) in the following relation:

$$\mathcal{R} = \{((a, b), C) : a = \mathsf{COM}(C) \ \wedge \ C(a) = b\}.$$

Note that if $\mathcal{R}$ was an $\mathcal{NP}$-relation, then we would be done, as the sender could then prove knowledge of a witness C by simply revealing C. However, if we bound $\mathcal{R}$ to be an $\mathcal{NP}$-relation by bounding all the witnesses to be of size at most $p(n)$ (for some fixed polynomial $p(\cdot)$), then we could *only* prove that the corresponding FS signature schemes with hash-function ensembles of size at most $p(\cdot)$ are insecure, and we would not know if *all* the corresponding FS signature schemes (with respect to *all* hash-function ensembles) are insecure. Thus, instead we bound the witnesses of $\mathcal{R}$ by some super-polynomial function (say $n^{\log n}$), and define $\mathcal{R}$ as follows:

$$\mathcal{R} = \{((a, b), C) : a = \mathsf{COM}(C) \ \wedge \ C(a) = b \ \wedge \ |C| \leq n^{\log n}\},$$

which results with $\mathcal{R}$ being an $\mathsf{NTIME}(n^{\log n})$ relation.

This brings about a new issue that needs to be resolved: In the above extended ID scheme, in the third round the sender needs to prove knowledge of a witness for (a, b) in the relation $\mathcal{R}$, where a is the message sent by the sender in the first round and b is the message sent by the receiver in the second round. This requires a proof-

[5]Note that all known constructions of size-reducing commitment schemes that rely on the existence of a collision resistant hash-function ensemble consist of two rounds. We ignore this issue here, and treat the commitment scheme as a one-round commitment scheme (eventually, the first round of the commitment scheme will be appended to the public key). This issue will be elaborated on in Section 3.5.

of-knowledge system for $\mathcal{R}$ which consists of either one round or two rounds in which the first round consists of the verifier's random coin tosses. Unfortunately, we do not know if such a proof-of-knowledge system for $\mathcal{R}$ exists, as $\mathcal{R}$ is not an $\mathcal{NP}$-relation.

To summarize so far, if there somehow existed a 2-round public-coin proof-of-knowledge system for $\mathcal{R}$ then we would be done, since we could take the secure canonical ID scheme, and extend its verdict function so as to also accept transcripts of the form

$$\xrightarrow{\quad a \quad}$$

$$\xleftarrow{\quad b, q \quad}$$

$$\xrightarrow{\quad ans \quad}$$

where $(q; ans)$ is a 2-round public-coin proof-of-knowledge of C such that $((a, b), C) \in \mathcal{R}$.

Thus, our next attempt is to try to construct a 2-round public-coin proof-of-knowledge system for $\mathcal{R}$. Barak and Goldreich [3], based on the works of [24, 27], presented a 4-round public-coin argument, called *universal argument*, for every language in NEXP, and in particular for $\mathcal{R}$. Our approach is to apply the FS paradigm to this universal argument system to obtain a 2-round system, which we call a *2-round universal system*. This seems like a strange approach, since our goal in this paper is to prove the insecurity of the FS paradigm. In particular, the resulting 2-round universal system may not be a proof-of-knowledge and may not even be sound. Nevertheless, it will take us one step further in the proof.

Thus, in the above ID scheme we use a 2-round universal system obtained by applying the FS paradigm to the universal argument system of [3]. Note that the 2-round universal system, and hence the above ID scheme, depend on the hash-function ensemble used when applying the FS paradigm to the 4-round universal argument system. For any hash-function ensemble $\mathcal{G}$, we denote by $\mathsf{ID}^1_{\mathcal{G}}$ the above ID scheme, where the 2-round universal system is obtained by applying the FS paradigm with respect to the hash-function ensemble $\mathcal{G}$.

Note, that since we do not know whether the FS paradigm is secure, we do not know whether there exists a hash-function ensemble $\mathcal{G}$ such that $\mathsf{ID}^1_{\mathcal{G}}$ is secure. However, we do know that for every $\mathcal{G}$, all the FS signature schemes corresponding to $\mathsf{ID}^1_{\mathcal{G}}$ are insecure. In other words, we know that for every hash-function ensembles $\mathcal{G}$ and $\mathcal{H}$, the signature scheme $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}})$ is insecure. (This follows from the fact that the resulting 2-round universal system is complete, since applying a FS step always preserves the completeness property.)

We proceed by considering the following two cases. Case 1 is the case that there exists a hash-function ensemble $\mathcal{G}$ such that $\mathsf{ID}^1_{\mathcal{G}}$ is secure, in which case we are done, as $\mathsf{ID}^1_{\mathcal{G}}$ demonstrates the insecurity of the FS paradigm. Case 2 is the case that $\mathsf{ID}^1_{\mathcal{G}}$ is insecure for every hash-function ensemble $\mathcal{G}$. In this case we construct another ID scheme, called ID^2 that demonstrates the insecurity of the FS paradigm.

The idea behind the construction of ID^2 is the following: Note that the assumption that $\mathsf{ID}^1_{\mathcal{G}}$ is insecure implies that there exits an impersonator S^* that for infinitely many n's finds an a such that for a random b it can convince the 2-round universal system (obtained by applying the FS paradigm with respect to $\mathcal{G}$) that it "knows" a witness for $(a,b) \in L_{\mathcal{R}}$ with non-negligible probability (where $L_{\mathcal{R}}$ denotes the language corresponding to the relation $\mathcal{R}$). This implies that there exits S^* such that for infinitely many n's, finds a, b_1, b_2 such that it can convince this 2-round universal system that it "knows" a witness for both $(a,b_1) \in L_{\mathcal{R}}$ and $(a,b_2) \in L_{\mathcal{R}}$. In contrast, it is hard for any S^* to convince the 4-round universal argument that it knows a witness for both $(a,b_1) \in L_{\mathcal{R}}$ and $(a,b_2) \in L_{\mathcal{R}}$, since the 4-round universal argument system is a proof-of-knowledge system, and knowledge of a witness for both (a,b_1) and (a,b_2) implies knowledge of $C_1 \neq C_2$ such that $\mathsf{COM}(C_1) = \mathsf{COM}(C_2)$, contradicting the binding property of the commitment scheme.[6] This contrast between the 2-round universal system and the 4-round universal system, suggests constructing the

[6]Since C_1 and C_2 are not a priori bounded by any polynomial, it seems like we need super-polynomial hardness assumptions in order to contradict the binding property of the commitment scheme. We eliminate the need of a super-polynomial hardness assumption, by using a tree-commitment scheme rather than a regular commitment scheme. The notion of tree-commitment was introduced by Merkle [26] and has the advantageous property that it allows decommitment to individual bits. We shall elaborate on this in Section 3.5.

following ID scheme, denoted by ID^2.

From now on we denote an accepting transcript of the 4-round universal system by $(\alpha;\beta;\gamma;\delta)$, where α and γ are random strings. ID^2 is defined by taking any secure ID scheme and extending its public-key by appending random α_1, α_2 to it, and extending its verdict function so as to also accept transcripts of the form

$$\xrightarrow{a, b_1, b_2, \beta_1, \beta_2}$$

$$\xleftarrow{\gamma_1, \gamma_2}$$

$$\xrightarrow{\delta_1, \delta_2}$$

where $(\alpha_1;\beta_1;\gamma_1;\delta_1)$ is an accepting transcript of the 4-round universal system for $(a, b_1) \in L_{\mathcal{R}}$ and $(\alpha_2;\beta_2;\gamma_2;\delta_2)$ is an accepting transcript of the 4-round universal system for $(a, b_2) \in L_{\mathcal{R}}$.

The security of ID^2 follows from the fact that the 4-round universal system is a proof of knowledge and from the binding property of the commitment scheme.[7] Intuitively, the insecurity of the FS signature schemes corresponding to ID^2 seems to follow from our assumption that $\mathsf{ID}^1_{\mathcal{G}}$ is insecure for every hash-function ensemble $\mathcal{G}$. The reason being that for every hash-function ensemble $\mathcal{H}$, in order to forge a signature in the signature scheme $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$, it suffices to find a, b_1, b_2 and to convince the 2-round universal system that it "knows" a witness for both $(a, b_1) \in L_{\mathcal{R}}$ and $(a, b_2) \in L_{\mathcal{R}}$, where the first 2-round universal system is with respect to $\mathcal{H}^1$ (which outputs the $|\gamma_1|$ most significant bits of $\mathcal{H}$) and the second is with respect to $\mathcal{H}^2$ (which outputs the $|\gamma_2|$ least significant bits of $\mathcal{H}$). Thus it seems that by taking $\mathcal{G} = \mathcal{H}^1 \cup \mathcal{H}^2$, an impersonator for $\mathsf{ID}^1_{\mathcal{G}}$ can be used as a forger for $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$. However, this is not quite so, since in order to forge a signature in $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$ one needs to generate two

[7] As previously mentioned, the commitment scheme used will be a tree-commitment scheme (to be defined in Section 3.5), which is based on the existence of a collision resistant hash-function ensemble. We will get a contradiction to the collision resistant property by using the knowledge extractor of the universal system to find $2n$ bits of C_1 (which is a witness for (a, b_1)) and $2n$ bits of C_2 (which is a witness for (a, b_2)) that form a collision.

accepting transcripts for the two-round universal system (one for $(a, b_1) \in L_{\mathcal{R}}$ and one for $(a, b_2) \in L_{\mathcal{R}}$) that *depend* on one another (i.e., γ_1 depends on both β_1 and β_2, and same holds for γ_2), whereas in order to impersonate the sender in $\mathsf{ID}^1_{\mathcal{G}}$ it suffices to generate *independent* accepting transcripts.

In order to overcome this problem we construct yet another (and final!) ID scheme, denoted by ID^3. We prove that the insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^3)$ (for every $\mathcal{H}$) follows from the insecurity of $\mathsf{ID}^1_{\mathcal{H}}$ (for every $\mathcal{H}$). Moreover, we prove that either ID^3 is secure or $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$ is insecure for every $\mathcal{H}$. Thus, either ID^2 or ID^3 demonstrate the insecurity of the FS paradigm.

This concludes the overview of the proof of Theorem 12. We warn the reader that the actual proof contains several technical difficulties that were omitted from the overview. In the remaining of Chapter 3 we give a more formal presentation of the proof. In Section 3.5 we formally define the relation $\mathcal{R}$, which from now on will be called *the central relation* and will be denoted by $\mathcal{R}_{\mathcal{F}}$ (as it depends on a collision resistant hash-function ensemble $\mathcal{F}$). In Section 3.6 we define our 2-round universal system for $\mathcal{R}_{\mathcal{F}}$. In Section 3.7 we present in more detail the constructions of $\mathsf{ID}^1_{\mathcal{G}}$, ID^2, and ID^3, and prove that one of them demonstrates the insecurity of the FS paradigm, assuming that $\mathcal{F}$ is a collision resistant hash-function ensemble.

3.5 Central Relation $\mathcal{R}_{\mathcal{F}}$

Recall that in Section 3.4 we informally defined the relation $\mathcal{R}$ as follows:

$$\mathcal{R} = \{((a, b), C) : a = \mathsf{COM}(C) \ \wedge \ C(a) = b \ \wedge \ |C| \leq n^{\log n}\}.$$

As we mentioned, COM is a size-reducing commitment scheme. More specifically (as was mentioned in footnote 11), the type of commitment we use is a tree-commitment, which not only allows a fixed polynomial-size commitment for any polynomial-size string, but also has the advantageous property that it allows decommitment to individual bits. The notion of *tree-commitment* was introduced by Merkle [26].

Definition 16 (Tree-Commitment): *A tree-commitment to $x \in \{0,1\}^*$, with respect to the function $f : \{0,1\}^{2n} \to \{0,1\}^n$, is defined as follows. Consider a complete binary tree of depth $\lg(|x|/n)$, where each node has a label in $\{0,1\}^n$. The leaves are labeled by the bits of x (n bits per leaf). Each internal node is labeled by applying f to the label of its children. The tree-commitment to x, with respect to f, is denoted by $\mathsf{TC}_f(x)$, and consists of the label of the root and the depth of the tree.*[8]

We let $auth_f(x,i)$ denote the authentication path of the ith bit of x with respect to f. Namely, $auth_f(x,i)$ consists of the label of the leaf corresponding to x_i, the label its sibling, the labels of its ancestors and the labels of its ancestors siblings. We let $auth_f(x)$ denote the entire tree, which contains the authentication path of x_i, for every i.

We are now ready to define the central relation more formally. Let $\mathcal{F}$ be a collision resistant hash-function ensemble.

Definition 17 (Central Relation):

$$\mathcal{R}_{\mathcal{F}} = \{((f,a,b),\hat{C}) \ : \ \mathsf{TC}_f(\hat{C}) = a \ \wedge \ C(a) = b \ \wedge \ \ |\hat{C}| < n^{\lg n}\}$$

where $C \to \hat{C}$ is a special circuit-encoding which satisfies the following properties.

1. *It is an efficient encoding. Namely, there is a polynomial-time algorithm that given any circuit C, outputs $\hat{C}$.*

2. *It has high minimum distance. Namely, for every $C_1 \neq C_2$, $\hat{C}_1$ and $\hat{C}_2$ differ in a polynomial fraction of their coordinates.*

3. *Given y, it is easy to check whether y is a codeword. Namely, there is a polynomial-time algorithm that given y, outputs 1 if and only if there exists a circuit C such that $y = \hat{C}$.*

[8] Note that if f is chosen at random from a collision resistant hash-function ensemble then the tree-commitment with respect to f is computationally binding.

4. *There exists a polynomial-time algorithm that given any circuit-encoding $\hat{C}$ (where C is defined on inputs of size n) and given any $x \in \{0,1\}^n$, computes $C(x)$.*

Remarks:

1. For technical reasons to be clarified in the proof of Lemma 3.7.4, rather than having merely $\hat{C}$ as a witness, we actually include the entire tree $auth_f(\hat{C})$ in the witness. So $\mathcal{R}_{\mathcal{F}}$ will actually be defined by:

$$\mathcal{R}_{\mathcal{F}} \triangleq \{((f,a,b), auth_f(\hat{C})) \ : \ \mathsf{TC}_f(\hat{C}) = a \ \wedge \ C(a) = b \ \wedge \ \ |\hat{C}| < n^{\lg n}\}$$

2. The reason we bound the size of $\hat{C}$ by $n^{\lg n}$ is because the function that replaces the receiver in the FS paradigm can be of *any* polynomial-size. Hence, we cannot bound the size of $\hat{C}$ by a fixed polynomial, and so we bound it by some super-polynomial, such as $n^{\lg n}$.

3. We defined $\mathcal{R}_{\mathcal{F}}$ using a tree-commitment, as opposed to a regular length-reducing commitment, for the following technical reason. In our proof we get a contradiction to the security of the Fiat-Shamir paradigm, by claiming knowledge of $\hat{C}_1 \neq \hat{C}_2$ which commit to the same value. However, the size of these circuits is not a-priori bounded by some polynomial, and hence we cannot extract this knowledge using a polynomial-time algorithm. We get around this technical problem by using a tree-commitment, which allows decommitment to individual bits.

4. Property 1 of the encoding is needed in the proof of Lemma 3.7.1. Property 2 of the encoding is needed in the proof of Lemma 3.7.4. Properties 3 and 4 of the encoding are needed in the proof of Proposition 1.

5. Without loss of generality, we assume that $auth_f(\hat{C})$ is of the following form: After every bit of $\hat{C}$ there are exactly $(\lg n)^2$ bits of the authentication path of that bit. Namely, we assume that the i'th bit of $\hat{C}$ is represented in the $(1+(i-$

1)((lg n)^2 + 1))'th bit of $auth_f(\hat{C})$, followed by (lg n)^2 bits of its authentication path. We will need this precision of representation in proving Lemma 3.7.4.

6. Sometimes we refer to a witness of (f, a, b) by $w_{(f,a,b)}$ or simply by w.

7. For simplicity, we assume that for every $((f, a, b), w) \in \mathcal{R}_{\mathcal{F}}$ such that $f \in \mathcal{F}_n$, it holds that $|a| = |b| = n$. We are being imprecise here, since actually $|a| > n$. We assume $|a| = n$ only in order to simplify notations.

Protocol 1 *[3]:* $L_{\mathcal{R}_{\mathcal{F}}} \in \mathsf{NTIME}(n^{\lg n})$.

Proof: Follows immediately from the definition of $\mathcal{R}_{\mathcal{F}}$ and from properties 3 and 4 of the circuit-encoding $C \to \hat{C}$. ■

3.6 Interactive Arguments for $\mathcal{R}_{\mathcal{F}}$

In this subsection we try to find a 2-round proof-of-knowledge for $\mathcal{R}_{\mathcal{F}}$. From the theory on Probabilistic-Checkable-Proofs it follows that for every relation $\mathcal{R}$ in NEXP there exists a polynomial-time Turing machine $\mathsf{P}_{\mathsf{PCP}}$ and a probabilistic polynomial-time oracle machine $\mathsf{V}_{\mathsf{PCP}}$ with the following properties.

1. (*Relatively-efficient oracle construction*)*:* for every $(x, w) \in \mathcal{R}$, $\mathsf{P}_{\mathsf{PCP}}(x, w) = \pi$ such that $\Pr[\mathsf{V}^{\pi}_{\mathsf{PCP}}(x) = 1] = 1$. Throughout the paper, we refer to π as a PCP proof.

2. (*Non-adaptive verifier*)*:* The verifier's queries are determined based only on its input and on its internal coin tosses. That is, there exists a probabilistic polynomial-time algorithm Q_{PCP} such that on input x and random coins r, the verifier makes the query sequence $\{q_i\}$, where for every i, $q_i = Q_{\mathsf{PCP}}(x, r, i)$.[9]

3. (*Efficient reverse-sampling*)*:* There exists a probabilistic polynomial-time oracle machine S such that, on input any string x and integers i and q, outputs a uniformly distributed r that satisfies $Q_{\mathsf{PCP}}(x, r, i) = q$.

[9]Throughout this paper, we let $Q_{\mathsf{PCP}}(x, r)$ denote the query sequence of $\mathsf{V}_{\mathsf{PCP}}$ on input x and random tape r.

4. (*Proof-of-knowledge*): There exists a probabilistic polynomial-time oracle machine E and a negligible function $\epsilon : \mathbb{N} \to [0,1]$ such that for every x, and for every π, if $\Pr[\mathsf{V}^{\pi}_{\mathsf{PCP}}(x) = 1] > \epsilon(|x|)$ then there exists w, such that $(x,w) \in \mathcal{R}$ and for every i, $\Pr[E^{\pi}(x,i) = w_i] \geq 2/3$.

Based on the above theory of PCP and based on the works of [24, 27], Barak and Goldreich [3] presented a 4-round public-coin argument, called *universal argument*, for every language in NEXP, and in particular for $\mathcal{R}_{\mathcal{F}}$.

We begin by presenting this 4-round universal argument for $\mathcal{R}_{\mathcal{F}}$. Then we reduce interaction by applying a Fiat-Shamir type step, this time in the context of universal arguments. This seems like a strange idea, since our goal is to prove the insecurity of the FS paradigm, but it will take us one step further in the proof.

3.6.1 First Interactive Argument [3]: $(\mathsf{P}^0, \mathsf{V}^0)$

- Common input: (f, a, b)
- Auxiliary input to the prover: w such that supposedly $((f,a,b),w) \in \mathcal{R}_{\mathcal{F}}$.

1. V^0: Uniformly select $f^{\mathsf{UA}} \in_R \mathcal{F}_n$ and send it to the prover.
2. P^0:
   (a) Construct a PCP proof of $((f,a,b),w)$ by computing $\pi = \mathsf{P}_{\mathsf{PCP}}((f,a,b),w)$.
   (b) Compute $\beta = \mathsf{TC}_{f^{\mathsf{UA}}}(\pi)$,[10] which is the tree-commitment to π with respect to f^{UA}.
   (c) Send β to the verifier.
3. V^0: Uniformly select a random-tape γ for $\mathsf{V}_{\mathsf{PCP}}$, and send γ to the prover.

[10] Note that there are two levels of use of the tree-commitment:
i. In the definition of $\mathcal{R}_{\mathcal{F}}$: $\mathsf{TC}_f(w) = a$.
ii. In the interactive argument for $\mathcal{R}_{\mathcal{F}}$: $\mathsf{TC}_{f^{\mathsf{UA}}}(\pi) = \beta$.
In both cases we use a tree-commitment since the size of both w and π may be to large to extract. Using a tree-commitment we can extract only a few coordinates, with the ability to verify that these values were committed to.

4. P^0: Provide the answers to the (PCP) queries of $\mathsf{V}_{\mathsf{PCP}}((f,a,b),\gamma)$ augmented by proofs of consistency to these answers, as follows.

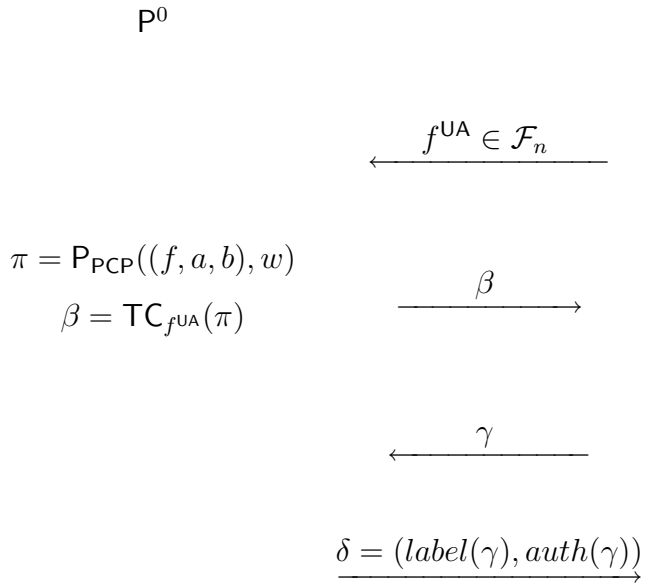
   (a) Determining the queries: Invoke $Q_{\mathsf{PCP}}((f,a,b),\gamma)$, in order to determine the sequence of queries that $\mathsf{V}_{\mathsf{PCP}}$ makes on input (f,a,b), given a random string γ.

   (b) For each query q_i of $Q_{\mathsf{PCP}}((f,a,b),\gamma)$, send the label of the leaf that contains π_{q_i} and send the labels of the path corresponding to this leaf, which consists of the label of its sibling, the labels of its ancestors and the labels of its ancestors siblings, which are needed in order to verify consistency with β.

   We denote this response by $\delta = (label(\gamma), auth(\gamma))$.

V^0 accepts if and only if the answers provided by the prover would have been accepted by $\mathsf{V}_{\mathsf{PCP}}$, and all the proofs of consistency are valid.

$(\mathsf{P}^0,\mathsf{V}^0)$, on input (f,a,b), can be schematically viewed as follows.



Lemma 3.6.1 *[27],[3]:* $(\mathsf{P}^0, \mathsf{V}^0)$ *satisfies the following properties.*

- Completeness: *For every* $((f,a,b),w) \in \mathcal{R}_{\mathcal{F}}$,

$$\Pr[(\mathsf{P}^0(w), \mathsf{V}^0)(f,a,b) = 1] = 1$$

(*where the probability is over the random coin tosses of* V^0).

- CS-proof-of-knowledge: *For every polynomial* $p(\cdot)$ *there exists a polynomial* $p'(\cdot)$ *and a probabilistic polynomial-time oracle machine* E *such that for every polynomial-size circuit family* $\tilde{\mathsf{P}} = \{\tilde{\mathsf{P}}_n\}$, *for every sufficiently large* n, *and for every input* (f,a,b) *such that* $f \in \mathcal{F}_n$ *and* $|a| = |b| = n$, *if*

$$\Pr[(\tilde{\mathsf{P}}_n, \mathsf{V}^0)(f,a,b) = 1] \geq 1/p(n)$$

(*where the probability is over the random coin tosses of* V^0), *then*

$$\Pr[\exists w \text{ } s.t. \text{ } ((f,a,b),w) \in \mathcal{R}_{\mathcal{F}} \text{ } and \text{ } \forall i \text{ } E^{\tilde{\mathsf{P}}_n}((f,a,b),i) = w_i] \geq 1/p'(n)$$

(*where the probability is over the random coin tosses of* E).

Remarks:

1. We will not prove this Lemma since it was proved in [3] (using the four properties of $(\mathsf{P}_{\mathsf{PCP}}, \mathsf{V}_{\mathsf{PCP}})$). However, we would like to stress that following the proof in [3], it can be easily seen that the above proof-of-knowledge property holds even if $\tilde{\mathsf{P}}_n$ chooses (f,a,b) after receiving the verifier's first message f^{UA}.

2. We assume for simplicity that $|\beta| = |\gamma| = n$. Note that we are being imprecise by assuming $|\beta| = n$, though this is done only for the ease of notations.

3.6.2 Reduced-Interaction Argument: $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$

Next, we reduce the number of rounds in $(\mathsf{P}^0, \mathsf{V}^0)$ by applying a Fiat-Shamir type step to it. Namely, we replace V^0's second message with some function applied to

P^0's first message. For technical reasons to be clarified later, we append randomness to the prover's first message.

For any hash-function ensemble $\mathcal{G}$, we define a reduced-interaction argument $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ for $\mathcal{R}_{\mathcal{F}}$, as follows.

- Common input: (f, a, b).
- Auxiliary input to the prover: w such that supposedly $((f, a, b), w) \in \mathcal{R}_{\mathcal{F}}$.

1. $\mathsf{V}^{\mathcal{G}}$: Uniformly select
   - $f^{\mathsf{UA}} \in \mathcal{F}_n$ (a function for the tree-commitment)
   - $g \in \mathcal{G}_n$
   - $r \in \{0,1\}^t$ (we will actually set $t = 4n$).

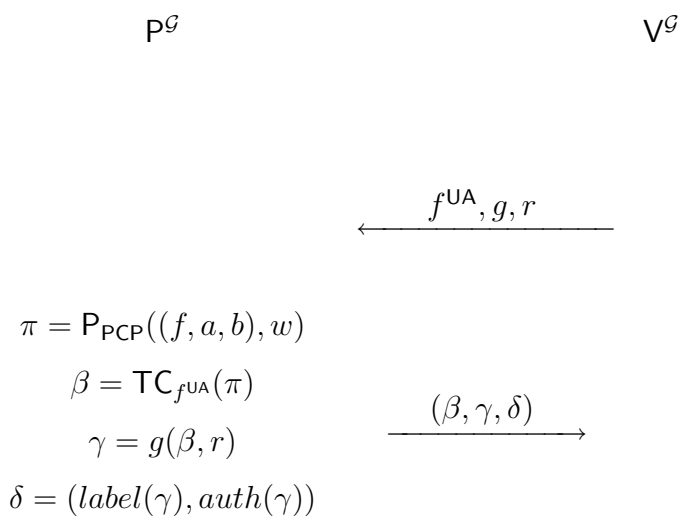
   Send (f^{UA}, g, r) to the prover.
2. $\mathsf{P}^{\mathcal{G}}$:
   (a) Invoke $\mathsf{P}_{\mathsf{PCP}}$ on $((f, a, b), w)$ to obtain $\pi = \mathsf{P}_{\mathsf{PCP}}((f, a, b), w)$.
   (b) Compute $\beta = \mathsf{TC}_{f^{\mathsf{UA}}}(\pi)$.
   (c) compute $\gamma = g(\beta, r)$.
   (d) Let δ be the (PCP) answers corresponding to the queries $Q_{\mathsf{PCP}}((f, a, b), \gamma)$ augmented by proofs of consistency to these answers.

   send (β, γ, δ).

$\mathsf{V}^{\mathcal{G}}$ accepts if and only if the following conditions hold.

1. $\gamma = g(\beta, r)$.
2. $(f^{\mathsf{UA}}; \beta; \gamma; \delta) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b))$.

$(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$, on input (f, a, b), can be schematically viewed as follows.



3.6.3 Another Reduced-Interaction Argument: $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$

For technical reasons to be clarified later, we define an additional reduced-interaction argument for $\mathcal{R}_{\mathcal{F}}$, which is essentially a parallel repetition of $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$. For any two hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$, we define $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$, as follows.

- Common input: (f, a, b).
- Auxiliary input to the prover: w such that supposedly $((f, a, b), w) \in \mathcal{R}_{\mathcal{F}}$.

1. $\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}$: Uniformly select
   - $f^{\mathsf{UA}} \in \mathcal{F}_n$ (a function for the tree-commitment)
   - $g_1 \in \mathcal{G}^1_n$
   - $g_2 \in \mathcal{G}^2_n$
   - $r_1, r_2 \in \{0,1\}^t$ (where $t = 4n$).

   Send $(f^{\mathsf{UA}}, g_1, g_2, r_1, r_2)$ to the prover.
2. $\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}$: For $i = 1, 2$,

(a) Invoke $\mathsf{P}_{\mathsf{PCP}}$ on $((f,a,b),w)$ to obtain $\pi_i = \mathsf{P}_{\mathsf{PCP}}((f,a,b),w)$.[11]

(b) Compute $\beta_i = \mathsf{TC}_{f^{\mathsf{UA}}}(\pi_i)$.[12]

(c) compute $\gamma_i = g_i(\beta_i, r_i)$.

(d) Let δ_i be the (PCP) answers corresponding to the queries $Q_{\mathsf{PCP}}((f,a,b),\gamma_i)$ augmented by proofs of consistency to these answers.

send $\{\beta_i, \gamma_i, \delta_i\}_{i=1,2}$.

$\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}$ accept if and only if the following conditions hold for $i = 1, 2$.

1. $\gamma_i = g_i(\beta_i, r_i)$.
2. $(f^{\mathsf{UA}}; \beta_i; \gamma_i; \delta_i) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b))$.

$(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$, on input (f,a,b), can be schematically viewed as follows.

$\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}$		$\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}$
	$\xleftarrow{\quad f^{\mathsf{UA}}, g_1, g_2, r_1, r_2 \quad}$	
$\pi_i = \mathsf{P}_{\mathsf{PCP}}((f,a,b),w)$ $\beta_i = \mathsf{TC}_{f^{\mathsf{UA}}}(\pi_i)$ $\gamma_i = g_i(\beta_i, r_i)$ $\delta_i = (label(\gamma_i), auth(\gamma_i))$	$\xrightarrow{\quad \{\beta_i, \gamma_i, \delta_i\}_{i=1,2} \quad}$	

We introduce some notation which will be useful later. We typically let q denote the message sent by $\mathsf{V}^{\mathcal{G}}$ or by $\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}$, and we let ans denote the response of $\mathsf{P}^{\mathcal{G}}$ or of $\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}$.

[11] Note that since $\mathsf{P}_{\mathsf{PCP}}$ is deterministic, $\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}$ will obtain $\pi_1 = \pi_2$.
[12] Note that $\mathsf{P}^{\mathcal{G}_1,\mathcal{G}_2}$ will obtain $\beta_1 = \beta_2$.

It is easy to see that both $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ and $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$ satisfy the completeness requirement. However, we do not know whether they satisfy the CS-proof-of-knowledge property. Notice that if $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ satisfies the CS-proof-of-knowledge property, then so does $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$. Thus, all the arguments in the following subsection also apply to $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$.

3.6.4 $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ and CS-Proofs

The proof system $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ is closely related to CS-proofs, defined by Micali [27], since CS-proofs are essentially a non-interactive version of $(\mathsf{P}^0, \mathsf{V}^0)$ obtained by replacing the verifier V^0 with a random oracle. Micali proved that, in the Random Oracle Model, CS-proofs satisfy both the completeness property and the CS-proof-of-knowledge property.[13] One can make the following hypothesis.

Hypothesis (CSP). *There exists a function ensemble $\mathcal{G}$ such that if the random oracle is replaced with a function uniformly chosen from $\mathcal{G}$, then CS-proofs still satisfy both the completeness property and the CS-proof-of-knowledge property.*

Looking carefully into the definition of CS-Proofs one can easily verify the following.

Protocol 2 *The CSP hypothesis implies that there exists a function ensemble $\mathcal{G}$ for which $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ satisfies both the completeness property and the CS-proof-of-knowledge property.*

This is quite surprising, since it essentially implies that if CS-proofs exist in the real world (i.e., if the CSP Hypothesis holds), then the FS paradigm is insecure. In other words, if the FS paradigm applied to $(\mathsf{P}^0, \mathsf{V}^0)$ results with a secure scheme, then the FS paradigm applied to canonical ID schemes results with insecure schemes. This was intuitively argued in Section 3.4 and will be formally shown in Section 3.7.1 (via the construction of $\mathsf{ID}^1_{\mathcal{G}}$).

[13] The definitions of completeness and of CS-proof-of-knowledge were given in Lemma 3.6.1.

It turns out that the bulk of complication is in showing that if the CSP hypothesis is false then still the FS paradigm is insecure. Actually, we don't know how to use the assumption that the CSP hypothesis is false in order to prove the insecurity of the FS paradigm, and we prove it directly.

3.7 Proof of Theorem 12

Our goal is to construct a secure canonical ID scheme ID such that for any hash-function ensemble $\mathcal{H}$, $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID})$ is an insecure signature scheme. In fact we cannot point to one explicit construction of such an ID scheme. Instead, we show three explicit constructions of ID schemes: ID^1, ID^2, ID^3, and prove that the FS paradigm must be insecure with respect to one of the three. The constructions of these ID schemes is based on the intuition given in Section 3.4.

3.7.1 Construction of ID^1

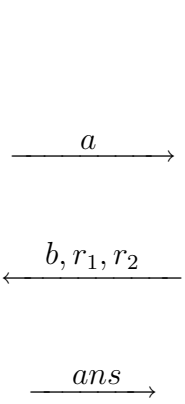
Let $\mathcal{F}$ be a collision resistant hash-function ensemble, let $\mathcal{G}^1$ and $\mathcal{G}^2$ be some a-priori fixed hash-function ensembles, and let $\mathsf{ID} = (\mathsf{G}, \mathsf{S}, \mathsf{R})$ be any secure canonical ID scheme. We extend ID to obtain a new ID scheme $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2} = (\mathsf{G}^1, \mathsf{S}^1, \mathsf{R}^1)$, by extending the public-key and the verdict function of ID, as follows.

- G^1: on input 1^n,

  1. Run $\mathsf{G}(1^n)$, to obtain a pair $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{G}(1^n)$.
  2. Choose randomly $f \in_R \mathcal{F}_n$.
  3. Choose randomly $g_1 \in_R \mathcal{G}^1_n$ and $g_2 \in_R \mathcal{G}^2_n$.

  Output SK as the secret-key and $\mathsf{PK}' = (\mathsf{PK}, f, g_1, g_2)$ as the public-key.

- R^1: On input a public-key $\mathsf{PK}' = (\mathsf{PK}, f, g_1, g_2)$, R^1 will accept either views that $\mathsf{R}(\mathsf{PK})$ accepts or views of the form



such that $(q; ans) \in \mathsf{ACC}(\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}(f, a, b))$, where $q = (f, g_1, g_2, r_1, r_2)$.

Remarks:

1. Notice that the first part of q appears in the public key, whereas the second part of q appears in the verifier's message. The reason we split q into two parts, rather then simply letting all of q be part of the verifier's message, as suggested in the intuition given in Section 3.4, is due to a subtle point made by Hsiao and Reyzin. (Thanks Hsiao and Reyzin!). Their point is that $\mathcal{F}, \mathcal{G}^1, \mathcal{G}^2$ are not necessarily public-coin hash-function ensembles, and therefore cannot be sent by a public-coin verifier. Due to this observation, f, g_1, g_2 are appended to the public key.

2. Notice that f is used for two different purposes. It is used both in the instance (f, a, b) and in the query $q = (f, g_1, g_2, r_1, r_2)$. The reason that we can use the same function, is that both cases require the same property from f, namely it being collision resistant.

To establish $\neg(\mathsf{FS})$, we need to show that the ID scheme $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ is secure and that the signature scheme $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2})$ is insecure with respect to any hash-function ensemble $\mathcal{H}$.

3.7.2 On the Insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2})$

We begin by proving the insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2})$. We denote $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2})$ by $\mathsf{SIG}^1_{\mathcal{G}^1,\mathcal{G}^2,\mathcal{H}} = (\mathsf{GEN}^1_{\mathcal{H}}, \mathsf{SIGN}^1_{\mathcal{H}}, \mathsf{VERIFY}^1_{\mathcal{H}})$.

Lemma 3.7.1 *For any function ensemble $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$, the signature scheme $\mathsf{SIG}^1_{\mathcal{G}^1,\mathcal{G}^2,\mathcal{H}}$ is insecure.*

Intuitively, the insecurity of $\mathsf{SIG}^1_{\mathcal{G}^1,\mathcal{G}^2,\mathcal{H}}$ follows from the fact that the receiver's message is replaced with the value of a succinct (deterministic) function $h \in \mathcal{H}$ (given as part of the public key), applied to the sender's first message. Thus, a forger can easily convince the verifier that he knows a function that computes the "receiver's message," which in turn corresponds to a legitimate signature.

Proof: We construct a forger that, on input any message M and any verification-key $\mathsf{VK} = (\mathsf{PK}', h)$, where $\mathsf{PK}' = (\mathsf{PK}, f, g_1, g_2) \leftarrow \mathsf{G}^1(1^n)$ and $h \in \mathcal{H}_n$, generates a signature of M with respect to VK, as follows.

1. Let C be a circuit computing the hash function h. Let C_{M} be a circuit such that for every x, $C_{\mathsf{M}}(x) = n$ most significant bits of $C(x, \mathsf{M})$.

2. Compute the tree-commitment $a = \mathsf{TC}_f(\hat{C}_{\mathsf{M}})$, and let $w = auth_f(\hat{C}_{\mathsf{M}})$.

3. Compute $(b, r_1, r_2) = C(a, \mathsf{M})$.[14]

4. Let $q = (f, g_1, g_2, r_1, r_2)$, and emulate the interaction $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}(w), \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}|_q)(f, a, b)$, to produce a transcript $(q; ans) \leftarrow (\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}(w), \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}|_q)(f, a, b)$.

5. Output $(a; b, r_1, r_2; ans)$.

It is easy to verify that all forger steps are polynomial-time computable, and by completeness of $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$, the forger will always be successful. ■

[14] Notice that $((f, a, b), w) \in \mathcal{R}_{\mathcal{F}}$.

3.7.3 On the Security of ID^1

To establish $\neg(\mathsf{FS})$ it remains to show that there exist hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$, such that $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ is secure. Notice that it is easy to prove the security of $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ under the CSP hypothesis.

Lemma 3.7.2 *Under the* CSP *hypothesis, there exist hash-function ensembles* $\mathcal{G}^1$ *and* $\mathcal{G}^2$*, such that* $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ *is secure.*

Proof: The CSP hypothesis implies that there exist hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$ for which $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2},\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$ satisfies both the completeness property and the CS-proof-of-knowledge property (follows from Proposition 2).[15] It is easy to verify that $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ is secure, with respect to these hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$. ■

Thus, we proved $(\mathsf{CSP}) \implies \neg(\mathsf{FS})$. It remains to prove $\neg(\mathsf{CSP}) \implies \neg(\mathsf{FS})$. Unfortunately, we do not know how to prove this directly. Instead we proceed by considering the following two cases.

- (Case 1): There exist hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$ such that $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$, is secure.

- (Case 2): For all hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$, $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$ is not secure.

If we are in Case 1 we are done, since then there exist hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$ such that $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$, is secure, whereas $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2})$ is insecure with respect to any hash-function ensemble $\mathcal{H}$, and $\neg(\mathsf{FS})$ is established. Hence, we assume that we are in Case 2. Namely, we assume that for every hash-function ensembles $\mathcal{G}^1$ and $\mathcal{G}^2$, there exists an impersonator (formalized as a polynomial-size circuit family) for $\mathsf{ID}^1_{\mathcal{G}^1,\mathcal{G}^2}$. We formalize this impersonator by breaking it into two parts. The first part (denoted by $\tilde{\mathsf{F}} = \{\tilde{\mathsf{F}}^n\}$) impersonates the first message of the sender, and the second part (denoted by $\tilde{\mathsf{P}} = \{\tilde{\mathsf{P}}^n\}$) impersonates the second message of the sender. Formally, case 2 corresponds to the case that for every hash-function ensembles $\mathcal{G}^1$

[15] Actually, Proposition 2 implies that there exist hash-function ensembles $\mathcal{G}^1 = \mathcal{G}^2$ such that $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2},\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$ satisfies both the completeness property and the CS-proof-of-knowledge property.

and $\mathcal{G}^2$, there exists a polynomial-size circuit family $\tilde{\mathsf{F}}_1 = \{\tilde{\mathsf{F}}_1^n\}$, a polynomial-size circuit family $\tilde{\mathsf{P}}_1 = \{\tilde{\mathsf{P}}_1^n\}$, and a polynomial $p(\cdot)$, such that for infinitely many n's,

$$\Pr[(\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}|_q)(f,a,b) = 1 \ : \ a = \tilde{\mathsf{F}}_1^n(f,g_1,g_2) \ \wedge \ q = (f,,g_1,g_2,r_1,r_2)] \geq \frac{1}{p(n)}$$

(where the probability is over a random q and a random $b \in_R \{0,1\}^n$). We denote the set of all such n's by $S^1_{\mathcal{G}^1,\mathcal{G}^2}$.

We refer to this case by ($\forall \mathcal{G}^1$,$\mathcal{G}^2$ $\exists$IMPERSONATOR)

It remains to prove the following lemma.

Lemma 3.7.3 $(\forall \mathcal{G}^1, \mathcal{G}^2 \ \exists\mathsf{IMPERSONATOR}) \Rightarrow \neg(\mathsf{FS})$.

To prove this Lemma we construct yet two more ID schemes ID^2 and ID^3, such that one of them demonstrates the insecurity of the FS paradigm.

3.7.4 Construction of ID^2

The assumption $(\forall \mathcal{G}^1, \mathcal{G}^2 \ \exists\mathsf{IMPERSONATOR})$ implies that for every $n \in S^1_{\mathcal{G}^1,\mathcal{G}^2}$, given random (f, g_1, g_2), it is easy to find a and $b_1 \neq b_2$, and to convince, with non-negligible probability, both $\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}|_q(f,a,b_1)$ and $\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}|_q(f,a,b_2)$, where $q = (f,g_1,g_2,r_1,r_2)$ and r_1, r_2 are uniformly distributed. Note that in contrast, it is hard to convince both $\mathsf{V}^0(f,a,b_1)$ and $\mathsf{V}^0(f,a,b_2)$, since $(\mathsf{P}^0,\mathsf{V}^0)$ is a proof of knowledge, and anyone who knows a witness to both (f,a,b_1) and (f,a,b_2) can be used to find collisions to f. This contrast between V^0 and $\mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2}$ suggests constructing a new ID scheme, ID^2, whose security follows from the proof-of-knowledge property of $(\mathsf{P}^0,\mathsf{V}^0)$, and the insecurity of the corresponding FS signature scheme follows from the assumption $(\forall \mathcal{G}^1, \mathcal{G}^2 \ \exists\mathsf{IMPERSONATOR})$.

Let $\mathcal{F}$ be a collision resistant hash-function ensemble, and let $\mathsf{ID} = (\mathsf{G},\mathsf{S},\mathsf{R})$ be any secure canonical ID scheme. We extend ID to obtain a new ID scheme $\mathsf{ID}^2 = (\mathsf{G}^2,\mathsf{S}^2,\mathsf{R}^2)$, by extending the public key and the verdict function, as follows.

- G^2: On input 1^n,

  1. Run $\mathsf{G}(1^n)$, to obtain a pair $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{G}(1^n)$.
  2. Choose uniformly
     - $f \in \mathcal{F}_n$
     - $r \in \{0,1\}^t$, where $t = 4n$.
     - γ_1' (randomness for $\mathsf{V}_{\mathsf{PCP}}$).

  Output SK as the secret-key and $\mathsf{PK}' = (\mathsf{PK}, f, r, \gamma_1')$ as the public-key.

- R^2: On input a public-key $\mathsf{PK}' = (\mathsf{PK}, f, r, \gamma_1')$, R^2 accepts either views that $\mathsf{R}(\mathsf{PK})$ accepts or views of the form

$$\mathsf{S}^2 \qquad\qquad \mathsf{R}^2$$

$$\xrightarrow{\beta_2,\, r \oplus (a, b_1, b_2, \beta_1)}$$

$$\xleftarrow{\gamma_1, \gamma_2}$$

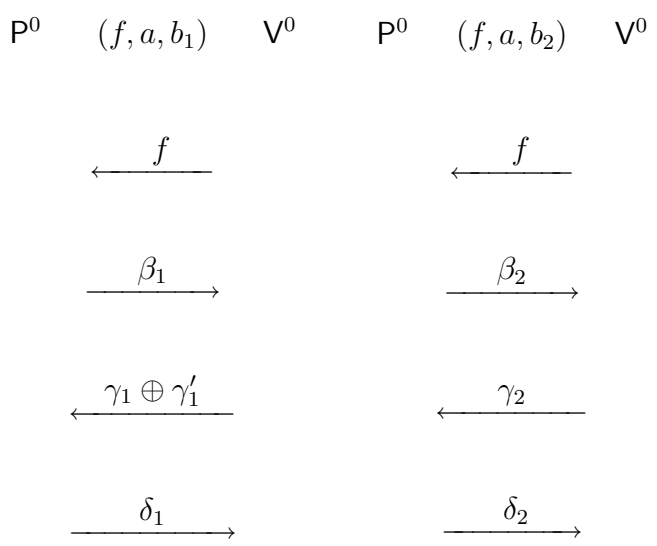
$$\xrightarrow{\delta_1, \delta_2}$$

  where

  - $b_1 \neq b_2$.
  - $(f; \beta_1; \gamma_1 \oplus \gamma_1'; \delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$.
  - $(f; \beta_2; \gamma_2; \delta_2) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_2))$.

Intuitively, the above view can be thought of as an interleaved execution of the fol-

lowing two views:



Remarks:

1. It is necessary to append γ_1' to the public-key in order to later establish the insecurity of the corresponding signature scheme. The reason is that when ID^2 will be converted into a signature scheme (by applying the Fiat-Shamir paradigm), the verifier will be replaced with a hash-function, and thus γ_1 will no longer necessarily be uniformly distributed. Yet, as we shall see, we only know how to establish the insecurity of the corresponding signature scheme if γ_1 is uniformly distributed. We get around this problem by XORing γ_1 with a uniformly distributed string γ_1', from the public-key.

   Notice that we do not append γ_2' to the public-key, and thus in some sense ID^2 is asymmetric. The reason for this asymmetry is quite technical and will become clearer in the proof of Lemma 3.7.5.

2. As in ID^1, f is used for two purposes. It is used both in the instances (f, a, b_1) and (f, a, b_2), and in the views $\mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$ and $\mathsf{ACC}(\mathsf{V}^0(f, a, b_2))$. As in ID^1, we can use one function for both purposes since both require the same property from f, namely, it being collision resistant.

3. Note that we mask (a, b_1, b_2, β_1) by XORing it with a random r, rather then simply sending $(a, b_1, b_2, \beta_1, \beta_2)$ in the clear. The reason is quite technical and will become clearer in the proof of Lemma 3.7.5.

3.7.5 The Security of ID^2

Lemma 3.7.4 *Assuming $\mathcal{F}$ is collision resistant, ID^2 is secure.*

Proof Idea. Assume for contradiction that ID^2 is not secure. That is, assume that there exists a cheating sender $\tilde{\mathsf{S}} = \{\tilde{\mathsf{S}}_n\}$ and a polynomial $p(\cdot)$ such that for infinitely many n's,

$$\Pr[(\tilde{\mathsf{S}}_n, \mathsf{R}^2)(\mathsf{PK}') = 1] \geq \frac{1}{p(n)}$$

(where the probability is over $\mathsf{PK}' \leftarrow \mathsf{G}^2(1^n)$ and over the random coin tosses of R^2).

We prove that the existence of $\tilde{\mathsf{S}}$ implies the existence of a circuit that finds collisions in $\mathcal{F}$. This is done in two parts, as follows.

- **(Part 1):** We first show that there exist non-uniform probabilistic polynomial-time Turing machines $\tilde{\mathsf{F}} = \{\tilde{\mathsf{F}}_n\}$ and $\tilde{\mathsf{P}} = \{\tilde{\mathsf{P}}_n\}$, such for infinitely many n's the following holds.

  For $(a, b_1, b_2, aux_1, aux_2) = \tilde{\mathsf{F}}_n(f)$,

  $$\Pr\left[(\tilde{\mathsf{P}}_n(aux_1), \mathsf{V}^0|_f)(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}_n(aux_2), \mathsf{V}^0|_f)(f, a, b_2) = 1\right] \geq 1/p(n)^2$$

  (where the probability is over a uniformly chosen $f \in \mathcal{F}_n$, and over the random coin tosses of F_n, $\tilde{\mathsf{P}}_n$, and both independent instances of $\mathsf{V}^0|_f$).[16]
  The proof-of-knowledge property of $(\mathsf{P}^0, \mathsf{V}^0)$ implies that there exists a probabilistic polynomial-time oracle machine E and a polynomial $p'(\cdot)$ such that for

[16] recall that $\mathsf{V}^0|_f$ is V^0 restricted to sending f as the first message.

any $(a, b_1, b_2, aux_1, aux_2)$ which satisfy the above inequality,

$$\Pr\left[\begin{array}{c} \forall i\ E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),i) = w_i^1 \text{ s.t. } ((f,a,b_1),w^1) \in \mathcal{R}_{\mathcal{F}} \\ \textit{and} \\ \forall i\ E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),i) = w_i^2 \text{ s.t. } ((f,a,b_2),w^2) \in \mathcal{R}_{\mathcal{F}} \end{array}\right] \geq \frac{1}{p'(n)}$$

(where the probability is over the random coin tosses of $E^{\tilde{\mathsf{P}}_n(aux_1)}$ and $E^{\tilde{P}_n(aux_2)}$).

- **(Part 2):** We then show that there exists a probabilistic polynomial-time oracle machine, with oracle access to E, $\tilde{\mathsf{F}}_n$ and $\tilde{\mathsf{P}}_n$, such that on input a uniformly chosen $f \in_R \mathcal{F}_n$, outputs a collision in f, with non-negligible probability.

Since non-uniform probabilistic polynomial-time Turing machines can be modeled as polynomial-size circuits, Part 1 together with Part 2 imply the existence of a polynomial-size circuit such that, on input a uniformly chosen $f \in_R \mathcal{F}_n$, outputs a collision in f, with non-negligible probability. This contradicts the assumption that $\mathcal{F}$ is collision resistant.

The formal proof is quite tedious and is deferred to Appendix A.1.

3.7.6 On the Insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$

We next consider the insecurity of the corresponding signature scheme $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$, which we denote by $\mathsf{SIG}^2_{\mathcal{H}} = (\mathsf{GEN}^2_{\mathcal{H}}, \mathsf{SIGN}^2_{\mathcal{H}}, \mathsf{VERIFY}^2_{\mathcal{H}})$. Proving the insecurity of $\mathsf{SIG}^2_{\mathcal{H}} = \mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^2)$ is tricky. Intuitively, we would like to use the assumption

$$(\forall \mathcal{G}^1, \mathcal{G}^2\ \exists \mathsf{IMPERSONATOR})$$

to forge signatures, as follows. Let $\mathcal{H}^1$ be a hash-function ensemble that computes the n most significant bits of the output of $\mathcal{H}$, and let $\mathcal{H}^2$ be a hash-function ensemble that computes the n least significant bits of the output of $\mathcal{H}$. We would like to use an impersonator for $\mathcal{H}^1, \mathcal{H}^2$. We denote this impersonator by $\mathsf{IMPERSONATOR}_{\mathcal{H}^1,\mathcal{H}^2}$.

Loosely speaking, given a random verification key $\mathsf{VK} = (\mathsf{PK}', h)$, where $h \in \mathcal{H}$ and $\mathsf{PK}' = (\mathsf{PK}, f, r, \gamma_1')$, we would like to prove the insecurity of $\mathsf{SIG}^2_{\mathcal{H}}$ by using $\mathsf{IMPERSONATOR}_{\mathcal{H}^1,\mathcal{H}^2}$ to find an a such that for random $b_1 \neq b_2$, $\mathsf{IMPERSONATOR}_{\mathcal{H}^1,\mathcal{H}^2}$ can fool both $\mathsf{V}^{\mathcal{H}^1}(f, a, b_1)$ and $\mathsf{V}^{\mathcal{H}^2}(f, a, b_2)$ to accept. However, in this approach $\mathsf{IMPERSONATOR}_{\mathcal{H}^1,\mathcal{H}^2}$ finds $\beta_1, \beta_2, \gamma_1, \gamma_2$ such that γ_1 depends only on β_1 and γ_2 depends only on β_2, whereas in valid signatures γ_1 and γ_2 are functions of both β_1 and β_2. Thus, to obtain a valid signature, we cannot simply run $\tilde{\mathsf{P}}^n_1$ twice independently, since the value of β_2 affects the value of γ_1 and vice versa.

To get around this problem we distinguish between the following two cases:

- *Case 2a:* ($\forall \mathcal{G}$ $\exists$strong-IMPERSONATOR)
- *Case 2b:* $\neg$($\forall \mathcal{G}$ $\exists$strong-IMPERSONATOR)

($\forall \mathcal{G}$ $\exists$strong-IMPERSONATOR) refers to the case that for every function ensemble $\mathcal{G}$ there exists a "*strong*"-impersonator, that for infinitely many n's, on input a random $f \in \mathcal{F}_n$, finds a and b_1 such that he can convince $\mathsf{V}^0(f, a, b_1)$ to accept, and can convince $\mathsf{V}^{\mathcal{G}}(f, a, b_2)$ to accept for a random b_2. We denote the set of all such n's by $\mathsf{S}^2_{\mathcal{G}}$. Formally speaking, ($\forall \mathcal{G}$ $\exists$strong-IMPERSONATOR) refers to the case that for every function ensemble $\mathcal{G}$ there exists a polynomial-size circuit family $\tilde{\mathsf{F}}_2 = \{\tilde{\mathsf{F}}^n_2\}$, a polynomial-size circuit family $\tilde{\mathsf{P}}_2 = \{\tilde{\mathsf{P}}^n_2\}$, a polynomial $p(\cdot)$, and an infinite set $S^2_{\mathcal{G}}$, such that for every $n \in S^2_{\mathcal{G}}$,

$$\Pr[(\tilde{\mathsf{P}}^n_2, \mathsf{V}^0|_f)(f, a, b_1) = 1 \wedge (\tilde{\mathsf{P}}^n_2, \mathsf{V}^{\mathcal{G}}|_q)(f, a, b_2) = 1 : (a, b_1) = \tilde{\mathsf{F}}^n_2(f, g) \wedge q = (f, g, r)] \geq \frac{1}{p(n)}$$

(where the probability is over a random q (i.e., over $f \in_R \mathcal{F}_n$, $g \in_R \mathcal{G}_n$, $r \in_R \{0,1\}^{4n}$), a random $b_2 \in_R \{0,1\}^n$, and over the random coin tosses of $\mathsf{V}^0|_f$).

We proceed by proving the insecurity of the FS paradigm is case 2a and in case 2b.

The insecurity of the FS paradigm in case 2a. In this case, we proceed with ID^2 and show that $\mathsf{SIG}^2_{\mathcal{H}}$ is insecure for every $\mathcal{H}$. More specifically, we show that for

every $\mathcal{H}$ and for every message M there exists a forger, who succeeds in forging a signature of M for every $n \in \mathsf{S}^2_{\mathcal{H}^{\mathsf{M}}}$, where $\mathcal{H}^{\mathsf{M}}$ is a hash-function ensemble which will be defined shortly.

Lemma 3.7.5 *Assuming* ($\forall \mathcal{G}$ $\exists$strong-IMPERSONATOR)*, for any function ensemble* $\mathcal{H}$ *the signature scheme* $\mathsf{SIG}^2_{\mathcal{H}}$ *is insecure.*

Proof: Fix a function ensemble $\mathcal{H}$ and any message M. We show that there exists a forger $\mathsf{FORG}^{\mathsf{M}}$ which, on input a random verification-key VK, outputs a signature of M, with non-negligible probability.

For any $n \in \mathbb{N}$ and for any $h \in \mathcal{H}_n$, define $h^{\mathsf{M}}(x) =$ n least-significant-bits of $h(x, \mathsf{M})$, and let $\mathcal{H}^{\mathsf{M}} = \{h^{\mathsf{M}}\}_{h \in \mathcal{H}}$. From our assumption there exist two polynomial-size circuit families $\tilde{\mathsf{F}}_2 = \{\tilde{\mathsf{F}}^n_2\}_{n \in \mathbb{N}}$ and $\tilde{\mathsf{P}}_2 = \{\tilde{\mathsf{P}}^n_2\}$ such that for every $n \in S^2_{\mathcal{H}^{\mathsf{M}}}$, for $(a, b_1) = \tilde{\mathsf{F}}^n_2(f, h^{\mathsf{M}})$, and for $q = (f, h^{\mathsf{M}}, r)$,

$$\Pr[(\tilde{\mathsf{P}}^n_2, \mathsf{V}^0|_f)(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}^n_2, \mathsf{V}^{\mathcal{H}^M}|_q)(f, a, b_2) = 1] \geq \frac{1}{\mathrm{poly}(n)}$$

(where the probability is over a randomly chosen q, a random $b_2 \in_R \{0,1\}^n$, and the random coin tosses of $\mathsf{V}^0|_f$).

On input a random verification-key $\mathsf{VK} = (\mathsf{PK}', h)$, where $h \in \mathcal{H}_n$ and $\mathsf{PK}' = (\mathsf{PK}, f, r, \gamma'_1)$, the forger $\mathsf{FORG}^{\mathsf{M}}$ generates a signature of M as follows.

1. Compute $(a, b_1) = \tilde{\mathsf{F}}^n_2(f, h^{\mathsf{M}})$.
2. Emulate the interaction of $(\tilde{\mathsf{P}}^n_2, \mathsf{V}^0|_f)(f, a, b_1)$, to obtain a transcript

$$(f; \beta_1; *; *) \leftarrow (\tilde{\mathsf{P}}^n_2, \mathsf{V}^0|_f)(f, a, b_1).$$

3. Choose randomly $b_2 \in \{0,1\}^n$. Let $r' = r \oplus (a, b_1, b_2, \beta_1)$ and let $q' = (f, h^{\mathsf{M}}, r')$.[17]
4. Emulate the interaction of $(\tilde{\mathsf{P}}^n_2, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q'})(f, a, b_2)$, to obtain a transcript

$$(q'; ans) \leftarrow (\tilde{\mathsf{P}}^n_2, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q'})(f, a, b_2).$$

[17] When proving that this forger is successful, we use the fact that r' is uniformly distributed. This is why we masked (a, b_1, b_2, β_1) by XORing it with a random r.

Denote $ans = (\beta_2, \gamma_2, \delta_2)$.

5. Compute $(\gamma_1, *) = h(\beta_2, r', \mathsf{M})$.[18]

6. Emulate the interaction $(\tilde{\mathsf{P}}_2^n, \mathsf{V}^0|_{f,\gamma_1 \oplus \gamma_1'})(f, a, b_1)$ to obtain a transcript

$$(f; \beta_1; \gamma_1 \oplus \gamma_1'; \delta_1) \leftarrow (\tilde{\mathsf{P}}_2^n, \mathsf{V}^0|_{f,\gamma_1 \oplus \gamma_1'})(f, a, b_1).$$

7. Output $(\beta_2, r \oplus (a, b_1, b_2, \beta_1); \gamma_1, \gamma_2; \delta_1, \delta_2)$ as a signature of M.

We claim that the forger will be successful with non-negligible probability.

Claim 18

$$\Pr[\mathsf{VERIFY}_{\mathcal{H}}^2(\mathsf{VK}, \mathsf{M}, \mathsf{FORG}^{\mathsf{M}}(\mathsf{VK})) = 1] = \mathsf{non\text{-}negl}(n)$$

(*where the probability is over* VK *and over the random coin tosses of* $\mathsf{FORG}^{\mathsf{M}}$).

Since the proof is quite technical it is deferred to Appendix A.2.

■

It remains to prove the insecurity of the FS paradigm in case 2*b*. We construct yet another and final ID scheme ID^3, which demonstrates the insecurity of the FS paradigm in this case.

3.7.7 Construction of ID^3

Throughout the remaining of Chapter 3, we assume

$$(\forall \mathcal{G}^1, \mathcal{G}^2\ \exists \mathsf{IMPERSONATOR}) \ \wedge \ \neg(\forall \mathcal{G}\ \exists \mathsf{strong\text{-}IMPERSONATOR}).$$

We establish $\neg(\mathsf{FS})$ by extending any secure ID scheme into a new ID scheme $\mathsf{ID}^3 = (\mathsf{G}^3, \mathsf{S}^3, \mathsf{R}^3)$. The security of ID^3 follows from the assumption $\neg(\forall \mathcal{G}\ \exists\ \mathsf{strong\text{-}IMPERSONATOR})$, and the insecurity of the corresponding signature scheme $\mathsf{SIG}_{\mathcal{H}}^3 = \mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^3)$ (for

[18] Notice that $(\gamma_1, \gamma_2) = h(\beta_2, r', \mathsf{M})$.

$n \in S^1_{\mathcal{H}}$) follows from the assumption ($\forall \mathcal{G}^1, \mathcal{G}^2\ \exists$IMPERSONATOR). Recall that, roughly speaking, in ID^1 there was one execution of $(\mathsf{P}^{\mathcal{G}^1,\mathcal{G}^2}, \mathsf{V}^{\mathcal{G}^1,\mathcal{G}^2})$, and in ID^2 there were two parallel executions of $(\mathsf{P}^0, \mathsf{V}^0)$. ID^3 is, in some sense, a hybrid of ID^1 and ID^2. It once executes $(\mathsf{P}^{\mathcal{G}}, \mathsf{V}^{\mathcal{G}})$ and once executes $(\mathsf{P}^0, \mathsf{V}^0)$.

Fix a hash-function ensemble $\mathcal{G}$ that does not have a strong-IMPERSONATOR (one exists by our assumption). Take any secure canonical ID scheme $\mathsf{ID} = (\mathsf{G}, \mathsf{S}, \mathsf{R})$ and define ID^3 by extending the public key and the verdict function, as follows.

- G^3: On input 1^n,

  1. Run $\mathsf{G}(1^n)$, to obtain a pair $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{G}(1^n)$.
  2. Choose uniformly
     - $f \in \mathcal{F}_n$
     - $g \in \mathcal{G}$
     - $r_1, r'_2 \in \{0,1\}^{4n}$
     - $b'_2 \in \{0,1\}^n$.

  Output SK as the secret-key and $\mathsf{PK}' = (\mathsf{PK}, f, g, r_1, r'_2, b'_2)$ as the public-key.

- R^3: On input a public-key $\mathsf{PK}' = (\mathsf{PK}, f, g, r_1, r'_2, b'_2)$, R^3 accepts either views that $\mathsf{R}(\mathsf{PK})$ accepts or views of the form

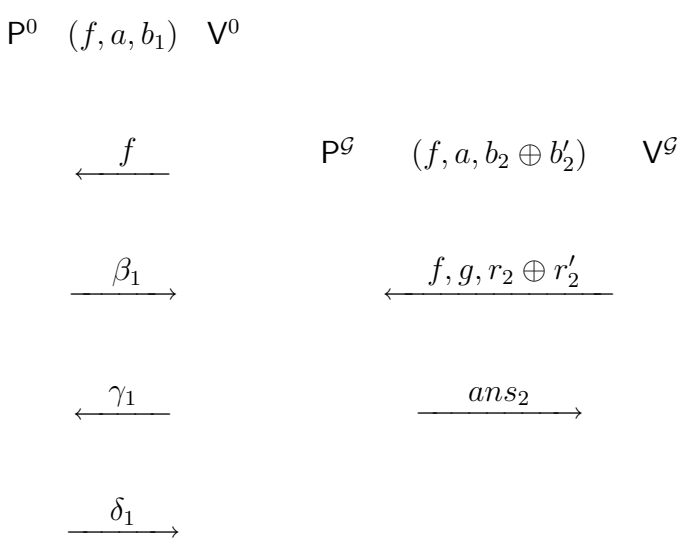
$$\begin{array}{ccc} S^3 & & R^3 \\ & \xrightarrow{\quad\beta_1, r_1 \oplus (a, b_1)\quad} & \\ & \xleftarrow{\quad\gamma_1, b_2, r_2\quad} & \\ & \xrightarrow{\quad\delta_1, ans_2\quad} & \end{array}$$

  where

  - $(f; \beta_1; \gamma_1; \delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$

– $(q_2; ans_2) \in \mathsf{ACC}(\mathsf{V}^{\mathcal{G}}(f, a, b_2 \oplus b_2'))$, where $q_2 = (f, g, r_2 \oplus r_2')$.

Intuitively, the above view can be thought of as an interleaved execution of the following two views:



Remarks:

1. As in ID^2, it is necessary to append b_2', r_2' to the public-key in order to later establish the insecurity of $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^3)$. More specifically, when ID^3 will be converted into a signature scheme (by applying the FS paradigm), the verifier will be replaced with a hash-function, and thus b_2 and r_2 will no longer necessarily be chosen at random. Yet, we only know how to establish the insecurity of the signature scheme assuming that b_2 and r_2 are chosen at random. We get around this problem by XORing b_2 with a uniformly distributed string b_1' and XORing r_2 with a uniformly distributed string r_2'.

2. As in ID^1 and ID^2, f serves two purposes. It is used both in the instances (f, a, b_1) and $(f, a, b_2 \oplus b_2')$, and in the transcripts of $\mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$ and $\mathsf{ACC}(\mathsf{V}^{\mathcal{G}}(f, a, b_2 \oplus b_2'))$.

3. As in ID^2 and for a similar reason, we mask (a, b_1), by XORing it with a random r_1.

Lemma 3.7.6 *Assuming $\mathcal{G}$ does not have a* strong-IMPERSONATOR, ID^3 *is secure.*

Proof: Follows easily from the definition of a strong-IMPERSONATOR. ■

We denote $\mathsf{FS}_{\mathcal{H}}(\mathsf{ID}^3)$ by $\mathsf{SIG}^3_{\mathcal{H}} = (\mathsf{GEN}^3_{\mathcal{H}}, \mathsf{SIGN}^3_{\mathcal{H}}, \mathsf{VERIFY}^3_{\mathcal{H}})$.

Lemma 3.7.7 *Assuming* $(\forall \mathcal{G}^1, \mathcal{G}^2\ \exists$IMPERSONATOR$)$*, for any hash-function ensemble $\mathcal{H}$ the signature scheme $\mathsf{SIG}^3_{\mathcal{H}}$ is insecure.*

Proof: Fix any hash-function ensemble $\mathcal{H}$ and any message M. We show that exists a forger $\mathsf{FORG}^{\mathsf{M}}$ which, on input a random verification key $\mathsf{VK} = (\mathsf{PK}', h)$, where $\mathsf{PK}' = (\mathsf{PK}, f, g, r_1, r'_2, b'_2)$, outputs a signature of M with non-negligible probability.

For any $n \in \mathbb{N}$ and for any $h \in \mathcal{H}_n$, define $h^{\mathsf{M}}(x) =$ n most-significant-bits of $h(x, \mathsf{M})$, and let $\mathcal{H}^{\mathsf{M}} = \{h^{\mathsf{M}}\}_{h \in \mathcal{H}}$. By our assumption $(\forall \mathcal{G}^1, \mathcal{G}^2\ \exists$IMPERSONATOR$)$ there exists an impersonator for $\mathcal{H}^{\mathsf{M}}$ and $\mathcal{G}$. Namely, there exist $\tilde{\mathsf{F}}_1 = \{\tilde{\mathsf{F}}^n_1\}_{n \in \mathbb{N}}$, $\tilde{\mathsf{P}}_1 = \{\tilde{\mathsf{P}}^n_1\}$, and a polynomial $p(\cdot)$, such that for every $n \in S^1_{\mathcal{H}^{\mathsf{M}}, \mathcal{G}}$, for $a = \tilde{\mathsf{F}}^n_1(f, h^{\mathsf{M}}, g)$, and for $q = (f, h^{\mathsf{M}}, g, r_1, r_2)$,

$$\Pr[(\tilde{\mathsf{P}}^n_1, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}, \mathcal{G}}|q)(f, a, b) = 1] \geq \frac{1}{p(n)}$$

(where the probability is over a randomly chosen q and a random $b \in_R \{0,1\}^n$).

This implies that there exists $\tilde{\tilde{\mathsf{P}}}_1 = \{\tilde{\tilde{\mathsf{P}}}^n_1\}$, and a polynomial $p'(\cdot)$ such that for every $n \in S^1_{\mathcal{H}^{\mathsf{M}}, \mathcal{G}}$, and for $a = \tilde{\mathsf{F}}^n_1(f, h^{\mathsf{M}}, g)$,

$$\Pr[(\tilde{\tilde{\mathsf{P}}}^n_1, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{(f, h^{\mathsf{M}}, r_1)})(f, a, b_1) = 1\ \wedge\ (\tilde{\tilde{\mathsf{P}}}^n_1, \mathsf{V}^{\mathcal{G}}|_{(f, g, r_2)})(f, a, b_2) = 1] \geq \frac{1}{p'(n)}$$

(where the probability is over $f \in_R \mathcal{F}_n$, $h^{\mathsf{M}} \in_R \mathcal{H}^{\mathsf{M}}$, $g \in_R \mathcal{G}$, $r_1, r_2 \in_R \{0,1\}^{4n}$, and $b_1, b_2 \in_R \{0,1\}^n$). For simplicity, we abuse the notations and denote $\tilde{\tilde{\mathsf{P}}}_1$ by $\tilde{\mathsf{P}}_1$.

The forger $\mathsf{FORG}^{\mathsf{M}}$ generates a signature of M, with respect to VK, as follows.

1. Compute $a = \tilde{\mathsf{F}}^n_1(f, h^{\mathsf{M}}, g)$.

2. (a) Choose $b_1 \in_R \{0,1\}^n$, and compute $r'_1 = r_1 \oplus (a, b_1)$.

(b) Emulate the interaction of $(\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q_1})(f, a, b_1)$, where $q_1 = (f, h^{\mathsf{M}}, r_1')$ to obtain a transcript

$$(q_1; ans_1) \leftarrow (\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q_1})(f, a, b_1).$$

Denote $ans_1 = (\beta_1, \gamma_1, \delta_1)$.

3. Compute $(*, b_2, r_2) = h(\beta_1, r', \mathsf{M})$.[19]

4. Emulate the interaction of $(\tilde{\mathsf{P}}_n^1, \mathsf{V}^{\mathcal{G}}|_{q_2})(f, a, b_2 \oplus b_2')$, where $q_2 = (f, g, r_2 \oplus r_2')$, to obtain a transcript $(q_2; ans_2) \leftarrow (\tilde{\mathsf{P}}_n^1, \mathsf{V}^{\mathcal{G}}|_{q_2})(f, a, b_2 \oplus b_2')$.

5. Output $(\beta_1, r_1 \oplus (a, b_1); \gamma_1, b_2, r_2; \delta_1, ans_2)$ as a signature of M.

We claim that the forger is successful with non-negligible probability.

Claim 19 *There exists a polynomial $p(\cdot)$ such that for every $n \in S^1_{\mathcal{H}^{\mathsf{M}},\mathcal{G}}$*

$$\Pr[\mathsf{VERIFY}^3_{\mathcal{H}}(\mathsf{VK}, \mathsf{M}, \mathsf{FORG}^{\mathsf{M}}(\mathsf{VK})) = 1] \geq \frac{1}{p(n)}$$

(*where the probability is over* VK *and over the random coin tosses of* $\mathsf{FORG}^{\mathsf{M}}$).

Again, due to the technical flavor of the proof of the above claim, we defer it to Appendix A.3.

Thus, we have established the insecurity of $\mathsf{SIG}^3_{\mathcal{H}}$. ■

This concludes the proof of Theorem 12, which is summarized in Figure 1.

[19] Notice that $(\gamma_1, \beta_2, r_2) = h(\beta_1, r', \mathsf{M})$.

Chapter 4

The Impossibility of Program Obfuscation

In this chapter we state and prove our results regarding the impossibility of program obfuscation. As was mentioned in Chapter 2, we model adversaries as non-uniform probabilistic polynomial-time Turing machines. We refer to such machines as PPT Turing machines. We also distinguish between a family of circuits (which has one circuit for each input size) and a class of circuits (which has many circuits for each input size). A class of circuits is of the form $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ (denoted by calligraphic letters), where for every $n \in \mathbb{N}$, $\mathcal{C}_n$ is a set of many circuits, each on inputs of size n.

4.1 Definitions

In this subsection we present the original definition of obfuscation due to Barak *et al.* [4] (Section 4.1.1) and present our definitions of obfuscation w.r.t. auxiliary input (Section 4.1.2). We note that in [4] two definitions of obfuscation were presented: one in which programs were modeled as Turing machines, and one in which programs were modeled as Boolean circuits. Throughout this manuscript we always model programs as Boolean circuits. We note that modeling programs as Boolean circuits, rather than Turing machines, weakens the definition and thus strengthen the impossibility results.

4.1.1 Obfuscation

Definition 20 *[4]: A probabilistic algorithm $\mathcal{O}$ is an* obfuscator *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if the following three conditions are satisfied:*

- (Functionality): *There exists a negligible function $\mu(\cdot)$ such that for every $n \in \mathbb{N}$ and every $C \in \mathcal{C}_n$, with probability $1 - \mu(n)$ over the internal coin tosses of the obfuscator, $\mathcal{O}(C)$ describes a circuit that computes the same function as C.*[1]

- (Polynomial blowup): *There is a polynomial $l(\cdot)$ such that for every $C \in \mathcal{C}$, $|\mathcal{O}(C)| \leq l(|C|)$.*

- ("Virtual black-box" property): *For every* PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function $\mu(\cdot)$, such that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, and every predicate $\pi(\cdot)$,*

$$|\Pr[\mathcal{A}(\mathcal{O}(C)) = \pi(C)] - \Pr[\mathcal{S}^C(1^n) = \pi(C)]| < \mu(n).$$[2]

Throughout this thesis, we restrict our attention to efficient obfuscators, defined as follows.

Definition 21 *An obfuscator $\mathcal{O}$ is said to be* efficient *if it runs in probabilistic polynomial time.*

A few positive results for obfuscation (in the plain model) exist in the literature [5, 6, 36]. All these positive results are for *weak obfuscators*, which have the following weaker variant of the "virtual black-box" property:

For every PPT $\mathcal{A}$ *and every polynomial $p(\cdot)$ there exists a* PPT $\mathcal{S}$ *such*

[1] The original definition in [4] considered a slightly stronger functionality property: they required that $\mathcal{O}(C)$ *always* computes the same function as C. This was relaxed as above in [36].

[2] [4] formalized the "virtual black-box" property in a different, yet equivalent, way. They required that for every PPT $\mathcal{A}$ there exists a PPT $\mathcal{S}$ and a negligible function $\mu(\cdot)$, such that for every $n \in \mathbb{N}$, and every $C \in \mathcal{C}_n$,

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - \Pr[\mathcal{S}^C(1^n) = 1]\right| < \mu(n).$$

that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, and every predicate $\pi(\cdot)$,

$$|\Pr[\mathcal{A}(\mathcal{O}(C)) = \pi(C)] - \Pr[\mathcal{S}^C(1^n) = \pi(C)]| < \frac{1}{p(n)}.$$

We note that the negative results of [4] hold also for weak obfuscators. Similarly, our negative results hold also for weak obfuscators w.r.t. auxiliary input.

4.1.2 Obfuscation w.r.t. Auxiliary Input

We consider two definitions of obfuscation w.r.t. auxiliary input. In the first definition we allow the auxiliary input to *depend* on the function being obfuscated, whereas in the second definition we require the auxiliary input to be *independent* of the function being obfuscated. Both definitions follow the spirit of the original definition of obfuscation given in [4].

Definition 22 (Obfuscation w.r.t. dependent auxiliary input): *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator w.r.t. dependent auxiliary input *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if it satisfies the functionality property and the polynomial blowup property as in Definition 20, and in addition it satisfies the following "virtual black box" property:*

For every PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function $\mu(\cdot)$, such that for every polynomial $q(\cdot)$, every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, every auxiliary input z of size $q(n)$ (z may depend on C), and every predicate $\pi(\cdot)$,*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - \Pr[\mathcal{S}^C(1^n, z) = \pi(C, z)]\right| \leq \mu(n).$$

Definition 23 (Obfuscation w.r.t. independent auxiliary input): *A probabilistic algorithm $\mathcal{O}$ is an* obfuscator w.r.t. independent auxiliary input *for a class of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ if it satisfies the functionality property and the polynomial blowup property as in Definition 20, and in addition it satisfies the following "virtual black box" property:*

For every PPT $\mathcal{A}$ *there exists a* PPT $\mathcal{S}$ *and a negligible function* $\mu(\cdot)$*, such that for every polynomial* $q(\cdot)$*, every* $n \in \mathbb{N}$*, every auxiliary input* z *of size* $q(n)$*, and every predicate* $\pi(\cdot)$*,*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - \Pr[\mathcal{S}^C(1^n, z) = \pi(C, z)]\right| \leq \mu(n),$$

where the probabilities are over $C \in_R \mathcal{C}_n$*, and over the random coin tosses of* $\mathcal{A}$*,* $\mathcal{S}$*, and* $\mathcal{O}$*.*

Notice that Definition 23 is weaker than Definition 22, not only because the auxiliary input is independent of the function being obfuscated, but also because in Definition 23 the simulator $\mathcal{S}$ is required to succeed only for random $C \in_R \mathcal{C}_n$ (whereas in Definition 22 the simulator $\mathcal{S}$ is required to succeed for every $C \in \mathcal{C}_n$). As was noted in the Introduction, considering only randomly chosen circuits seems to suffice for most cryptographic applications. Moreover, even if Definition 23 does seem to be too weak, this is not a concern to us, since we are mainly proving impossibility results, and an impossibility of achieving a weak definition implies an impossibility of achieving a stronger one.

Our negative results hold also for the notion of *weak obfuscation w.r.t. auxiliary input*, which is defined analogously to weak obfuscation (without auxiliary input). The definition of weak obfuscation w.r.t. *dependent* auxiliary input has the following weaker variant of the "virtual black-box" property:

> *For every* PPT $\mathcal{A}$*, every polynomial* $p(\cdot)$*, and every polynomial* $q(\cdot)$ *there exists a* PPT $\mathcal{S}$ *such that for every* $n \in \mathbb{N}$*, every* $C \in \mathcal{C}_n$*, every auxiliary input* z *of size* $q(n)$*, and every predicate* $\pi(\cdot)$*,*
>
> $$\left|\Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - \Pr[\mathcal{S}^C(1^n, z) = \pi(C, z)]\right| < \frac{1}{p(n)}.$$

The definition of weak obfuscation w.r.t. *independent* auxiliary input has the following weaker variant of the "virtual black-box" property:

For every PPT $\mathcal{A}$*, every polynomial* $p(\cdot)$*, and every polynomial* $q(\cdot)$ *there exists a* PPT $\mathcal{S}$ *such that for every* $n \in \mathbb{N}$*, every auxiliary input* z *of size* $q(n)$*, and every predicate* $\pi(\cdot)$*,*

$$\left|\Pr[\mathcal{A}(\mathcal{O}(C), z) = \pi(C, z)] - \Pr[\mathcal{S}^C(1^n, z) = \pi(C, z)]\right| < \frac{1}{p(n)},$$

where the probabilities are over $C \in_R \mathcal{C}_n$*, and over the random coin tosses of* $\mathcal{A}$*,* $\mathcal{S}$*, and* $\mathcal{O}$*.*

4.2 Road Map

When proving our impossibility results, we exploit the following distinction between an obfuscated circuit and black-box access to a circuit:

1. An obfuscation $\mathcal{O}(C)$ is a small (polynomial size) circuit that agrees with C.
2. Given black-box access to C, it is hard to construct a small circuit that agrees with C.

For item 2 to hold, we need to assume that $\mathcal{C}$ is "sufficiently unpredictable." To this end, we define in Section 4.3 the notion of *pseudo entropy of a class of circuits*. We use this notion to present our negative results for weak obfuscation w.r.t. *independent* auxiliary input in Section 4.4, and our negative results for weak obfuscation w.r.t. *dependent* auxiliary input in Section 4.5.

4.3 Pseudo Entropy of a Class of Circuits

Loosely speaking, we say that a class of circuits $\mathcal{C}$ has pseudo entropy at least $p(\cdot)$ if there exist polynomial size sets $I_n \subseteq \{0,1\}^n$ such that the set $C(I_n)$ looks as if it has min-entropy at least $p(n)$, even given oracle access to C on $\bar{I}_n \triangleq \{0,1\}^n \setminus I_n$. This is formalized as follows.

Definition 24 (Pseudo entropy of a class of circuits): *We say that a class of circuits* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *has* pseudo entropy *at least* $p(\cdot)$ *if there exists a polynomial* $t(\cdot)$ *and sets* $I_n \subseteq \{0,1\}^n$ *of size* $t(n)$*, and for every* $C \in \mathcal{C}_n$ *there is a random variable* $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ *such that the following holds:*

1. $\vec{Y}^C$ *has (statistical) min entropy at least* $p(n)$.[3]
2. *For every* PPT *oracle machine* $\mathcal{D}$ *there is a negligible function* $\mu(\cdot)$ *such that for every* $n \in \mathbb{N}$*,*

$$\left|\Pr[\mathcal{D}^{C|_{\bar{I}_n}}(\vec{Y}^C) = 1] - \Pr[\mathcal{D}^{C|_{\bar{I}_n}}(C(I_n)) = 1]\right| \leq \mu(n),$$

where the probabilities are over $C \in_R \mathcal{C}_n$*,* $\vec{Y}^C$*, and the random coin tosses of* $\mathcal{D}$*. The circuit* $C|_{\bar{I}_n}$ *agrees with* C *on every* $x \notin I_n$ *and outputs* $\perp$ *on every* $x \in I_n$*.*

There is a slight abuse of notations here. We use I_n to denote both a set and a list (or a vector). For $I_n = (x_1, \ldots, x_{t(n)})$ we let $C(I_n) = (C(x_1), \ldots, C(x_{t(n)}))$.

Definition 25 *We say that a class of circuits* $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *has* super-polynomial pseudo entropy *if it has pseudo entropy at least* $p(\cdot)$*, for every polynomial* $p(\cdot)$*.*

We give a few examples of natural classes of circuits that have super-polynomial pseudo entropy.

Claim 26 *The following classes of circuits all have super-polynomial pseudo entropy:*

1. Every *class of pseudo-random functions.*
2. Every *randomized digital signature algorithm,*[4] *in which the signer replaces the randomness by applying a (secret) pseudo-random function to the message to be signed.*

[3] A random variable X over some set S is said to have (statistical) min-entropy at least k if for every $x \in S$, $\Pr[X = x] \leq 2^{-k}$.

[4] A signature scheme is said to be randomized if for every message M and for every signing key SK, the random variable $\mathsf{SIGN}_{\mathsf{SK}}(\mathsf{M})$ has (statistical) min-entropy at least 1.

3. Every *semantic secure secret-key encryption algorithm,*[5] *in which the randomness is replaced by applying a (secret) pseudo-random function to the message to be encrypted.*

Proof:

1. Follows immediately from the definition of pseudo-random functions.

2. Let $\mathsf{SIG} = (\mathsf{GEN}, \mathsf{SIGN}, \mathsf{VERIFY})$ be *any* randomized digital signature algorithm, and let SIGN' be the deterministic signature algorithm obtained by taking any pseudo-random function ensemble $\mathcal{F} = \{f_s\}$, and modifying the (randomized) signing algorithm $\mathsf{SIGN}_{\mathsf{SK}}(\cdot)$ by appending a (random) seed s of $\mathcal{F}$ to its signing key, and by setting $\mathsf{SIGN}'_{\mathsf{SK},s}(\mathsf{M}) \triangleq \mathsf{SIGN}_{\mathsf{SK}}(\mathsf{M}; f_s(\mathsf{M}))$ (i.e., $\mathsf{SIGN}'_{\mathsf{SK},s}(\mathsf{M})$ runs the signing algorithm $\mathsf{SIGN}_{\mathsf{SK}}$ on message M with randomness $f_s(\mathsf{M})$). For every set of $t(n)$ messages $I_n = (\mathsf{M}_1, \ldots, \mathsf{M}_{t(n)})$, let $\vec{Y}^{\mathsf{SK}} = (Y_1, \ldots, Y_{t(n)})$ be a sequence of $t(n)$ independent random variables, where each Y_i is identically distributed to $\mathsf{SIGN}_{\mathsf{SK}}(\mathsf{M}_i)$. The fact that each Y_i has (statistical) min-entropy at least 1, implies that $\vec{Y}^{\mathsf{SK}}$ has (statistical) min-entropy at least $t(n)$. It remains to notice that the pseudo-randomness of $\mathcal{F}$ implies that every PPT oracle machine $\mathcal{D}^{\mathsf{SIGN}'_{\mathsf{SK},s}|\bar{I}_n}$ cannot distinguish between the random variable $\mathsf{SIGN}'_{\mathsf{SK},s}(I_n) = (\mathsf{SIGN}'_{\mathsf{SK},s}(\mathsf{M}_1), \ldots, \mathsf{SIGN}'_{\mathsf{SK},s}(\mathsf{M}_{t(n)}))$ and the random variable $\vec{Y}^{\mathsf{SK}} = (Y_1, \ldots, Y_{t(n)})$ (for randomly chosen s), implying that the class $\{\mathsf{SIGN}'_{\mathsf{SK},s}\}$ has super-polynomial pseudo entropy.

3. Let ENC be any secure (possibly randomized) secret-key encryption scheme, and let ENC' be the deterministic encryption scheme obtained by taking any pseudo-random function ensemble $\mathcal{F} = \{f_s\}$, and modifying the (possibly randomized) encryption scheme $\mathsf{ENC}_k(\cdot)$ by appending a (random) seed s of $\mathcal{F}$ to its secret-key, and by setting $\mathsf{ENC}'_{k,s}(\mathsf{M}) \triangleq \mathsf{ENC}_k(\mathsf{M}; f_s(\mathsf{M}))$. For any polynomial $t(\cdot)$, any $n \in \mathbb{N}$, and any set of messages $I_n = (\mathsf{M}_1, \ldots, \mathsf{M}_{t(n)})$, let $\vec{Y}^k = (Y_1, \ldots, Y_{t(n)})$ be $t(n)$ identical and independent random variables: Each Y_i chooses at random $\mathsf{M}'_i \in \{0,1\}^n$

[5] We refer the reader to [14] for the precise definition of semantic security.

and sets $Y_i = \mathsf{ENC}'_{k,s}(\mathsf{M}'_i)$. Clearly, each Y_i has (statistical) min-entropy n. Using a standard hybrid argument, it is not hard to see that the fact that ENC is semantic secure implies that every PPT oracle machine $\mathcal{D}^{\mathsf{ENC}'_{k,s}|_{\bar{I}_n}}$ cannot distinguish between the random variable $\mathsf{ENC}'_{k,s}(I_n) = (\mathsf{ENC}'_{k,s}(\mathsf{M}_1), \ldots, \mathsf{ENC}'_{k,s}(\mathsf{M}_{t(n)}))$ and the random variable $\vec{Y}^k = (Y_1, \ldots, Y_{t(n)})$ (for randomly chosen k, s), implying that ENC has super-polynomial pseudo entropy.

■

Remark. As was mentioned in Section 4.2, when proving negative results for obfuscation we exploit the difference between having an obfuscation $\mathcal{O}(C)$, which is a small circuit that agrees with C, and having black-box access to C, from which it is hard to construct a small circuit that agrees with C. However, for this distinction to hold we need to assume that C is sufficiently "unpredictable." This is exactly where the notion of super-polynomial pseudo entropy comes into play. When proving impossibility results, we exploit the following distinction:

1. An obfuscation $\mathcal{O}(C)$ is a small (polynomial size) circuit that agrees with C on I_n.

2. Given black-box access to C, it is hard to construct a small circuit that agrees with C on I_n.

When arguing for (2), we use the fact that $\mathcal{C}$ has super-polynomial pseudo entropy. If $\mathcal{C}$ has super-polynomial pseudo entropy, then given black box access to $C|_{\bar{I}_n}$, it is hard to distinguish (for $C \in_R \mathcal{C}_n$) between the pair $(I_n, C(I_n))$ and the pair $(I_n, \vec{Y}^C)$, where $\vec{Y}^C$ is a random variable with high (statistical) min-entropy. Using the connection between (statistical) min entropy and compression, it can be shown that with high probability (over the random variable $\vec{Y}^C$) there does not exist a small circuit v such that $v(x_i) = Y_i$, where $I_n = (x_1, \ldots, x_{t(n)})$ and $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$. Thus, it must be the case that for a random $C \in \mathcal{C}_n$ it is hard to come up with a small circuit that agrees with C on I_n. Otherwise, this can be used to distinguish between the pair $(I_n, C(I_n))$ and the pair $(I_n, \vec{Y}^C)$. We elaborate on this in Sections 4.4 and 4.5.

4.4 Impossibility of Obfuscation w.r.t. Independent Auxiliary Input

In this subsection, we define the classes of *filter functions*, and show that many natural classes of filter functions are not even weakly obfuscatable w.r.t. independent auxiliary input.

Definition 27 *For any $\mathcal{NP}$ language L and for any class of circuits $\mathcal{C}$, the class of* filter functions $\mathcal{C}^L$ *is defined by $\mathcal{C}^L \triangleq \{C^L : C \in \mathcal{C}\}$, where each function C^L is defined by*

$$C^L(x,w) \stackrel{\text{def}}{=} \begin{cases} C(x,w) & \text{if } (x,w) \in \mathcal{R}_L \\ \perp & \text{otherwise} \end{cases}$$

We show that the class of filter functions $\mathcal{C}^L$ is not weakly obfuscatable w.r.t. *independent* auxiliary input, for L and $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ that satisfy the following properties:

1. *L is an $\mathcal{NP}$-complete language.*

2. *$\mathcal{C}$ is strongly unpredictable:* For every $x \in \{0,1\}^n$, and for a random $C \in_R \mathcal{C}_n$, given oracle access to C everywhere except at the point x, it is hard to guess $C(x)$ (except with negligible probability).

3. *$\mathcal{C}$ has super-polynomial pseudo entropy over elements in L*: for every polynomial $p(\cdot)$ there exists a polynomial $t(\cdot)$ and sets $I_n \subseteq L \cap \{0,1\}^n$ of $t(n)$ elements, and for every $C \in \mathcal{C}_n$ there exists a random variable $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ with (statistical) min entropy at least $p(n)$, such that every PPT oracle machine $\mathcal{D}^{C|_{\bar{I}_n}}$ cannot distinguish (for $C \in_R \mathcal{C}_n$) between the random variable $C(I_n)$ and the random variable $\vec{Y}^C$.

We note that all our natural examples of classes of circuits given in Claim 26, satisfy both properties 2 and 3 (for every $\mathcal{NP}$-complete language L).

Theorem 28 *For every L and $\mathcal{C}$ that satisfy the above three properties, the class of circuits $\mathcal{C}^L$ is not weakly obfuscatable w.r.t. independent auxiliary input.*

Before presenting the full proof of Theorem 28, we sketch the main ideas.

Proof Sketch. The proof is by contradiction. Assume that there exist $\mathcal{C}$ and L as above, such that $\mathcal{C}^L$ is weakly obfuscatable w.r.t. independent auxiliary input by an obfuscator $\mathcal{O}$. The main idea is to exploit the fact that $\mathcal{C}^L$ has the property, that for every $n \in \mathbb{N}$ there is a set $I_n = (x_1, \ldots, x_{t(n)})$ of $t(n)$ elements in $L \cap \{0,1\}^n$, and a set $W_n = (w_1, \ldots, w_{t(n)})$ of $t(n)$ corresponding witnesses, such that:

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C^L \in_R \mathcal{C}_n^L$, it is hard to come up with a "small" circuit that computes C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$. This follows from the fact that $\mathcal{C}$ has super-polynomial pseudo entropy over L.

2. For every $C \in \mathcal{C}_n$, the obfuscated circuit $\mathcal{O}(C^L)$ is itself a "small" circuit that computes C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$.

We exploit this difference to obtain a contradiction. However, for this we need to show that given $\mathcal{O}(C^L)$ it is easy to compute a *single bit* b, such that every PPT oracle machine $\mathcal{S}^{(C^L)}$ fails to compute this bit (with noticeable probability). This is where our auxiliary input comes into play.

Our first idea was to take the auxiliary input z to be an obfuscation of a circuit that outputs a secret bit b only on inputs that encode a small circuit that agrees with C^L on $(x_1, w_1), \ldots, (x_{t(n)}, w_{t(n)})$. Such an approach is actually taken in the proof of Theorem 31. However, here we need to take a different approach: first, because we need the auxiliary input z to be independent of our obfuscated circuit C^L, and z as defined above does depend on C^L; and second, because we do not want to add the additional assumption that the above z is obfuscatable.

Instead, we let z be an obfuscation of a random filter function $K^L \in_R \mathcal{C}^L$.[6] We show that there exists a string $x^* \in \{0,1\}^*$ (that depends on I_n and $C(I_n)$) such that:

[6]Notice that K^L is independent of C^L.

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C^L \in_R \mathcal{C}_n^L$, and given $\mathcal{O}(K^L)$ it is hard to compute $K(x^*)$.

2. For every $C \in \mathcal{C}_n$, given $(\mathcal{O}(C^L), \mathcal{O}(K^L))$ it is easy to compute $K(x^*)$.

We next present the full proof of Theorem 28. This proof makes use of the following claim, which loosely speaking, asserts the following: Let $\mathcal{O}$ be a weak obfuscator (w.r.t. independent auxiliary input) for the class $\mathcal{C}$. Let $\mathcal{A}$ be a PPT algorithm, that for every $C \in \mathcal{C}$ and for every auxiliary input z, on input $(\mathcal{O}(C), z)$ outputs the function $f(C, z)$, where f is a deterministic (not necessarily Boolean) function. Then there exists a PPT algorithm $\mathcal{S}$, that given auxiliary input z and black-box access to C, computes the function $f(C, z)$ with almost the same probability.

Claim 29 *Let $\mathcal{C}$ be a class of functions, and let $\mathcal{O}$ be a weak obfuscator (w.r.t. independent auxiliary input) for $\mathcal{C}$. Then for every* PPT $\mathcal{A}$*, every deterministic function f (not necessarily Boolean), every sequence of (polynomial size) auxiliary inputs $\{z_n\}$, and every polynomial $p(\cdot)$, there exists a* PPT $\mathcal{S}$ *such that for every $n \in \mathbb{N}$,*

$$\Pr[\mathcal{A}(\mathcal{O}(C), z_n) = f(C, z_n)] - \Pr[\mathcal{S}^C(1^n, z_n) = f(C, z_n)] < \frac{1}{p(n)},$$

where the probabilities are over $C \in_R \mathcal{C}_n$, and over the random coin tosses of $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{O}$.

Proof of Claim 29. Assume for the sake of contradiction that there exists a PPT $\mathcal{A}$, a deterministic function f, a sequence of (polynomial size) auxiliary inputs $\{z_n\}$, and a polynomial $p(\cdot)$ such that for every PPT $\mathcal{S}$ there exists $n \in \mathbb{N}$ such that

$$\Pr[\mathcal{A}(\mathcal{O}(C), z_n) = f(C, z_n)] - \Pr[\mathcal{S}^C(1^n, z_n) = f(C, z_n)] \geq \frac{1}{p(n)},$$

where the probabilities are over $C \in_R \mathcal{C}_n$, and over the random coin tosses of $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{O}$.

Consider a PPT machine $\mathcal{A}$' that on input $(\mathcal{O}(C), z_n, r)$ outputs the inner product $\langle r, \mathcal{A}(\mathcal{O}(C), z_n)\rangle$. Our assumption, together with the work of [17], implies that there

exists a polynomial $\alpha(\cdot)$ such that for every PPT $\mathcal{S}$ there exists $n \in \mathbb{N}$ such that

$$\Pr[\mathcal{A}'(\mathcal{O}(C), z_n, r) = \langle r, f(C, z_n)\rangle] - \Pr[\mathcal{S}^C(1^n, z_n, r) = \langle r, f(C, z_n)\rangle] > \frac{1}{\alpha(n)},$$

where the probabilities are over $C \in_R \mathcal{C}_n$, over the random string r, and over the random coin tosses of $\mathcal{A}$, $\mathcal{S}$ and $\mathcal{O}$. This contradicts the fact that $\mathcal{O}$ is a *weak obfuscator w.r.t. independent auxiliary input* (since the inner product is a Boolean value). ■

We are now ready to present the full proof of Theorem 28, which makes use of Claim 29.

Proof of Theorem 28. Assume for the sake of contradiction that there exists an $\mathcal{NP}$-complete language L and a class of circuits $\mathcal{C}$, that has super-polynomial pseudo entropy over L and is strongly unpredictable, such that the class of circuits $\mathcal{C}^L$ *can* be weakly obfuscated w.r.t. independent auxiliary input, by some obfuscator $\mathcal{O}$.

Fix any polynomial $l(\cdot)$ such that for every $n \in \mathbb{N}$ and for every $C \in \mathcal{C}_n$, $|\mathcal{O}(C^L)| \leq l(n)$. The fact that $\mathcal{C}$ has super-polynomial pseudo entropy over L implies in particular that its pseudo entropy is greater than $l(n) + n$. This in turn implies that there exists a polynomial $t(\cdot)$, and for every $n \in \mathbb{N}$ there exists a set of $t(n)$ elements $I_n \triangleq (x_1 \ldots, x_{t(n)})$, where each $x_i \in L \cap \{0,1\}^n$, and for every $C \in \mathcal{C}_n$ there exists a random variable $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ with (statistical) min entropy greater than $l(n) + n$, such that every PPT oracle machine $M^{C|_{\bar{I}_n}}$ cannot distinguish between the random variable $C(I_n)$ and the random variable $\vec{Y}^C$ (for a randomly chosen $C \in_R \mathcal{C}_n$).

Consider the language $L' \in \mathcal{NP}$ corresponding to the $\mathcal{NP}$-relation $\mathcal{R}_{L'} = \{\mathcal{R}^n_{L'}\}_{n\in\mathbb{N}}$, defined by

$$\mathcal{R}^n_{L'} \stackrel{\text{def}}{=} \{((\vec{x}, \vec{w}, \vec{y}), v) : |\vec{x}| = |\vec{w}| = |\vec{y}| = t(n),\ |v| \leq l(n),\ v(x_i, w_i) = y_i \text{ for } i = 1, \ldots t(n)\},$$

where the witness v is a Boolean circuit.[7] Since $L' \in \mathcal{NP}$ it is reducible to L via a polynomial time reduction. Namely, there exists a polynomial time computable

[7] As was done in Chapter 2, we abuse notations by letting v denote both a circuit and the description of the circuit.

function $\varphi(\cdot)$ such that $(\vec{x}, \vec{w}, \vec{y}) \in L'$ if and only if $\varphi(\vec{x}, \vec{w}, \vec{y}) \in L$. Moreover, there exists a polynomial time computable function $\psi(\cdot)$ such that v is a witness of $(\vec{x}, \vec{w}, \vec{y})$ in L' if and only if $\psi((\vec{x}, \vec{w}, \vec{y}), v)$ is a witness of $\varphi(\vec{x}, \vec{w}, \vec{y})$ in L.[8]

For every $n \in \mathbb{N}$, consider the auxiliary input $z = (I_n, W_n, \mathcal{O}(K^L; r))$, where $I_n = (x_1, \ldots, x_{t(n)})$, $W_n = (w_1 \ldots, w_{t(n)})$ are $t(n)$ witnesses corresponding to I_n in $\mathcal{R}_L$, $K^L \in \mathcal{C}^L_{\tau(n)}$, where $\tau(n) = |\varphi(\vec{x}, \vec{w}, \vec{y})|$ (where $|\vec{x}| = |\vec{w}| = |\vec{y}| = t(n)$), and $r \in \{0,1\}^n$.[9] Claim 29 implies that in order to get a contradiction it suffice to prove the following two statements:

1. For every $n \in \mathbb{N}$, every $C^L \in \mathcal{C}^L_n$, every $K^L \in \mathcal{C}^L_{\tau(n)}$, and every $r \in \{0,1\}^n$, given the pair $(\mathcal{O}(C^L), z)$, where $z = (I_n, W_n, \mathcal{O}(K^L; r))$, it is easy to compute $K(\varphi(I_n, W_n, C(I_n)))$.

2. For every PPT oracle machine $\mathcal{S}$ and for infinitely many n's, $\mathcal{S}^{C^L}(z)$ can compute $K(\varphi(I_n, W_n, C(I_n)))$ only with negligible probability, for random circuits $C \in_R \mathcal{C}_n$ and $K \in_R \mathcal{C}_{\tau(n)}$, and for random $r \in_R \{0,1\}^n$.

It is easy to see that (1) holds since given a pair $(\mathcal{O}(C^L), z)$, where $z = (I_n, W_n, \mathcal{O}(K^L; r))$, $K(\varphi(I_n, W_n, C(I_n)))$ can be easily computed by evaluating

$$K(\varphi(I_n, W_n, C(I_n))) = K^L(\varphi(I_n, W_n, C(I_n)), \psi((I_n, W_n, C(I_n)), \mathcal{O}(C^L)))$$

We argue that (2) holds by using the fact that both $\mathcal{O}(C^L)$ and $\mathcal{O}(K^L)$ are weak obfuscations w.r.t. independent auxiliary input.

Assume for the sake of contradiction that (2) does not hold. Then, there exists a PPT oracle machine $\mathcal{S}_1$ and a polynomial $\alpha(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_1^{C^L}(z) = K(\varphi(I_n, W_n, C(I_n)))] \geq \frac{1}{\alpha(n)}$$

[8] This witness preserving property does not necessarily hold for every $\mathcal{NP}$-complete language (though it does hold for most of the $\mathcal{NP}$-complete languages that we are aware of). We assume that our $\mathcal{NP}$-complete language L is witness preserving.

[9] For simplicity, we assume without loss of generality that $\mathcal{O}(K^L)$ uses n bits of randomness.

(where the probability is over $C \in_R \mathcal{C}_n$, $K \in_R \mathcal{C}_{\tau(n)}$, and $r \in_R \{0,1\}^n$).[10] Thus, there exists a PPT oracle machine $\mathcal{S}_2$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{C|_{\bar{I}_n}}(z, C(I_n)) = K(\varphi(I_n, W_n, C(I_n)))] \geq \frac{1}{\alpha(n)}$$

(where the probability is over $C \in_R \mathcal{C}_n$, $K \in_R \mathcal{C}_{\tau(n)}$, and $r \in_R \{0,1\}^n$, and where $C|_{\bar{I}_n}$ is a circuit that on input $x \notin I_n$ outputs $C(x)$, and on input $x \in I_n$ outputs $\perp$).

Next we replace the value $C(I_n)$ with the random variable $\vec{Y}^C$. Notice that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{C|_{\bar{I}_n}}(z, \vec{Y}^C)) = K(\varphi(I_n, W_n, \vec{Y}^C))] \geq \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is over $C \in_R \mathcal{C}_n$, $K \in_R \mathcal{C}_{\tau(n)}$, $r \in_R \{0,1\}^n$, and over the random variable $\vec{Y}^C$). This is so since otherwise $\mathcal{S}_2^{C|_{\bar{I}_n}}$ can be used to distinguish between the random variable $C(I_n)$ and the random variable $\vec{Y}^C$ (for a randomly chosen $C \in_R \mathcal{C}_n$).

This implies that there exists a PPT machine $\mathcal{S}_3$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_3(C, z, \vec{Y}^C) = K(\varphi(I_n, W_n, \vec{Y}^C))] \geq \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is over $C \in_R \mathcal{C}_n$, $K \in_R \mathcal{C}_{\tau(n)}$, $r \in_R \{0,1\}^n$, and over the random variable $\vec{Y}^C$).

Recall that $z = (I_n, W_n, \mathcal{O}(K^L; r))$, where $\mathcal{O}$ is a weak obfuscator w.r.t. independent auxiliary input. Thus, from the definition of weak obfuscation w.r.t. independent auxiliary input, we conclude that there exists a PPT oracle machine $\mathcal{S}_4$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_4^{K^L}(C, I_n, W_n, \vec{Y}^C) = K(\varphi(\mathcal{I}_n, W_n, \vec{Y}^C))] \geq \frac{1}{2\alpha(n)}$$

[10] For simplicity, we assume without loss of generality that $\mathcal{S}_1$ is deterministic.

(where the probability is over $C \in_R \mathcal{C}_n$, $K \in_R \mathcal{C}_{\tau(n)}$, and over the random variable $\vec{Y}^C$). The fact that $\mathcal{C}$ is strongly unpredictable, together with the definition of K^L, implies that the only way an oracle machine $\mathcal{S}^{K^L}$ can compute $K(x)$, for some element $x \in L$, is by querying its oracle with $(x, w) \in \mathcal{R}_L$. Thus, there exists a PPT machine $\mathcal{S}_5$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_5(C, I_n, W_n, \vec{Y}^C) = (\varphi(I_n, W_n, \vec{Y}^C), w^*) \in \mathcal{R}_L] \geq \frac{1}{2\alpha(n)}$$

(where the probability is over $C \in_R \mathcal{C}_n$ and over the random variable $\vec{Y}^C$). Recall that L is an $\mathcal{NP}$-complete language and φ is an $\mathcal{NP}$-reduction from L' to L. Thus, it is easy to compute a witness for $(I_n, W_n, \vec{Y}^C) \in L'$ from a witness for $\varphi(I_n, W_n, \vec{Y}^C) \in L$, and vice versa.[11] Therefore, there exists a PPT machine $\mathcal{S}_6$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_6(C, I_n, W_n, \vec{Y}^C) = v : |v| \leq l(n),\ v(x_i, w_i) = Y_i \text{ for } i = 1, \ldots, t(n)] \geq \frac{1}{2\alpha(n)}$$

(where $I_n = (x_1, \ldots, x_{t(n)})$, $W_n = (w_1, \ldots, w_{t(n)})$ and the probability is over $C \in_R \mathcal{C}_n$ and over the random variable $\vec{Y}^C \stackrel{\text{def}}{=} (Y_1, \ldots, Y_{t(n)})$).

This contradicts the fact that $\vec{Y}^C$ has min entropy at least $l(n) + n$. Formally, we employ a simple counting argument. Notice that there can be at most $2^{l(n)}$ distinct values of $\vec{Y}^C$ for which there exists a value $v \in \{0,1\}^{l(n)}$ such that $v(x_i, w_i) = Y_i$ for $i = 1, \ldots, t(n)$. Thus, the fact that there exists such a value v with probability at least $\frac{1}{2\alpha(n)}$ over the values of $\vec{Y}^C$, implies that $\vec{Y}^C$ obtains one of these $2^{l(n)}$ values with probability at least $\frac{1}{2\alpha(n)}$, which in turn implies that some values of $\vec{Y}^C$ are obtained with probability at least $\frac{1}{2\alpha(n)2^{l(n)}}$. This implies that $\vec{Y}^C$ has (statistical) min-entropy at most $l(n) + \log(2\alpha(n)) < l(n) + n$, contradicting our assumption that $\vec{Y}^C$ has (statistical) min-entropy at least $l(n) + n$. ■

[11] Recall that we assume that our $\mathcal{NP}$-complete language is witness preserving.

4.5 Impossibility of Obfuscation w.r.t. Dependent Auxiliary Input

In this subsection we define the classes of *point-filter functions*. We show that if *point-filter functions* can be weakly obfuscated w.r.t. dependent auxiliary input then *every* class of circuits with super-polynomial pseudo entropy cannot be weakly obfuscated w.r.t. dependent auxiliary input.

Definition 30 *For every $L \in \mathcal{NP}$, the class $\Delta^L = \{\Delta^L_n\}_{n\in\mathbb{N}}$ is a class of* point-filter functions*, where $\Delta^L_n = \{\delta_{x,b}\}_{x\in\{0,1\}^n, b\in\{0,1\}}$ and each function $\delta_{x,b}$ is defined by*

$$\delta_{x,b}(w) \stackrel{\text{def}}{=} \begin{cases} (x,b) & \text{if } (x,w) \in \mathcal{R}_L \\ x & \text{otherwise} \end{cases}$$

Theorem 31 *At least one of the following conditions hold:*

1. *For every $\mathcal{NP}$-complete language L the class Δ^L is not weakly obfuscatable w.r.t. dependent auxiliary input.*
2. *Every class $\mathcal{C}$ with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input.*

Before presenting the full proof of Theorem 31, we sketch the main ideas.

Proof Sketch. Assume that condition (1) does not hold. Namely, assume that there exists an $\mathcal{NP}$-complete language L such that the class Δ^L is weakly obfuscatable w.r.t. dependent auxiliary input. This implies that for every $\mathcal{NP}$ language L' the class $\Delta^{L'}$ is also weakly obfuscatable w.r.t. dependent auxiliary input. (This follows from the existence of an $\mathcal{NP}$-reduction.[12])

We prove that condition (2) holds. Namely, we prove that every class $\mathcal{C}$ with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input. Assume for the sake of contradiction that there exists a class $\mathcal{C}$ with

[12] As in footnote 8, we actually assume that the $\mathcal{NP}$-reduction is witness preserving.

super-polynomial pseudo entropy that can be weakly obfuscated w.r.t. dependent auxiliary input by an obfuscator $\mathcal{O}$.

The main idea is to exploit the fact for every $n \in \mathbb{N}$ there is a polynomial size set $I_n \subseteq \{0,1\}^n$, such that:

1. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C \in_R \mathcal{C}_n$, it is hard to come up with a "small" circuit that computes C on I_n.

2. For every $C \in \mathcal{C}_n$, $\mathcal{O}(C)$ is itself a "small" circuit that computes C on I_n.

We want to exploit this difference to obtain a contradiction. However, for this we need to show that given $\mathcal{O}(C)$ it is easy to compute a *single bit* b, whereas every PPT oracle machine $\mathcal{S}^C$ fails to compute this bit (with noticeable probability). This is where the auxiliary input comes into play.

We let the auxiliary input $z = z_C$ be a point-filter function that is associated with a secret bit b. z_C outputs its secret bit b only on inputs that encode a "small" circuit that agrees with C on the elements in I_n. More precisely, z_C is the point filter function $\delta_{(I_n,C(I_n)),b}$, where a valid witness for $(I_n, C(I_n))$ is a "small" circuit that agrees with C on all the elements in I_n.

Thus, it remains to show:

1. For every $C \in \mathcal{C}_n$, given $(\mathcal{O}(C), z_C)$ it is easy to compute the secret bit b.

2. For every PPT $\mathcal{S}$, given black-box access to a random circuit $C \in_R \mathcal{C}_n$, and given z_C it is hard to compute the secret bit b.

It is easy to see that (1) holds, since $\mathcal{O}(C)$ is a valid witness of $(I_n, C(I_n))$. Thus, given $(\mathcal{O}(C), z_C)$, the secret bit b can be easily computed by evaluating z_C on input $\mathcal{O}(C)$. However, it is not clear that (2) holds, since it may actually be easy to extract the secret bit b from z_C, which can be any circuit computing $\delta_{(I_n,C(I_n)),b}$. To hide the secret bit b from $\mathcal{S}$, we obfuscate this point-filter function. Namely, the auxiliary input z_C that we consider is an obfuscation of $\delta_{(I_n,C(I_n)),b}$. Now, it is easy to see intuitively that (2) holds since $\mathcal{S}$ does not have any valid witness of $(I_n, C(I_n))$, and

therefore does not have any advantage in guessing the secret bit b from its obfuscated point-filter function at hand.

We proceed by presenting the full proof of Theorem 31.

Proof of Theorem 31. Assume that every class of point-filter functions is weakly obfuscatable w.r.t. dependent auxiliary input. Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be any class of poly-size circuits that has super-polynomial pseudo entropy. Assume for the sake of contradiction that $\mathcal{C}$ is weakly obfuscatable w.r.t. dependent auxiliary input, by some obfuscator $\mathcal{O}_1$. Then by definition, for every PPT $\mathcal{A}$ and every polynomials $p(\cdot)$ and $q(\cdot)$, there exists a PPT $\mathcal{S}$ such that for every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, and for every auxiliary input z of size at most $q(n)$ (that may depend on C),

$$\left|\Pr[\mathcal{A}(\mathcal{O}_1(C), z) = 1] - \Pr[\mathcal{S}^C(1^n, z) = 1]\right| < \frac{1}{p(n)}.$$

Fix any polynomial $l(\cdot)$ such that for every $n \in \mathbb{N}$ and for every $C \in \mathcal{C}_n$, $|\mathcal{O}_1(C)| \leq l(n)$. Recall that $\mathcal{C}$ has super-polynomial pseudo entropy, and in particular its pseudo entropy is greater than $l(n) + n$. This implies that there exists a polynomial $t(\cdot)$, for every $n \in \mathbb{N}$ there exists a set of $t(n)$ values $I_n \triangleq (x_1 \ldots, x_{t(n)})$, and for every $C \in \mathcal{C}_n$ there exists a random variable $\vec{Y}^C = (Y_1, \ldots, Y_{t(n)})$ with (statistical) min entropy greater than $l(n) + n$, such that every PPT oracle machine $M^{C|_{\bar{I}_n}}$ cannot distinguish between the random variable $C(I_n)$ and the random variable $\vec{Y}^C$ (where $C \in_R \mathcal{C}_n$).

Consider the language $L' \in \mathcal{NP}$ corresponding to the $\mathcal{NP}$-relation $\mathcal{R}_{L'} = \{\mathcal{R}^n_{L'}\}_{n\in\mathbb{N}}$, defined by

$$\mathcal{R}^n_{L'} \stackrel{\text{def}}{=} \{((\vec{x}, \vec{y}), v) : |\vec{x}| = |\vec{y}| = t(n),\ |v| \leq l(n),\ v(x_i) = y_i\},^{13}$$

and consider the class of point-filter functions $\Delta^{L'} = \{\delta_{(\vec{x},\vec{y}),b}\}$. From our assumption, there exists a weak obfuscator w.r.t. dependent auxiliary input for the class $\Delta^{L'}$. We denote this obfuscator by $\mathcal{O}_2$.

For any $n \in \mathbb{N}$ and any $C \in \mathcal{C}_n$, we consider the auxiliary input $z = z(C, b, r)$,

[13] As in the proof of Theorem 28, we abuse notations by letting v denote both a Boolean circuit and the description of the circuit.

which is a weak obfuscation of the point-filter function $\delta_{(I_n,C(I_n)),b}$ with respect to randomness $r \in \{0,1\}^n$.[14] Namely, $z(C,b,r) \triangleq \mathcal{O}_2(\delta_{(I_n,C(I_n)),b};r)$.

In order to get a contradiction it suffice to prove that the following two statements are true:

1. For every $n \in \mathbb{N}$, every $C \in \mathcal{C}_n$, every bit b, and every $r \in \{0,1\}^n$, given the pair $(\mathcal{O}_1(C), z(C,b,r))$ it is easy to compute the secret bit b.

2. For every PPT oracle machine $\mathcal{S}$ and for infinitely many n's, $\mathcal{S}^C(1^n, z(C,b,r))$ does not have any advantage in guessing the secret bit b, for a random circuit $C \in_R \mathcal{C}_n$, a random bit $b \in_R \{0,1\}$ and random $r \in_R \{0,1\}^n$.

It is easy to see that (1) holds since given a pair $(\mathcal{O}_1(C), z(C,b,r))$, the secret bit b can be easily computed by simply evaluating $z(C,b,r)$ on input $\mathcal{O}_1(C)$. We argue that (2) holds by using the fact that both $\mathcal{O}_1(C)$ and z_C are weak obfuscations w.r.t. dependent auxiliary input.

Assume for the sake of contradiction that (2) does not hold. Then, there exists a PPT oracle machine $\mathcal{S}_1$ and a polynomial $\alpha(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_1^C(z(C,b,r)) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)}$$

(where the probability is over $C \in_R \mathcal{C}_n$, $b \in_R \{0,1\}$ and $r \in_R \{0,1\}^n$).[15] Since $z(C,b,r)$ is an obfuscation of the point-filter function $\delta_{(I_n,C(I_n)),b}$, the values I_n and $C(I_n)$ are public, and thus there exists a PPT oracle machine $\mathcal{S}_2$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{C|_{\bar{I}_n}}(z(C,b,r)) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)}$$

(where the probability is over $C \in_R \mathcal{C}_n$, $b \in_R \{0,1\}$ and $r \in_R \{0,1\}^n$). Next, we replace the auxiliary input $z(C,b,r) = \mathcal{O}_2(\delta_{(I_n,C(I_n)),b};r)$ with the auxiliary input

[14] For simplicity, we assume without loss of generality that $\mathcal{O}_2$ uses n bits of randomness for every function in $\Delta_n^{L'}$.

[15] For simplicity, we assume without loss of generality that $\mathcal{S}_1$ is deterministic.

$z'(C,b,r) = \mathcal{O}_2(\delta_{(I_n,\vec{Y}^C),b}; r)$. Notice that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{C|_{\bar{I}_n}}(z'(C,b,r)) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is over $C \in_R \mathcal{C}_n$, $b \in_R \{0,1\}$, $r \in_R \{0,1\}^n$, and over the random variable $\vec{Y}^C$). This is so since otherwise, $\mathcal{S}_2^{C|_{\bar{I}_n}}$ can be used to distinguish between $C(I_n)$ and $\vec{Y}^C$, contradicting the pseudo entropy condition.

Thus, there exists a PPT machine $\mathcal{S}_3$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_3(z'(C,b,r), C) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is also over $C \in_R \mathcal{C}_n$, $b \in_R \{0,1\}$, $r \in_R \{0,1\}^n$, and over the random variable $\vec{Y}^C$).

Recall that $z'(C,b,r)$ is a weak obfuscation (w.r.t. dependent auxiliary input) of $\delta_{(I_n,\vec{Y}^C),b}$. Thus, from the definition of weak obfuscation (w.r.t. dependent auxiliary input) we conclude that there exists a PPT oracle machine $\mathcal{S}_4$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_4^{\delta_{(I_n,\vec{Y}^C),b}}(C) = b] \geq \frac{1}{2} + \frac{1}{2\alpha(n)}$$

(where the probability is also over $C \in_R \mathcal{C}_n$, $b \in_R \{0,1\}$, and over the random variable $\vec{Y}^C$).

Recall that a point-filter function $\delta_{(\vec{x},\vec{y}),b}$ reveals its secret bit b only on inputs v that are circuits of size $l(n)$ such that $v(x_i) = y_i$. Thus, the only way for a PPT oracle machine $\mathcal{S}^{\delta_{(\vec{x},\vec{y}),b}}$ to guess the bit b with a noticeable advantage is by finding, with non-negligible probability, a circuit of size $l(n)$ that on input x_i outputs y_i. This, together with the above inequality, implies that there exists a PPT machine $\mathcal{S}_5$ and a polynomial $\beta(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_5(C, I_n, (y'_1, \ldots, y'_{t(n)})) = v : |v| = l(n) \ \wedge \ v(x_i) = y'_i \text{ for } i = 1, \ldots, t(n)] \geq \frac{1}{\beta(n)}$$

(where $I_n = (x_1, \ldots, x_{t(n)})$ and where the probability is over $C \in_R \mathcal{C}_n$ and over the

values $(y'_1, \ldots, y'_{t(n)})$ assumed by the random variable $\vec{Y}^C$). This implies that for every $n \in \mathbb{N}$ there exists $C \in \mathcal{C}_n$ such that

$$\Pr[\mathcal{S}_5(C, I_n, (y'_1, \ldots, y'_{t(n)})) = v : |v| = l(n) \ \wedge \ v(x_i) = y'_i \text{ for } i = 1, \ldots, t(n)] \geq \frac{1}{\beta(n)}$$

(where $I_n = (x_1, \ldots, x_{t(n)})$ and where the probability is over the values $(y'_1, \ldots, y'_{t(n)})$ assumed by the random variable $\vec{Y}^C$).

This contradicts the fact that $\vec{Y}^C$ has min entropy at least $l(n) + n$. Formally, we employ a simple counting argument. Notice that there can be at most $2^{l(n)}$ distinct values of $(y'_1, \ldots, y'_{t(n)})$ for which there exists a circuit v of size $l(n)$ such that $v(x_i) = y'_i$ for $i = 1, \ldots, t(n)$. Thus, the fact that there exists such a circuit v with probability at least $\frac{1}{\beta(n)}$ over the values of $(y'_1, \ldots, y'_{t(n)})$, implies that some of these values are obtained with probability at least $\frac{1}{\beta(n)2^{l(n)}}$, which implies that $\vec{Y}^C$ has (statistical) min-entropy at most $l(n) + \log(\beta(n)) < l(n) + n$, contradicting our assumption that $\vec{Y}^C$ has (statistical) min-entropy at least $l(n) + n$. ■

Using similar ideas, we can prove the following theorem.

Theorem 32 *At least one of the following conditions hold.*

1. *For every $\mathcal{NP}$-complete language L the class Δ^L is not weakly obfuscatable w.r.t. dependent auxiliary input.*
2. *For* every *CCA2 secure* (*secret-key or public-key*) *encryption scheme* $(\mathsf{GEN}, \mathsf{ENC}, \mathsf{DEC})$, *the class of decryption algorithms* $\mathsf{DEC} = \{\mathsf{DEC}_{\mathsf{SK}}\}$ *is not weakly obfuscatable w.r.t. dependent auxiliary input.*

We prove Theorem 32 for the case of secure public-key encryption schemes. The proof for the case of secure secret-key encryption schemes is very similar, and is therefore omitted.

Proof of Theorem 32. Assume for the sake of contradiction that both condition (1) and condition (2) do not hold. Namely, assume that every class of point-filter functions is weakly obfuscatable w.r.t. dependent auxiliary input, and yet there exists

a CCA2 secure encryption scheme $(\mathsf{GEN}, \mathsf{ENC}, \mathsf{DEC})$, such that the class of decryption algorithms $\{\mathsf{DEC}_{\mathsf{SK}}\}$ is also weakly obfuscatable w.r.t. dependent auxiliary input, by some obfuscator $\mathcal{O}_1$. By definition, for every PPT $\mathcal{A}$ and every polynomials $p(\cdot)$ and $q(\cdot)$ there exists a PPT $\mathcal{S}$ such that for every $n \in \mathbb{N}$, every $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$, and for every auxiliary input z of size at most $q(n)$ (that may depend on SK),

$$|\Pr[\mathcal{A}(\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}}), z) = 1] - \Pr[\mathcal{S}^{\mathsf{DEC}_{\mathsf{SK}}}(1^n, z) = 1]| < \frac{1}{p(n)}.$$

Fix any polynomial $l(\cdot)$ such that for every $n \in \mathbb{N}$ and for every $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$, it holds that $|\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}})| \leq l(n)$. Let $t(\cdot)$ be any polynomial such that for every $n \in \mathbb{N}$, $t(n) > l(n) + n$.

For every $n \in \mathbb{N}$, let $\vec{\mathsf{M}} = \{\mathsf{M}_1, \ldots, \mathsf{M}_{t(n)}\}$ be any $t(n)$ messages from the message space. For every $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$ let $I = I(\mathsf{PK}, \pi, \vec{r}) = (\mathsf{C}_{\pi(1)}, \ldots, \mathsf{C}_{\pi(t(n))})$, where $\vec{r} = (r_1, \ldots, r_{t(n)})$, $\mathsf{C}_i = \mathsf{ENC}_{\mathsf{PK}}(\mathsf{M}_i; r_i)$, and π is a permutation over $\{1, \ldots, t(n)\}$.

The fact that $(\mathsf{GEN}, \mathsf{ENC}, \mathsf{DEC})$ is a CCA2 secure encryption scheme implies that every PPT oracle machine $M^{\mathsf{DEC}_{\mathsf{SK}}|_{\bar{I}}}$ cannot distinguish between the pair $(\vec{\mathsf{M}}_\pi, \vec{\mathsf{C}}_\pi)$ and the pair $(\vec{\mathsf{M}}_{\pi'}, \vec{\mathsf{C}}_\pi)$, where $\vec{\mathsf{M}}_\pi = (\mathsf{M}_{\pi(1)}, \ldots, \mathsf{M}_{\pi(t(n))})$, $\vec{\mathsf{M}}_{\pi'} = (\mathsf{M}_{\pi'(1)}, \ldots, \mathsf{M}_{\pi'(t(n))})$, $\vec{\mathsf{C}}_\pi = I(\mathsf{PK}, \pi, \vec{r})$, $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, π, π' are independent random permutations over $\{1, \ldots, t(n)\}$, and $r_1, \ldots, r_{t(n)} \in_R \{0,1\}^n$.

Consider the language $L' \in \mathcal{NP}$ corresponding to the $\mathcal{NP}$-relation $\mathcal{R}_{L'} = \{\mathcal{R}^n_{L'}\}_{n \in \mathbb{N}}$ defined by

$$\mathcal{R}^n_{L'} \stackrel{\text{def}}{=} \{((\vec{x}, \vec{y}), v) : |\vec{x}| = |\vec{y}| = t(n),\ |v| \leq l(n),\ v(x_i) = y_i\},$$

and consider the class of point-filter functions $\Delta^{L'} = \{\delta_{(\vec{x},\vec{y}),b}\}$. From our assumption, there exists a weak obfuscator w.r.t. dependent auxiliary input for the class $\Delta^{L'}$. We denote this obfuscator by $\mathcal{O}_2$.

For any $n \in \mathbb{N}$, any $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$, any permutation π, any bit b, and any $\vec{r}$ and $\vec{r}'$, consider the auxiliary input $z = z(\mathsf{PK}, \pi, \vec{r}, b, r') = \mathcal{O}_2(\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_\pi), b}; r')$ which is a weak obfuscation (w.r.t. dependent auxiliary input) of the point-filter function $\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_\pi), b}$, where $\vec{\mathsf{C}}_\pi = I(\mathsf{PK}, \pi, \vec{r})$. In order to get a contradiction it suffice to prove

the following two statements:

1. For every $n \in \mathbb{N}$, every $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$, every permutation π, every bit b, and every $\vec{r}$ and r', given a pair $(\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}}), z(\mathsf{PK}, \pi, \vec{r}, b, r'))$, it is easy to compute the secret bit b.

2. For every PPT oracle machine $\mathcal{S}$ and for infinitely many n's, $\mathcal{S}^{\mathsf{DEC}_{\mathsf{SK}}}(z(\mathsf{PK}, \pi, \vec{r}, b, r'))$ does not have any advantage in guessing the secret bit b, for $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutation π, random bit b, and random $\vec{r}$ and r'.

It is easy to see that (1) holds since given a pair $(\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}}), z(\mathsf{PK}, \pi, \vec{r}, b, r'))$, where $z(\mathsf{PK}, \pi, \vec{r}, b, r') = \mathcal{O}_2(\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_\pi), b}; r')$, the secret bit b can be easily computed by simply evaluating $z(\mathsf{PK}, \pi, \vec{r}, b, r')$ on input $\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}})$. We argue that (2) holds by using the fact that both $\mathcal{O}_1(\mathsf{DEC}_{\mathsf{SK}})$ and $z(\mathsf{PK}, \pi, \vec{r}, b, r')$ are weak obfuscations w.r.t. dependent auxiliary input, as follows.

Assume for the sake of contradiction that (2) does not hold. Then, there exists a PPT oracle machine $\mathcal{S}_1$ and a polynomial $\alpha(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_1^{\mathsf{DEC}_{\mathsf{SK}}}(z(\mathsf{PK}, \pi, \vec{r}, b, r')) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)}$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutation π, random bit b, and random $\vec{r}$ and r').[16]

Since $z(\mathsf{PK}, \pi, \vec{r}, b, r')$ is an obfuscation of the point-filter function $\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_\pi), b}$, the values $\vec{\mathsf{C}}_\pi = I(\mathsf{PK}, \pi, \vec{r})$ and $\vec{\mathsf{M}}_\pi$ are public, and thus there exists a PPT oracle machine $\mathcal{S}_2$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{\mathsf{DEC}_{\mathsf{SK}}|_{\bar{I}}}(z(\mathsf{PK}, \pi, \vec{r}, b, r')) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)}$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutation π, random bit b, and random $\vec{r}$ and r').[17]

[16] For simplicity, we assume without loss of generality that $\mathcal{S}_1$ is deterministic.

[17] Recall that we abuse notations by letting I denote both the vector $\vec{\mathsf{C}}_\pi$ and the set $\{\mathsf{C}_1, \ldots, \mathsf{C}_{t(n)}\}$.

Next, we replace the auxiliary input $z(\mathsf{PK}, \pi, \vec{r}, b, r') = \mathcal{O}_2(\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_\pi), b}; r')$ with the auxiliary input $z'(\mathsf{PK}, \pi, \pi', \vec{r}, b, r') = \mathcal{O}_2(\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_{\pi'}), b}; r')$, where $\vec{\mathsf{C}}_\pi = I(\mathsf{PK}, \pi, \vec{r})$. Notice that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_2^{\mathsf{DEC}_{\mathsf{SK}}|_{\bar{I}}}(z'(\mathsf{PK}, \pi, \pi', \vec{r}, b, r') = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutations π, π', random bit b, and random $\vec{r}$ and r'). This is so since otherwise, $\mathcal{S}_2^{\mathsf{DEC}_{\mathsf{SK}}|_{\bar{I}}}$ can be used to distinguish between the pair $(\vec{\mathsf{M}}_\pi, \vec{\mathsf{C}}_\pi)$ and the pair $(\vec{\mathsf{M}}_{\pi'}, \vec{\mathsf{C}}_\pi)$, contradicting the security of the encryption scheme.

Thus, there exists a PPT machine $\mathcal{S}_3$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_3(\mathsf{DEC}_{\mathsf{SK}}, z'(\mathsf{PK}, \pi, \pi', \vec{r}, b, r')) = b] \geq \frac{1}{2} + \frac{1}{\alpha(n)} - \mathsf{negl}(n)$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutations π, π', random bit b, and random $\vec{r}$ and r').

Recall that $z'(\mathsf{PK}, \pi, \pi', \vec{r}, b, r')$ is a weak obfuscation (w.r.t. dependent auxiliary input) of $\delta_{(\vec{\mathsf{C}}_\pi, \vec{\mathsf{M}}_{\pi'}), b}$. Thus, from the definition of weak obfuscation (w.r.t. dependent auxiliary input) we conclude that there exists a PPT oracle machine $\mathcal{S}_4$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_4^{\delta_{(\vec{\mathsf{C}_\pi}, \vec{\mathsf{M}}_{\pi'}), b}}(\mathsf{DEC}_{\mathsf{SK}}) = b] \geq \frac{1}{2} + \frac{1}{2\alpha(n)}$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutations π, π', random bit b, and random $\vec{r}$).

Recall that the point-filter function $\delta_{(\vec{x}, \vec{y}), b}$ reveals its secret bit b only on inputs v which are circuits of size $l(n)$ such that $v(x_i) = y_i$. Thus, the only way for a PPT oracle machine $\mathcal{S}^{\delta_{(\vec{x}, \vec{y}), b}}$ to guess the bit b with a noticeable advantage is by finding, with non-negligible probability, a circuit of size $l(n)$ that on input x_i outputs y_i. This, together with the above inequality, implies that there exists a PPT machine $\mathcal{S}_5$ and

a polynomial $\beta(\cdot)$ such that for every $n \in \mathbb{N}$,

$$\Pr[\mathcal{S}_5(\mathsf{DEC}_{\mathsf{SK}}, \{\mathsf{C}_{\pi(i)}, \mathsf{M}_{\pi'(i)}\}_{i=1}^{t(n)}) = v : |v| = l(n) \ \wedge \ v(\mathsf{C}_{\pi(i)}) = \mathsf{M}_{\pi'(i)}] \geq \frac{1}{\beta(n)}$$

(where the probability is over $(\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{GEN}(1^n)$, random permutations π, π', and random $\vec{r}$). This implies that for every $n \in \mathbb{N}$ there exists $(\mathsf{SK}, \mathsf{PK}) \in \mathsf{GEN}(1^n)$, a permutation π, and values $\vec{r}$ such that

$$\Pr[\mathcal{S}_5(\mathsf{DEC}_{\mathsf{SK}}, \{\mathsf{C}_{\pi(i)}, \mathsf{M}_{\pi'(i)}\}_{i=1}^{t(n)}) = v : |v| = l(n) \ \wedge \ v(\mathsf{C}_{\pi(i)}) = \mathsf{M}_{\pi'(i)}] \geq \frac{1}{\beta(n)}$$

(where the probability is over the random permutation π').

This contradicts the fact that each π' is chosen with probability $\frac{1}{t(n)!}$. Formally, we employ a simple counting argument. Notice that there can be at most $2^{l(n)}$ distinct values of $(\mathsf{M}_{\pi'(1)}, \ldots, \mathsf{M}_{\pi'(t(n))})$ for which there exists a value $v \in \{0,1\}^{l(n)}$ such that $v(\mathsf{C}_{\pi(i)}) = \mathsf{M}_{\pi'(i)}$ for $i = 1, \ldots, t(n)$. Thus, the fact that there exists such a value v with probability at least $\frac{1}{\beta(n)}$ over the values of $(\mathsf{M}_{\pi'(1)}, \ldots, \mathsf{M}_{\pi'(t(n))})$, implies that some values of $(\mathsf{M}_{\pi'(1)}, \ldots, \mathsf{M}_{\pi'(t(n))})$ are obtained with probability at least $\frac{1}{\beta(n)2^{l(n)}}$, contradicting the assumption that each such sequence is obtained with probability $\frac{1}{t(n)!} < \frac{1}{2^{t(n)}} < \frac{1}{2^{l(n)+n}} < \frac{1}{\beta(n)2^{l(n)}}$. ■

4.5.1 Are Point-filter Functions Obfuscatable?

Theorems 31 and 32 both give conditional results. We would much prefer to have the explicit result that every class with super-polynomial pseudo entropy is not weakly obfuscatable w.r.t. dependent auxiliary input, which would imply that many natural cryptographic tasks (such as pseudo-random functions, encryption algorithms, decryption algorithms, signature algorithms, etc.) cannot be weakly obfuscated w.r.t. dependent auxiliary input. Therefore, we think that it is worth investigating the question of whether point-filter functions are weakly obfuscatable w.r.t. dependent auxiliary input. We do not have a complete answer to this question. Rather, we

relate it to another (seemingly unrelated) problem that is of independent interest. We show that the existence of a weak obfuscator w.r.t. dependent auxiliary input for the class Δ^L is related to the existence of a *hard-core predicate for the language* L.

Intuitively, $B(\cdot)$ is a hard-core predicate for the language L if the following two conditions hold: (1) for every $x \in L$, given any witness of x it is easy to compute $B(x)$. (2) It is hard to compute $B(x)$ without knowing a witness for x. We formalize the hard-core predicate $B(\cdot)$ as a probabilistic predicate, as follows.

Definition 33 (Hard-core predicate for L): *A randomized predicate $B(\cdot)$ is said to be a* hard-core predicate *for $L \in \mathcal{NP}$, if the following two conditions hold:*

1. *There exists a* PPT *machine $\mathcal{A}_1$ and a polynomial $p(\cdot)$ such that for every $(x, w) \in \mathcal{R}_L$ and every $r \in \{0,1\}^{|x|}$,*

$$\Pr[\mathcal{A}_1(x, w, r) = B(x, r)] \geq \frac{1}{2} + \frac{1}{p(|x|)},$$

   where the probability is over the random coin tosses of $\mathcal{A}_1$.

2. *There exists a* PPT *oracle machine $\mathcal{A}_2$ such that for every polynomial $q(\cdot)$ there exists a polynomial $p(\cdot)$ such that for every $x \in L$ and every function f_x, that on input a random r outputs $B(x, r)$ with probability $\frac{1}{2} + \frac{1}{q(|x|)}$, it holds that*

$$\Pr[\mathcal{A}_2^{f_x}(x) = w \text{ s.t. } (x, w) \in \mathcal{R}_L] \geq \frac{1}{p(|x|)},$$

   where the probability is over the random coin tosses of $\mathcal{A}_2$.

Theorem 34 *If $L \in \mathcal{NP}$ has a hard-core predicate then the class Δ^L is weakly obfuscatable w.r.t. dependent auxiliary input.*

Proof of Theorem 34. Assume that $L \in \mathcal{NP}$ has a hard-core predicate $B(\cdot)$. Then, there exist algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ as above. By applying standard amplification techniques we can assume without loss of generality that for every $(x, w) \in \mathcal{R}_L$ and every $r \in \{0,1\}^{|x|}$,

$$\Pr[\mathcal{A}_1(x, w, r) = B(x, r)] \geq 1 - \frac{1}{2^{2\ell(|x|)}},$$

where the probability is over the random coin tosses of $\mathcal{A}_1$, and where $\ell(|x|)$ is an upper bound on the length of the witnesses of x.

Consider the (obfuscation) function $\mathcal{O}$, that on input an (un-obfuscated) point-filter function $\delta_{x,b}$ and randomness $r \in_R \{0,1\}^{|x|}$, generates an obfuscation $\mathcal{O}(\delta_{x,b}; r)$ of the point-filter function $\delta_{x,b}$, that operates as follows: It has the values x, r and $b' = \mathcal{A}_1(x, w, r) \oplus b$ hard-wired into it.[18] On any input w', $\mathcal{O}(\delta_{x,b}; r)$ checks whether $(x, w') \in \mathcal{R}_L$. If the check is not satisfied then it outputs x. If the check is satisfied, then it outputs the bit $b'' = \mathcal{A}_1(x, w', r) \oplus b'$ together with x.

In order to show that $\mathcal{O}$ is a weak obfuscator w.r.t. dependent auxiliary input for the class of point-filter functions Δ^L, we need to show that the following three conditions are satisfied:

- Functionality Condition: Notice that for every $x \in \{0,1\}^n$, every $b \in \{0,1\}$, and every w' such that $(x, w') \in \mathcal{R}_L$,

$$\begin{aligned}
&\Pr[\mathcal{O}(\delta_{x,b}; r)(w') = \delta_{x,b}(w')] \\
&= \Pr[\mathcal{A}_1(x, w', r) \oplus \mathcal{A}_1(x, w, r) \oplus b = b] \\
&= \Pr[\mathcal{A}_1(x, w', r) = \mathcal{A}_1(x, w, r)] \\
&\geq \Pr[\mathcal{A}_1(x, w', r) = \mathcal{A}_1(x, w, r) = B(x, r)] \\
&= 1 - \Pr[\mathcal{A}_1(x, w', r) \neq B(x, r) \vee \mathcal{A}_1(x, w, r) \neq B(x, r)] \\
&\geq 1 - \Pr[\mathcal{A}_1(x, w', r) \neq B(x, r)] - \Pr[\mathcal{A}_1(x, w, r) \neq B(x, r)] \\
&\geq 1 - \tfrac{2}{2^{2\ell(n)}}.
\end{aligned}$$

  Thus, for every $x \in \{0,1\}^n$ and $b \in \{0,1\}$,

$$\begin{aligned}
&\Pr[\forall w' \ \mathcal{O}(\delta_{x,b}; r)(w') = \delta_{x,b}(w')] \\
&= 1 - \Pr[\exists w' \text{ s.t. } \mathcal{O}(\delta_{x,b}; r)(w') \neq \delta_{x,b}(w')] \\
&\geq 1 - 2^{\ell(n)} \Pr[\mathcal{O}(\delta_{x,b}; r)(w') \neq \delta_{x,b}(w')] \\
&\geq 1 - 2^{\ell(n)} \tfrac{2}{2^{2\ell(n)}} \\
&= 1 - \mathsf{negl}(n).
\end{aligned}$$

[18] We assume that the (un-obfuscated) circuit $\delta_{x,b}$ has a witness w of x (if such exists) hard-wired into it in a "visible way." If $x \notin L$, then set $b' = 0$.

- Polynomial-blowup condition: Trivial.

- Virtual black-box condition: Fix any PPT machine $\mathcal{A}$ and any polynomials $p(\cdot)$ and $q(\cdot)$. We need to show that there exists a PPT oracle machine $\mathcal{S}$ such that for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and every auxiliary input z of size $q(n)$ (z may depend on $\delta_{x,b}$),

$$\left|\Pr[\mathcal{A}(\mathcal{O}(\delta_{x,b}), z) = 1] - \Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1]\right| < \frac{1}{p(n)}$$

(where the probabilities are over the random coin tosses of $\mathcal{A}$, $\mathcal{S}$ and the obfuscator $\mathcal{O}$).

As a first step, notice that it is easy to emulate $\mathcal{O}(\delta_{x,b}; r)$ given the values x, r and $\mathcal{A}_1(x, w, r) \oplus b$. Thus, there exists a PPT machine $\mathcal{T}$, such that for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and every auxiliary input z of size $q(n)$ (z may depend on $\delta_{x,b}$),

$$\Pr[\mathcal{A}(\mathcal{O}(\delta_{x,b}), z) = 1] = \Pr[\mathcal{T}(x, r, \mathcal{A}_1(x, w, r) \oplus b, z) = 1]$$

(where the probabilities are over the random coin tosses of $\mathcal{A}$, $\mathcal{A}_1$, $\mathcal{T}$, $\mathcal{O}$, and over $r \in_R \{0,1\}^n$).

Therefore, it suffice to show that there exists a PPT oracle machine $\mathcal{S}$ such that for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and every auxiliary input z of size $q(n)$ (z may depend on $\delta_{x,b}$),

$$\left|\Pr[\mathcal{T}(x, r, \mathcal{A}_1(x, w, r) \oplus b, z) = 1] - \Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1]\right| < \frac{1}{p(n)} \qquad (4.1)$$

(where the probabilities are over the random coin tosses of $\mathcal{A}_1$, $\mathcal{T}$, $\mathcal{S}$, and over $r \in_R \{0,1\}^n$).

For every $n \in \mathbb{N}$, let

$$S_0^n = \left\{(x, b, z) : \Pr_r[\mathcal{T}(x, r, B(x, r) \oplus b, z) = 1] - \Pr_{r,b'}[\mathcal{T}(x, r, b', z) = 1] > \frac{1}{p(n)}\right\},$$

$$S_1^n = \left\{ (x,b,z) : \Pr_{r,b'}[\mathcal{T}(x,r,b',z)=1] - \Pr_{r}[\mathcal{T}(x,r,B(x,r)\oplus b,z)=1] > \frac{1}{p(n)} \right\},$$

where $x \in \{0,1\}^n, b \in \{0,1\}, z \in \{0,1\}^{q(n)}$.

By applying standard amplification techniques, we can assume without loss of generality that the PPT oracle machine $\mathcal{A}_2$ (given to us from the definition of a hard-core predicate for L) satisfies that for every $x \in L$ and for every function f_x, that on input a random r outputs $B(x,r)$ with probability at least $\frac{1}{2} + \frac{1}{p(|x|)}$, it holds that

$$\Pr[\mathcal{A}_2^{f_x}(x) = w \text{ s.t. } (x,w) \in \mathcal{R}_L] = 1 - \mathsf{negl}(n),$$

where the probability is over the random coin tosses of $\mathcal{A}_2$.

We use this PPT oracle machine $\mathcal{A}_2$, together with the PPT machine $\mathcal{T}$, to define two PPT machines $\mathcal{U}_0$ and $\mathcal{U}_1$ such that for every $n \in \mathbb{N}$ and for every $(x,b,z) \in S_0^n$,

$$\Pr[\mathcal{U}_0(x,b,z) = w \text{ s.t. } (x,w) \in \mathcal{R}_L] = 1 - \mathsf{negl}(n),$$

and similarly for every $n \in \mathbb{N}$ and for every $(x,b,z) \in S_1^n$,

$$\Pr[\mathcal{U}_1(x,b,z) = w \text{ s.t. } (x,w) \in \mathcal{R}_L] = 1 - \mathsf{negl}(n).$$

Using these two PPT machines $\mathcal{U}_0$ and $\mathcal{U}_1$, we construct a simulator S, that satisfies Eq. (4.1). $S^{\delta_{x,b}}(1^n, z)$, operates as follows:

1. Choose a random $r \in \{0,1\}^n$, where $|x| = n$.

2. Run $\mathcal{U}_0(x,0,z)$ and $\mathcal{U}_0(x,1,z)$. If in one of these executions the output is w such that $(x,w) \in \mathcal{R}_L$ then retrieve the secret bit b by feeding the oracle to $\delta_{x,b}$ the witness w, and output $\mathcal{T}(x,r,\mathcal{A}_1(x,w,r) \oplus b, z)$.

3. Otherwise, run $\mathcal{U}_1(x, 0, z)$ and $\mathcal{U}_1(x, 1, z)$. If in one of these executions the output is w such that $(x, w) \in \mathcal{R}_L$ then retrieve the secret bit b by feeding the oracle to $\delta_{x,b}$ the witness w, and output $\mathcal{T}(x, r, \mathcal{A}_1(x, w, r) \oplus b, z)$.

4. Otherwise, choose a random bit $b' \in_R \{0,1\}$, and output $\mathcal{T}(x, r, b', z)$.

Notice that for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and $z \in \{0,1\}^{q(n)}$, such that $(x, b, z) \in S_0^n$,

$$\Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1] = \Pr[\mathcal{T}(x, r, B(x, r) \oplus b, z) = 1](1 - \mathsf{negl}(n)).$$

Similarly, for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and $z \in \{0,1\}^{q(n)}$, such that $(x, b, z) \in S_1^n$,

$$\Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1] = \Pr[\mathcal{T}(x, r, B(x, r) \oplus b, z) = 1](1 - \mathsf{negl}(n)).$$

Finally, for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and $z \in \{0,1\}^{q(n)}$, such that $(x, b, z) \notin S_0^n \cup S_1^n$,

$$\left|\Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1] - \Pr[\mathcal{T}(x, r, B(x, r) \oplus b, z) = 1]\right| \leq \frac{1}{p(n)}$$

(follows from the definition of S_0^n and S_1^n).

Thus, all in all, for every large enough $n \in \mathbb{N}$, $x \in \{0,1\}^n$, $b \in \{0,1\}$, and $z \in \{0,1\}^{q(n)}$,

$$\left|\Pr[\mathcal{S}^{\delta_{x,b}}(1^n, z) = 1] - \Pr[\mathcal{T}(x, r, B(x, r) \oplus b, z) = 1]\right| \leq \frac{1}{p(n)},$$

as desired.

Therefore in order to conclude the proof of Theorem 34, it suffices to construct the PPT oracle machines $\mathcal{U}_0$ and $\mathcal{U}_1$. In what remains we construct $\mathcal{U}_0$ (and $\mathcal{U}_1$ is constructed analogously).

$\mathcal{U}_0$, on input (x, b, z), outputs $\mathcal{A}_2^{f_x}(x)$, where f_x on input $r \in \{0,1\}^n$ (where

$n = |x|$) is defined as follows:

1. If $\mathcal{T}(x, r, b, z) = 1$ and $\mathcal{T}(x, r, b \oplus 1, z) = 0$, then predict $B(x, r) = 0$.

2. If $\mathcal{T}(x, r, b \oplus 1, z) = 1$ and $\mathcal{T}(x, r, b, z) = 0$, then predict $B(x, r) = 1$.

3. Otherwise, choose at random $b \in_R \{0, 1\}$, and predict $B(x, r) = b$.

For every $n \in \mathbb{N}$ and for every (x, b, z), we define the following three sets:

$$\mathsf{GOOD} = \{r \in \{0,1\}^n : \mathcal{T}(x, r, B(x, r) \oplus b, z) = 1 \ \wedge \ \mathcal{T}(x, r, B(x, r) \oplus b \oplus 1, z) = 0\}$$

$$\mathsf{BAD} = \{r \in \{0,1\}^n : \mathcal{T}(x, r, B(x, r) \oplus b, z) = 0 \ \wedge \ \mathcal{T}(x, r, B(x, r) \oplus b \oplus 1, z) = 1\}$$

$$\mathsf{USELESS} = \{r \in \{0,1\}^n : \mathcal{T}(x, r, B(x, r) \oplus b, z) = \mathcal{T}(x, r, B(x, r) \oplus b \oplus 1, z)\}.$$

Notice that for every $n \in \mathbb{N}$ and for every $(x, b, z) \in S_0^n$:

– $\forall r \in \mathsf{GOOD}$,

$$\Pr[f_x(r) = B(x, r)] = 1.$$

– $\forall r \in \mathsf{BAD}$,

$$\Pr[f_x(r) = B(x, r)] = 0.$$

– $\forall r \in \mathsf{USELESS}$,

$$\Pr[f_x(r) = B(x, r)] = \frac{1}{2}.$$

Therefore,

$$\Pr[f_x(r) = B(x,r)] =$$
$$\Pr\left[r \in \mathsf{GOOD}\right] + \tfrac{1}{2}\Pr\left[r \in \mathsf{USELESS}\right] =$$
$$\Pr[r \in \mathsf{GOOD}] + \tfrac{1}{2}\left(1 - \Pr[r \in \mathsf{GOOD}] - \Pr[r \in \mathsf{BAD}]\right) =$$
$$\tfrac{1}{2} + \tfrac{1}{2}(\Pr[r \in \mathsf{GOOD}] - \Pr[r \in \mathsf{BAD}]).$$

Thus, in order to conclude the proof of Theorem 34, it remains to notice that for every $n \in \mathbb{N}$, the definition of S_0^n implies that for every $(x,b,z) \in S_0^n$,

$$\Pr_r[r \in \mathsf{GOOD}] - \Pr_r[r \in \mathsf{BAD}] \geq \frac{2}{p(n)}.$$

■

Appendix A

Appendix

A.1 Proof of Lemma 3.7.4

Proof: Assume for contradiction that ID^2 is not secure. That is, assume that there exists a cheating sender $\tilde{\mathsf{S}} = \{\tilde{\mathsf{S}}_n\}$ and a polynomial $p(\cdot)$ such that for infinitely many n's,

$$\Pr[(\tilde{\mathsf{S}}_n, \mathsf{R}^2)(\mathsf{PK}') = 1] \geq \frac{1}{p(n)}$$

(where the probability is over $\mathsf{PK}' \leftarrow \mathsf{G}^2(1^n)$ and over the random coin tosses of R^2).

Proof Plan: We will prove that the existence of $\tilde{\mathsf{S}}$ implies the existence of a circuit that finds collisions in $\mathcal{F}$. This will be done in two parts, as follows.

- **(Part 1):** We first show that there exist non-uniform probabilistic polynomial-time Turing machines $\tilde{\mathsf{F}} = \{\tilde{\mathsf{F}}_n\}$ and $\tilde{\mathsf{P}} = \{\tilde{\mathsf{P}}_n\}$, such for infinitely many n's the following holds.

  For $(a, b_1, b_2, aux_1, aux_2) = \tilde{\mathsf{F}}_n(f)$,

  $$\Pr\left[(\tilde{\mathsf{P}}_n(aux_1), \mathsf{V}^0|_f)(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}_n(aux_2), \mathsf{V}^0|_f)(f, a, b_2) = 1\right] \geq 1/p(n)^2$$

  (where the probability is over a uniformly chosen $f \in \mathcal{F}_n$, and over the random

coin tosses of $\tilde{\mathsf{F}}_n$, $\tilde{\mathsf{P}}_n$, and over the random coin tosses of both independent instances of $\mathsf{V}^0|_f$).

The proof-of-knowledge property of $(\mathsf{P}^0, \mathsf{V}^0)$ implies that there exists a probabilistic polynomial-time oracle machine E and a polynomial $p'(\cdot)$ such that for any $(a, b_1, b_2, aux_1, aux_2)$ which satisfy the above inequality,

$$\Pr\left[\begin{array}{c} \forall i\ E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),i) = w_i^1 \text{ s.t. } ((f,a,b_1),w^1) \in \mathcal{R}_{\mathcal{F}} \\ \textit{and} \\ \forall i\ E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),i) = w_i^2 \text{ s.t. } ((f,a,b_2),w^2) \in \mathcal{R}_{\mathcal{F}} \end{array}\right] \geq \frac{1}{p'(n)}$$

(where the probability is over the random coin tosses of $E^{\tilde{\mathsf{P}}_n(aux_1)}$ and $E^{\tilde{\mathsf{P}}_n(aux_2)}$).

- **(Part 2):** We will then show that there exists a probabilistic polynomial-time oracle machine, with oracle access to E, $\tilde{\mathsf{F}}_n$ and $\tilde{\mathsf{P}}_n$, such that, on input a uniformly chosen $f \in_R \mathcal{F}_n$, outputs a collision in f, with non-negligible probability.

Note that since non-uniform probabilistic polynomial-time Turing machines can be modeled as polynomial-size circuits, Part 1 together with Part 2 imply the existence of a polynomial-size circuit such that, on input a uniformly chosen $f \in_R \mathcal{F}_n$, outputs a collision in f, with non-negligible probability. This contradicts the assumption that $\mathcal{F}$ is collision resistant.

We proceed to carry out the proof plan.

Part 1:

- $\tilde{\mathsf{F}}_n(f)$ operates as follows.

  1. Choose uniformly

     – $\mathsf{PK} \leftarrow \mathsf{G}(1^n)$

     – $r \in \{0,1\}^{4n}$

– γ_1' (randomness for $\mathsf{V}_{\mathsf{PCP}}$)

and set $\mathsf{PK}' = (\mathsf{PK}, f, r, \gamma_1')$.

2. Emulate an interaction of $(\tilde{\mathsf{S}}_n, \mathsf{R}^2)(\mathsf{PK}')$ to obtain a transcript

$$(\beta_2, r \oplus (a, b_1, b_2, \beta_1); *, *; *, *) \leftarrow (\tilde{\mathsf{S}}_n, \mathsf{R}^2)(\mathsf{PK}').$$

3. Set $aux_1 = (\beta_1, \mathsf{PK}')$ and $aux_2 = (\beta_2, \mathsf{PK}')$.

Output $(a, b_1, b_2, aux_1, aux_2)$.

- $\tilde{\mathsf{P}}_n(aux_1)$ interacts with $\mathsf{V}^0|_f(f, a, b_1)$ as follows.

  – V^0 sends f to $\tilde{\mathsf{P}}_n$.

  – $\tilde{\mathsf{P}}_n$ sends β_1 to V^0.

  – V^0 chooses γ_1 at random, and sends γ_1 to $\tilde{\mathsf{P}}_n$.

  – $\tilde{\mathsf{P}}_n$ chooses γ_2^d at random (d stands for dummy, as in some sense γ_2^d is a dummy message) and emulates the interaction of $(\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1 \oplus \gamma_1', \gamma_2^d})(\mathsf{PK}')$, to obtain a transcript

$$(\beta_2, r \oplus (a, b_1, b_2, \beta_1); \gamma_1 \oplus \gamma_1', \gamma_2^d; \delta_1, \delta_2^d) \leftarrow (\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1 \oplus \gamma_1', \gamma_2^d})(\mathsf{PK}').$$

  $\tilde{\mathsf{P}}_n$ sends δ_1 to V^0.

- $\tilde{\mathsf{P}}_n(aux_2)$ interacts with $\mathsf{V}^0|_f(f, a, b_2)$ as follows.

  – V^0 sends f to $\tilde{\mathsf{P}}_n$.

  – $\tilde{\mathsf{P}}_n$ sends β_2 to V^0.

  – V^0 chooses γ_2 at random and sends γ_2 to $\tilde{\mathsf{P}}_n$.

– $\tilde{\mathsf{P}}_n$ chooses γ_1^d at random and emulates the interaction of $(\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1^d,\gamma_2})(\mathsf{PK}')$ to obtain a transcript

$$(\beta_2, r \oplus (a, b_1, b_2, \beta_1); \gamma_1^d, \gamma_2; \delta_1^d, \delta_2) \leftarrow (\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1^d,\gamma_2})(\mathsf{PK}').$$

$\tilde{\mathsf{P}}_n$ sends δ_2 to V^0.

Claim 35 *Let $\tilde{\mathsf{F}}_n(f) = (a, b_1, b_2, aux_1, aux_2)$. Then, for infinitely many n's*

$$\Pr\left[(\tilde{\mathsf{P}}_n(aux_1), \mathsf{V}^0|_f)(f, a, b_1) = 1 \ \wedge (\tilde{\mathsf{P}}_n(aux_2), \mathsf{V}^0|_f)(f, a, b_2) = 1\right] \geq 1/p(n)^2$$

(where the probability is over $f \in_R \mathcal{F}_n$, and over the random coin tosses of both copies of $\mathsf{V}^0|_f$).

Proof: By the assumption made for contradiction, for infinitely many n's

$$\Pr[(\tilde{\mathsf{S}}_n, \mathsf{R}^2)(\mathsf{PK}') = 1] \geq 1/p(n)$$

(where the probability is over PK' and over the random coin tosses of R^2).
The fact that $(\gamma_1 \oplus \gamma_1', \gamma_2^d)$ and (γ_1^d, γ_2) are both independently uniformly distributed and independent of PK', implies that for infinitely many n's, the following two conditions hold with probability at least $1/p(n)^2$.

- $(\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1 \oplus \gamma_1', \gamma_2^d})(\mathsf{PK}') = 1$
- $(\tilde{\mathsf{S}}_n, \mathsf{R}^2|_{\gamma_1^d,\gamma_2})(\mathsf{PK}') = 1$

In other words,

- $(\beta_2, r \oplus (a, b_1, b_2, \beta_1); \gamma_1 \oplus \gamma_1', \gamma_2^d; \delta_1, \delta_2^d) \in \mathsf{ACC}(\mathsf{R}^2|_{\gamma_1 \oplus \gamma_1', \gamma_2^d})(\mathsf{PK}')$
- $(\beta_2, r \oplus (a, b_1, b_2, \beta_1); \gamma_1^d, \gamma_2; \delta_1^d, \delta_2) \in \mathsf{ACC}(\mathsf{R}^2|_{\gamma_1^2,\gamma_2^2})(\mathsf{PK}')$.

Equivalently, all the following conditions hold.

- 1. $(f; \beta_1; \gamma_1 \oplus \gamma_1' \oplus \gamma_1'; \delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$

2. $(f;\beta_2;\gamma_2^d;\delta_2^d) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b_2))$.

- 1. $(f;\beta_1;\gamma_1^d \oplus \gamma_1';\delta_1^d) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b_1))$

  2. $(f;\beta_2;\gamma_2;\delta_2) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b_2))$.

In particular,

1. $(f;\beta_1;\gamma_1;\delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b_1))$

2. $(f;\beta_2;\gamma_2;\delta_2) \in \mathsf{ACC}(\mathsf{V}^0(f,a,b_2))$.

■

The proof-of-knowledge property of $(\mathsf{P}^0,\mathsf{V}^0)$ implies that there exists a probabilistic polynomial-time oracle machine E and a polynomial $p'(\cdot)$ such that for infinitely many n's, for $(a,b_1,b_2,aux_1,aux_2) = \tilde{\mathsf{F}}_n(f)$,

$$\Pr\left[\begin{array}{c} \forall i\ E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),i) = w_i^1 \text{ s.t. } ((f,a,b_1),w^1) \in \mathcal{R}_{\mathcal{F}} \\ \textit{and} \\ \forall i\ E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),i) = w_i^2 \text{ s.t. } ((f,a,b_2),w^2) \in \mathcal{R}_{\mathcal{F}} \end{array}\right] \geq \frac{1}{p'(n)}$$

(where the probability is over uniformly chosen $f, \in \mathcal{F}_n$ and over the random coin tosses of $\tilde{\mathsf{F}}_n$, $E^{\tilde{\mathsf{P}}_n(aux_1)}$ and $E^{\tilde{\mathsf{P}}_n(aux_2)}$).

Part 2: We next show how one can use E and $\tilde{\mathsf{F}}_n$ and $\tilde{\mathsf{P}}_n$ to find a collision in $\mathcal{F}$. We define a probabilistic polynomial-time oracle machine $\mathcal{M}$, which is given oracle access to E, $\tilde{\mathsf{F}}_n$ and $\tilde{\mathsf{P}}_n$, and such that on input a random function $f \in \mathcal{F}_n$ outputs a collision in f, with non-negligible probability.

$\mathcal{M}^{E,\tilde{\mathsf{F}}_n,\tilde{\mathsf{P}}_n}$, on input $f \in \mathcal{F}_n$, operates as follows.

1. Compute $(a,b_1,b_2,aux_1,aux_2) = \tilde{\mathsf{F}}_n(f)$.

2. Choose a random i, and compute $\hat{C}_i^1$ and $\hat{C}_i^2$ by emulating $E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),1+(i-1)((\lg n)^2+1))$ and $E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),1+(i-1)((\lg n)^2+1))$.[1]

[1]Recall that we assumed that $\hat{C}_i$ is the k'th bit of the witness, where $k = 1+(i-1)((\lg n)^2+1)$.

3. Compute the authentication path of $\hat{C}_i^1$ with respect to f, by emulating $E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),1+j+(i-1)((\lg n)^2+1))$ for $j=1,...,(\lg n)^2$.

4. Compute the authentication path of $\hat{C}_i^2$ with respect to f, by emulating $E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),1+j+(i-1)((\lg n)^2+1))$ for $j=1,...,(\lg n)^2$.

Claim 36 *With non-negligible probability (over $f \in_R \mathcal{F}_n$ and over the random coin tosses of $\mathcal{M}$, E, $\tilde{\mathsf{F}}_n$, and $\tilde{\mathsf{P}}_n$) somewhere along these paths there will be a collision in f.*

Proof: With non-negligible probability (over the random coin tosses of $\mathcal{M}$, E, $\tilde{\mathsf{F}}_n$, and $\tilde{\mathsf{P}}_n$) the following three conditions hold:

1. $\hat{C}_i^1 = E^{\tilde{\mathsf{P}}_n(aux_1)}((f,a,b_1),1+(i-1)((\lg n)^2+1))$, where $auth_f(\hat{C}^1)$ is a witness of (f,a,b_1) in $\mathcal{R}_{\mathcal{F}}$.

2. $\hat{C}_i^2 = E^{\tilde{\mathsf{P}}_n(aux_2)}((f,a,b_2),1+(i-1)((\lg n)^2+1))$, where $auth_f(\hat{C}^2)$ is a witness of (f,a,b_2) in $\mathcal{R}_{\mathcal{F}}$.

3. In steps 3 and 4 above E gives the authentication paths of $\hat{C}_i^1$ and $\hat{C}_i^2$.

Since $C_1 \neq C_2$ and since the circuit-encoding $C \to \hat{C}$ has large minimum distance, it follows that with probability $\frac{1}{\text{poly}(n)}$, $\hat{C}_i^1 \neq \hat{C}_i^2$ (where poly(n) is a polynomial and the probability is over a randomly chosen i).
This implies that somewhere along these paths there will be a collision in f, since $\hat{C}_i^1 \neq \hat{C}_i^2$ and yet $\mathsf{TC}_f(\hat{C}^1) = \mathsf{TC}_f(\hat{C}^2) = a$. Thus, we obtain a contradiction to our assumption that $\mathcal{F}$ is a collision resistant function ensemble. ■

■

A.2 Proof of Claim 18

Proof: Denote the output of $\mathsf{FORG}^{\mathsf{M}}(\mathsf{VK})$ by $(\beta_2, r \oplus (a,b_1,b_2,\beta_1,\beta_2); \gamma_1,\gamma_2; \delta_1,\delta_2)$. By the definition of $\tilde{\mathsf{P}}_2^n$, there exists a polynomial $p(\cdot)$ such that for every $n \in S^2_{\mathcal{H}^{\mathsf{M}}}$,

for $(a, b_1) = \tilde{\mathsf{F}}_2^n(f, h^{\mathsf{M}})$, and for $q = (f, h^{\mathsf{M}}, r)$

$$\Pr[(\tilde{\mathsf{P}}_2^n, \mathsf{V}^0|_f)(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}_2^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_q)(f, a, b_2) = 1] \geq \frac{1}{p(n)} \tag{A.1}$$

(where the probability is over $f \in_R \mathcal{F}_n$, over q, over $b_2 \in_R \{0,1\}^n$ and over the random coin tosses of V^0).

We claim that similarly, for every $n \in S^2_{\mathcal{H}^{\mathsf{M}}}$, for $(a, b_1) = \tilde{\mathsf{F}}_2^n(f, h^{\mathsf{M}})$, and for $q' = (f, h^{\mathsf{M}}, r')$, where $r' = r \oplus (a, b_1, b_2, \beta_1)$,

$$\Pr[(\tilde{\mathsf{P}}_2^n, \mathsf{V}^0|_{f,\gamma_1\oplus\gamma_1'})(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}_2^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q'})(f, a, b_2) = 1] \geq \frac{1}{p(n)} \tag{A.2}$$

(where γ_1' is part of VK, and where the probability is over $f \in_R \mathcal{F}_n$, over q', over $b_2 \in_R \{0,1\}^n$, and over $\gamma_1 \oplus \gamma_1'$).

This is so for the following two reasons

1. $\gamma_1 \oplus \gamma_1'$ was chosen uniformly (follows from the fact that γ_1' was chosen uniformly and γ_1 was chosen independently of γ_1').
2. $\tilde{\mathsf{P}}_2^n$ (in step 5) cannot distinguish between the distribution of q' and the distribution of a random query q of $\mathsf{V}^{\mathcal{H}^{\mathsf{M}}}$.

For these two reasons, $\tilde{\mathsf{P}}_2^n$ in (2) should succeed with essentially the same probability as in (1).

The fact that $(\tilde{\mathsf{P}}_2^n, \mathsf{V}^0|_{f,\gamma_1\oplus\gamma_1'})(f, a, b_1) = 1$ implies that

- $(f; \beta_1; \gamma_1 \oplus \gamma_1'; \delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$.

The fact that $(\tilde{\mathsf{P}}_2^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q'})(f, a, b_2) = 1$ implies that $(q'; ans) \in \mathsf{ACC}(\mathsf{V}^{\mathcal{H}^{\mathsf{M}}}(f, a, b_2))$, which in turn implies that both of the following conditions hold.

- $(f; \beta_2; \gamma_2; \delta_2) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_2))$
- $(\gamma_1, \gamma_2) = h(\beta_2, r', \mathsf{M})$.

The satisfaction of the above three conditions imply that the forgery was successful.

■

A.3 Proof of Claim 19

Proof: Denote the output of the forger $\mathsf{FORG}^{\mathsf{M}}(\mathsf{VK})$ by $(\beta_1, r_1 \oplus (a, b_1); \gamma_1, b_2, r_2; \delta_1, ans_2)$. The existence of an impersonator for $\mathcal{H}^{\mathsf{M}}$ and $\mathcal{G}$, implies that there exist two poly-size circuit families $\tilde{\mathsf{F}} = \{\tilde{\mathsf{F}}_1^n\}$ and $\tilde{\mathsf{P}} = \{\tilde{\mathsf{P}}_1^n\}$, and a polynomial $p'(\cdot)$, such that for every $n \in S^1_{\mathcal{H}^{\mathsf{M}},\mathcal{G}}$ and for $a = \tilde{\mathsf{F}}_1^n(f, h^{\mathsf{M}}, g)$,

$$\Pr[(\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{(f,h^{\mathsf{M}},r_1)})(f, a, b_1) = 1 \ \wedge \ (\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{G}}|_{(f,g,r_2)})(f, a, b_2) = 1] \geq \frac{1}{p'(n)} \qquad \text{(A.3)}$$

(where the probability is over $f \in_R \mathcal{F}_n$, $h^{\mathsf{M}} \in_R \mathcal{H}^{\mathsf{M}}$, $g \in_R \mathcal{G}$, $r_1, r_2 \in_R \{0,1\}^{4n}$, and $b_1, b_2 \in_R \{0,1\}^n$).

We claim that similarly, for $a = \tilde{\mathsf{F}}_1^n(f, h^{\mathsf{M}}, g)$,

$$\Pr[(\tilde{\mathsf{P}}_1^n, \mathsf{V}^{\mathcal{H}^{\mathsf{M}}}|_{q_1=(f,h^{\mathsf{M}},r_1')})(f, a, b_1) = 1 \wedge (\tilde{P}_1^n, \mathsf{V}^{\mathcal{G}}|_{q_2=(f,g,r_2\oplus r_2')})(f, a, b_2 \oplus b_2') = 1] \geq \frac{1}{p'(n)} \qquad \text{(A.4)}$$

(where $r_1' = r_1 \oplus (a, b_1)$, (b_2, r_2) satisfies $(*, b_2, r_2) = h(\beta_1, r_1', \mathsf{M})$, and the probability is over $f, h, r_1, r_2', b_2', b_1$).

This is so for the following reasons

1. $b_2 \oplus b_2'$ is uniformly distributed in $\{0,1\}^n$.
2. $r_2 \oplus r_2'$ is uniformly distributed in $\{0,1\}^{4n}$.
3. r_1' is uniformly distributed in $\{0,1\}^{4n}$.

For these three reasons, $\tilde{\mathsf{P}}_n^1$ in (4) should succeed with essentially the same probability as in (3).

Thus, for every $n \in S^1_{\mathcal{H}^{\mathsf{M}},\mathcal{G}}$, the following conditions hold with probability $\geq \frac{1}{p'(n)}$.

1. $(q_2; ans_2) \in \mathsf{ACC}(\mathsf{V}^{\mathcal{G}}(f, a, b_2 \oplus b_2'))$.

2. $(f, h^{\mathsf{M}}, r_1'; ans_1) \in \mathsf{ACC}((\mathsf{V}^{\mathcal{H}^{\mathsf{M}}}(f, a, b_1))$, which in turn implies that the following conditions hold.

   (a) $\gamma_1 = h^{\mathsf{M}}(\beta_1, r_1')$, which implies that $(\gamma_1, (b_2, r_2)) = h(\beta_1, r_1', \mathsf{M})$

   (b) $(f; \beta_1; \gamma_1; \delta_1) \in \mathsf{ACC}(\mathsf{V}^0(f, a, b_1))$

Recall that $\mathsf{VERIFY}^3_{\mathcal{H}}(\mathsf{VK})$ accepts if conditions (1) and (2) hold, and thus for every $n \in S^1_{\mathcal{H}^{\mathsf{M}},\mathcal{G}}$, $\mathsf{FORG}^{\mathsf{M}}(\mathsf{VK})$ is successful with probability $\geq \frac{1}{p'(n)}$. ■

Appendix B

Figures

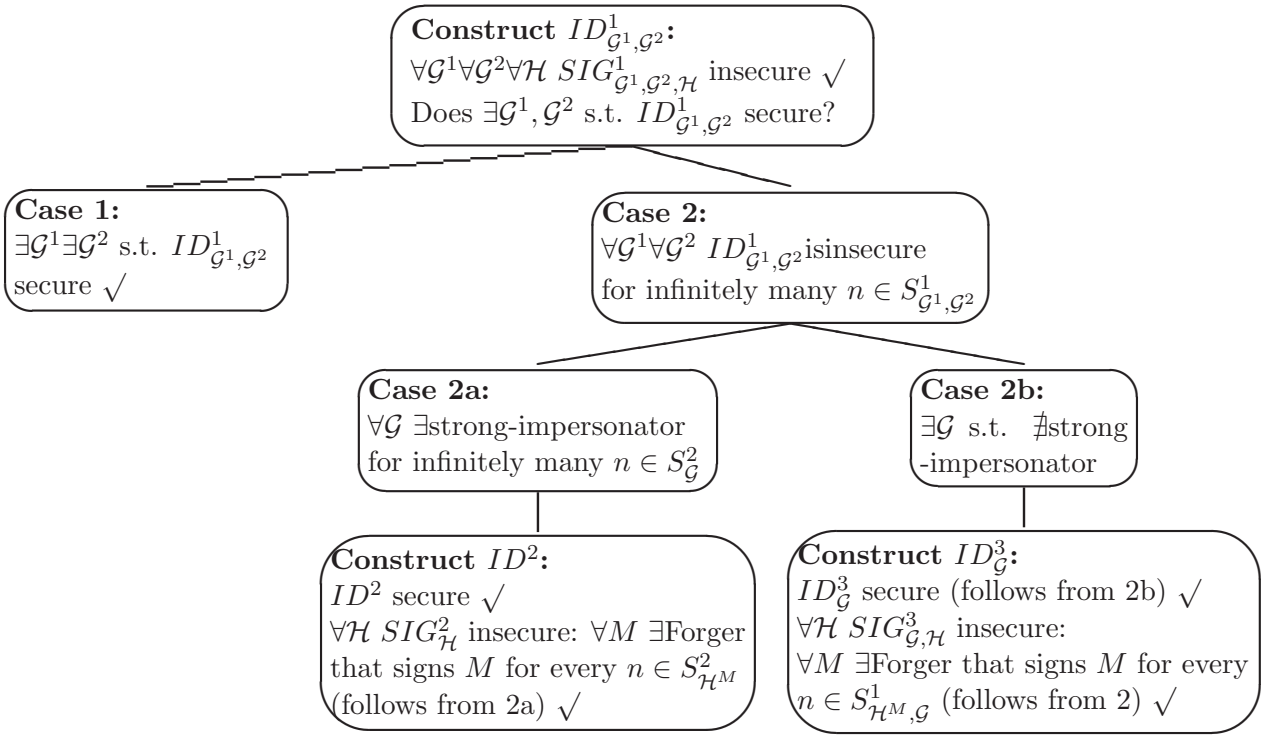


Figure B-1: Proof of Theorem 2

Bibliography

[1] M. Abdalla, J. An, M. Bellare and C. Namprempre. From identification to signatures via the Fiat-Shamir transform: minimizing assumptions for security and forward-security. In *Advances in Cryptography-EUROCRYPT 02, Lecture Notes in Computer Science, Springer-Verlag*, pages 418-433, 2002.

[2] B. Barak. How to Go Beyond the Black-Box Simulation Barrier. In 42*nd FOCS*, pages 106-115, 2001.

[3] B. Barak and O. Goldreich. Universal arguments and their applications. In *Proceedings of the 17th IEEE Annual Conference on Computational Complexity*, pages 194-203, 2002.

[4] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, K. Yang On the (Im)possibility of Obfuscating Programs. In *CRYPTO 2001*, pages 1-18.

[5] R. Canetti. Towards Realizing Random Oracles: Hash Functions That Hide All Partial Information. In *CRYPTO 1997*, pages 455-469.

[6] R. Canetti, D. Micciancio, and O. Reingold. Perfectly One-Way Probabilistic Hash Functions. In *Proceeding of STOC 1998*, pages 131-140.

[7] R. Cramer and V. Shoup. Signature schemes based on the strong RSA assumption. In *5th ACM Conference on Computer and Communications Security*, pages 46-51. Singapore, Nov. 1999. ACM Press.

[8] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22 (Nov.), pages 644-654, 1976.

[9] C. Dwork, M. Naor, O. Reingold and L.Stockmeyer. Magic funcitons. In *IEEE, editor, 40th Annual Symposium of Foundations of Computer Science*: October 17-19, 1999, New York City, New York, pages 523-534. *IEEE Computer Society Press*, 1999.

[10] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In *STOC 2005*, pages 654-663.

[11] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2), pages 77-94, 1988.

[12] Amos Fiat and Adi Shamir. How to prove to yourself: practical solutions to identification and signature problems. In *Advances in Cryptology–Crypto 86,* pages 186-194, Springer, Berlin, 1987.

[13] O. Goldreich. *Foundations of Cryptography: Volume 1 – Basic Tools.* Cambridge University Press, 2001.

[14] O. Goldreich. *Foundations of Cryptography: Volume 2 – Basic Applications.* Cambridge University Press, 2004.

[15] G. Goldreich, S. Goldwasser, and S. Micali. How to Construct Random Functions. In *Journal of the ACM,* 33(4), 1986, 792-804.

[16] R. Gennaro, S. Halevi and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *proceedings of Eurocrypt '99, LNCS vol. 1592, Springer-Verlag, 1999,* pages 123-139.

[17] O. Goldreich and L.A. Levin. A Hard-Core Predicate for all One-Way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, pages 25-32, 1989.

[18] S. Goldwasser and S. Micali. Probabilistic encryption. In *Journal of Computer and System Sciences,* 28:270-299, 1984.

[19] S. Goldwasser, S. Micali, R. L. Rivest A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. In *SIAM Journal of Computation,* 17(2): 281-308 (1988).

[20] O. Goldreich and Y. Oren. Definitions and Properties of Zero-Knowledge Proof Systems. In *Journal of Cryptology,* Vol. 7, No. 1, pages 1-32, 1994.

[21] L. Guillou and J. J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. *Advances in Cryptology-CRYPTO 88*, Lecture Notes in Computer Science Vol. 403, S. Goldwasser ed., Springer-Verlag, 1988.

[22] S. Hada. Zero-Knowledge and Code Obfuscation. In *Advances in Cryptology – ASIACRYPT '00*, pages 443-457.

[23] J. Hastad, R. Impagliazzo, L.A. Levin, M. Luby. A Pseudorandom Generator from any One-Way Function. In *SIAM Journal on Computing* 28 (1999), pages 1364-1396 (electronic).

[24] J. Kilian. A note on efficient zero-knowledge proofs and arguments. In *24'th STOC*, pages 723-732, 1992.

[25] B.Lynn, M. Prabhakaran, and A. Sahai. Positive Results and Techniques for Obfuscation. In *Proceedings of Eurocrypt*, 2004, pages 20-39.

[26] R.C. Merkle. A certified digital signature. *Proceedings on Advances in Cryptology*, pages 218-238, July 1989, Santa- Barbara, California.

[27] S. Micali. Computationally sound proofs. *SICOMP*, vol. 30(4), pages 1253-1298, 2000. Preliminary version in *35th FOCS*, 1994.

[28] R. Rivest. The MD5 message-digest algorithm. *RFC* 1321, April 1992.

[29] S. Micali and L. Reyzin. Improving the exact security of digital signature schemes. In *Jornal of Cryptology*, 15(1):1-18, 2002.

[30] M. Naor. Bit Commitment using Pseudorandom Generators. *Journal of Cryptology*, 4(2):151-158, 1991.

[31] M. Naor and M. Yung. Universal one-way hash functions and their cryptogracphic applications. In *STOC 89*, pages 33-43.

[32] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology - CRYPTO 92*, Lecture Notes in Computer Science Vol. 740, E. Brickell ed., Springer-Verlag, 1992.

[33] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Advances in Cryptology-EUROCRYPT 96*, vol.1070 of Lecutre Notes in Compute Science, pages 387-398. Springer-Verlag, 1996.

[34] John Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. In *STOC 1990*, pages 387-394.

[35] Schnorr. Efficient signature generation by smart cards. In *Journal of Cryptology* 4(3):161-174.

[36] H. Wee. On Obfuscating Point Functions. In *eprint* 2005/001.