2012

Functional Encryption as Mediated Obfuscation

Robert Perry Hooker
The University of Montana

FUNCTIONAL ENCRYPTION AS MEDIATED OBFUSCATION

By

ROBERT PERRY HOOKER

Bachelor of Science, Western State College, Gunnison, Colorado, 2005

Thesis

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

May 2012

Approved by:

Dr. Sandy Ross, Dean of The Graduate School
Graduate School

Dr. Mike Rosulek, Chair
Department of Computer Science

Dr. Douglas Raiford
Department of Computer Science

Dr. Mark Kayll
Department of Mathematical Sciences

Acknowledgments

This thesis would have been impossible without the advice, mentoring, and constant reassurance of my advisor, Mike Rosulek. I am continually impressed with his patience, work ethic, and equanimity. Mike is a consummate educator: always kind, never frustrated, and genuinely passionate about his work.

I'm also grateful to my teachers in the UM CS department: Michael Cassens, Min Chen, Glen Granzow, Joel Henry, Jesse Johnson, Doug Raiford, Yolanda Reimer, and Alden Wright. It takes a department to raise a researcher.

To my fellow CS grad students: you've been wonderful co-workers and better friends. May you all succeed in fabulous style.

Finally - to my family, my *sine qua non*: you've *always* been there. This thesis is dedicated to you.

Abstract

We introduce a new model for program obfuscation, called **mediated obfuscation.** A mediated obfuscation is a 3-party protocol for evaluating an obfuscated program that requires minimal interaction and limited trust. The party who originally supplies the obfuscated program need not be online when the client wants to evaluate the program. A semi-trusted third-party mediator allows the client to evaluate the program, while learning nothing about the obfuscated program or the client's inputs and outputs. Mediated obfuscation would provide the ability for a software vendor to safely outsource the less savory aspects (like accounting of usage statistics, and remaining online to facilitate access) of "renting out" access to proprietary software.

We give security definitions for this new obfuscation paradigm, and then present a simple and generic construction based on functional encryption. If a functional encryption scheme supports decryption functionality $F(m, k)$, then our construction yields a mediated obfuscation of the class of functions $\{F(m, \cdot) \mid m\}$. In our construction, the interaction between the client and the mediator is minimal (much more efficient than a general-purpose multi-party computation protocol). Instantiating with existing FE constructions, we achieve obfuscation for point-functions with output (under a strong "virtual black-box" notion of security), and a general feasibility result for obfuscating conjunctive normal form and disjunctive normal form formulae (under a weaker "semantic" notion of security).

Finally, we use mediated obfuscation to illustrate a connection between worst-case and average-case *static* obfuscation. In short, an average-case (static) obfuscation of some component of a suitable functional encryption scheme yields a worst-case (static) obfuscation for a related class of functions. We use this connection to demonstrate new impossibility results for average-case (static) obfuscation.

Contents

List of Figures

Chapter 1

Introduction

Cryptography is a means for controlling access to information. Traditionally, encrypted data is both *useless* to unauthorized parties and *fully accessible* to authorized users. However, there are many natural applications for encryption schemes that relax these restrictions.

Functional encryption (FE) [BSW11a, O'N10] is a new class of encryption which refines cryptography's traditional "all-or-nothing" pattern. Broadly speaking, a functional encryption system allows different users to hold unique secret keys which decrypt *arbitrary functions* of the plaintext.

Functional encryption has many potential applications. For example, consider a large, cloud-hosted, encrypted database of student information that contains both academic and financial data. Here, the level of access required by teachers and accountants is different: teachers only need to access the grades of students in their classes, while accountants only need to access student financial reports. Traditionally, this access policy would be enforced by storing the data on a trusted server that both authenticates users and supplies them with appropriate content. However, in a cloud-based model, it may be undesirable to outsource this level of trust in addition to data storage. In this case, the standard "all-or-nothing" model is insufficient, because all key holders (*i.e.*, both teachers and accountants) can decrypt the entire student database. Functional encryption provides an alternative approach: users get keys that only allow them to decrypt appropriate content.

In this work, we apply functional encryption to the problem of **program obfuscation.** Informally, the goal of obfuscation is to make the source code of a program "unintelligible" without altering its functionality. Though theoretical results such as the undecidability of

the HALTING PROBLEM and the conjectured difficulty of BOOLEAN SATISFIABILITY suggest that computer programs are fundamentally difficult to interpret, in practice computer programs are routinely analyzed[1] by studying their source code. This poses a problem for software vendors who want to create and distribute programs without revealing the (potentially valuable) details of their implementation. Though some software vendors have addressed this problem by hosting their programs in-house on secure servers, such systems require the vendor to maintain a significant on-line presence. Maintaining such a presence may be undesirable (or infeasible) for all software producers.

Obfuscation provides one solution to this dilemma. An obfuscated program is by definition protected against reverse-engineering: the program still "works" on arbitrary inputs, but its code is undecipherable (to anyone who does not hold the secret key). This implies that an obfuscated program is a "virtual black-box:" the program is functional, but its "inner workings" are disguised from view.

As a concrete example, consider the following scenario: Alice, a software vendor, wishes to sell Bob the ability to run some proprietary software *without revealing the program's source code*. Alice could simply run the program on Bob's behalf, but Alice would prefer not to remain constantly available to service Bob's requests. Instead, she would like to sell Bob an *obfuscated* version of her software, and then go off-line. The security of obfuscation implies that Bob would learn no more from this obfuscated program than he could have learned in the trivial scenario in which Alice simply runs the program on his behalf.

Unfortunately, this type of *static* obfuscation is impossible for arbitrary programs [BGI+01]. Even for very simple programs, the known positive results for obfuscation require significant security compromises.

As an alternative to static obfuscation, we propose a new relaxed notion of obfuscation called **mediated obfuscation** (MO). To achieve fully secure obfuscation of arbitrary programs, mediated obfuscation uses a minimal amount of *interaction* with a semi-trusted third party. Thus, instead of evaluating a static program locally, the user is required to engage in a 2-party interactive protocol to evaluate an obfuscated program on a given input. Furthermore, we exclude the trivial solution in which Alice evaluates the program completely on Bob's behalf (which can hardly be considered an "obfuscation"). To accomplish this, our model introduces a third party called the **mediator**. Alice can

[1] E.g., to determine the program's purpose and/or algorithmic complexity.

obfuscate her program, send it to Bob, and then *outsource* the requirement of constant, on-demand availability to the mediator. That is, Bob interacts with the always-online mediator whenever he wants to evaluate the program on an input. All this is achieved while still preventing reverse engineering (in this case, by either Bob or the mediator).

1.1 Our Results

In this thesis, we introduce and define the mediated obfuscation model. Briefly, a mediated obfuscation is a 3-party protocol with the following syntax: a **vendor** obfuscates the program f by generating an obfuscated program for the **client** and some key information for the mediator. When the client wants to evaluate f on input x, he engages in a protocol with the mediator, from which he learns $f(x)$.

We establish two security definitions for MO: a "virtual black-box" simulation-based definition, and a second, weaker definition in the style of semantic security. We show the general feasibility of MO based on one-round secure computation and fully homomorphic encryption, and then demonstrate a simple, efficient MO scheme based on functional encryption. To extend the MO paradigm to obfuscation of arbitrary functions, we show that a natural way of composing FE schemes preserves security and results in a more expressive FE system.

Finally, we use mediated obfuscation to connect worst-case and average-case static obfuscation. In short, an average-case (static) obfuscation of some component of a suitable functional encryption scheme yields a worst-case (static) obfuscation for a related class of functions. We use this connection to demonstrate new impossibility results for average-case (static) obfuscation.

This thesis is based on joint work conducted with Manoj Prabhakaran and Mike Rosulek [HPR12].

Chapter 2

Preliminaries

In this chapter, we present preliminary definitions and notation, including brief overviews of functional encryption, obfuscation, and the Universally Composable security framework. Readers who are familiar with these concepts may wish to review the new result on the composability of FE schemes.

2.1 Basic Definitions

When X is a finite set, the notation $x \leftarrow X$ means that x is chosen uniformly at random from X. We say a function $f : \mathbb{N} \to \mathbb{R}$ is **negligible** in λ if for every constant $c > 0$ there exists an integer n such that $f(\lambda) < \lambda^{-c}$ for all $\lambda > n$. A **probabilistic polynomial time** (PPT) algorithm $\mathcal{A}$ has access to a source of randomness and runs in polynomial time; that is, there exists a polynomial $p(\cdot)$ such that for every input $x \in \{0,1\}^*$, the computation $\mathcal{A}(x)$ terminates within $p(|x|)$ steps. For algorithms $\mathcal{A}$ and F and an input string x, we write $\mathcal{A}^F(x)$ to denote the output of $\mathcal{A}$ when executed on input x with oracle access to F (*i.e.*, $\mathcal{A}$ has access to a "black box" that can evaluate F in a single computational step). Given a countable set[1] I, a **probability ensemble** indexed by I is a collection of random variables $\{D_i \mid i \in I\}$. We say two probability ensembles $D = \{D_n \mid n \in \mathbb{N}\}$ and $D' = \{D'_n \mid n \in \mathbb{N}\}$ are **computationally indistinguishable** (written $D \approx D'$) if no efficient procedure can tell them apart; *i.e.*, for all PPT algorithms $\mathcal{A}$ the quantity

$$\left|\Pr\left[x \leftarrow D_n; \mathcal{A}(x) = 1\right] - \Pr\left[x \leftarrow D'_n; \mathcal{A}(x) = 1\right]\right| \text{ is negligible in } n.$$

[1] Typically either the natural numbers $\mathbb{N}$ or a polynomial-time computable subset of $\{0,1\}^*$.

2.2 Functional Encryption

Functional encryption (FE) forms the basis of this work. FE provides sophisticated and flexible relations between messages and secret keys: a message m is encrypted to a ciphertext using the public key pk, and a secret key sk_k is associated with a parameter k. Using sk_k allows the key holder to compute a specific function of k and m from an encryption of m.

FE encompasses a wide range of encryption primitives like identity-based encryption [BF01, SW04, YFDL04, Wat05, BW06, ABV+11], attribute-based encryption [BSW07, BSW11a, Wat08, AI09, OT10, LOS+10], and predicate encryption [KSW08, BW07]. Functional encryption's expressive power immediately yields natural applications in database access control, email services, and multimedia content distribution [GPSW06, BW07, BH08].

2.2.1 Syntax

A functional encryption scheme Σ is parameterized by a **functionality** F which describes the information about the plaintext that can be learned from the ciphertext: the result of decrypting a ciphertext associated with plaintext m, using a decryption key associated with value k, is $F(m, k)$. Formally, a functionality is a function $F : \mathcal{M} \times \mathcal{K} \to \{0,1\}^*$, where $\mathcal{K}$ is the key space and $\mathcal{M}$ is the message space.[2] A functional encryption scheme with functionality F is a tuple of PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, with the following syntax:

$(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$: given a security parameter, Setup samples a public key and a master key.

$\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{mk}, k)$: accepts a master key and a parameter $k \in \mathcal{K}$, and outputs a decryption key.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$: takes the public key and a plaintext $m \in \mathcal{M}$, and returns a ciphertext.

$y \leftarrow \mathsf{Dec}(\mathsf{sk}_k, c)$: accepts a decryption key and a ciphertext, and returns either a string or an error indicator $\perp$.

[2] Both the key space and the message space implicitly depend on the security parameter λ. For simplicity of notation, we treat the security parameter as implicit in this context.

The correctness requirement is that, for all $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and all $\mathsf{sk}_k \leftarrow \mathsf{KeyGen}(\mathsf{mk}, k)$, we have $\mathsf{Dec}(\mathsf{sk}_k, \mathsf{Enc}(\mathsf{pk}, m)) = F(m, k)$ with probability 1 over the randomness of Enc.

Additionally, we extend the standard definition of FE to include a natural notion of *key verifiability* wherein a user can verify the validity of a decryption key corresponding to a given value.

Definition 2.2.1 (Key verifiability)**.** *A functional encryption scheme with key verifiability is an FE scheme as defined above with an additional* CheckKey *algorithm:*

> $b \leftarrow \mathsf{CheckKey}(\mathsf{pk}, \widetilde{\mathsf{sk}}, x)$*: given a public key, a purported decryption key, and a parameter* x*,* CheckKey *returns 1 if* $\forall m : \mathsf{Dec}(\widetilde{\mathsf{sk}}, \mathsf{Enc}(\mathsf{pk}, m)) = F(m, x)$ *with overwhelming probability over the randomness used in* Enc*, and 0 otherwise.*

Note that $\forall x : \Pr[\mathsf{CheckKey}(\mathsf{pk}, \mathsf{KeyGen}(\mathsf{mk}, x), x) = 1]$ is overwhelming in λ.

2.2.2 Identity-based encryption

Identity-based encryption (IBE) [Sha85, BF01, Wat05] is a notable special case of functional encryption with particular relevance to the problem of obfuscation. In IBE, a ciphertext is "addressed" to an arbitrary nonempty string id (such as an email or IP address), and plaintext messages are pairs (id, m). In terms of the functionality F, an IBE system uses the function:

$$F((\mathsf{id}, m), \mathsf{id}') = \begin{cases} m & \text{if } \mathsf{id} = \mathsf{id}' \\ \bot & \text{otherwise} \end{cases}$$

In other words, the encrypted message $\mathsf{Enc}(\mathsf{pk}, (\mathsf{id}, m))$ can only be decrypted with the secret key corresponding to the identity id.

The first IBE schemes explicitly included id "in the clear" as an unencrypted part of the ciphertext. This partial information is represented in terms of the functionality F via a special empty key ε. Using the empty key[3] as input to Dec yields all the information about the plaintext that can be gleaned by simply examining the ciphertext, thus, $\mathsf{Dec}(\varepsilon, c) = F(\varepsilon, (\mathsf{id}, m)) = \mathsf{id}$. This simplified the constructions, but allowed anyone to determine a message's intended recipient simply by examining the ciphertext. However,

[3] Typically defined as the empty string.

the development of *anonymous* identity-based encryption (AIBE) [BW06, Gen06] solved this problem with ciphertexts that do not reveal the message's intended receiver.

Predicate encryption (PE) [KSW08, BW07] generalizes the AIBE concept: messages consist of a "payload" and a set of "attributes," secret keys are associated with a predicate, and the payload is revealed if and only if the predicate "accepts" the set of attributes. As before, a ciphertext in a PE system reveals nothing about the payload or the set of attributes.

Here, the salient point is that an anonymous IBE scheme *effectively obfuscates the functionality* F (which tests whether $\mathsf{id} = \mathsf{id}'$). This can be seen by considering an adversary who has access to the FE algorithms for F: the security of the FE scheme guarantees that the adversary cannot learn anything meaningful by examining secret keys and ciphertexts - at best, it can only execute the Dec algorithm on inputs of its choice (assuming it obtains secret keys for those inputs).

Notably, the AIBE functionality is analogous to the class of **point functions with output** $\mathcal{I} = \{I_{x,m} \mid x, m\}$, where a point function $I_{x,m}$ is a boolean function that assumes the value m at exactly one point x:

$$I_{x,m}(y) = \begin{cases} m & \text{if } x = y \\ \\ \perp & \text{otherwise} \end{cases}$$

Secure AIBE is thus akin to an obfuscation of the class $\mathcal{I}$ of point functions with output, which are used in practice to perform tasks like *password checking*.

2.2.3 Security definitions

Security definitions for functional encryption schemes come in two basic flavors: indistinguishability-based and simulation-based. Briefly, indistinguishability-based security implies that no efficient (*i.e.*, PPT) algorithm can distinguish between ciphertexts that encode different messages, while simulation-based security is defined relative to an "ideal world" where a perfectly trusted party is used to control access to messages. A real scheme is said to be "simulation-secure" if no (real) adversary can learn more about the encrypted messages in the real world than in the ideal one (where by definition no information is leaked unintentionally). Simulation-based security is stronger but typically harder to achieve.

Here, we present two variants of the indistinguishability-based and simulation-based security definitions given by Boneh, Sahai, & Waters [BSW11a]. In both cases, we require

that the ciphertext conceal the entire message m.

Definition 2.2.2 (Selective security)**.** *Let Σ be an FE scheme as above. Define the following interaction:*

$\underline{\mathsf{SelSecExp}(\Sigma, \mathcal{A}, \beta, \lambda) :}$

1. *Give 1^λ to $\mathcal{A}$, who then outputs plaintexts $m_0, m_1 \in \mathcal{M}$.*
2. *Compute $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $c = \mathsf{Enc}(\mathsf{pk}, m_\beta)$. Give (pk, c) to $\mathcal{A}$.*
3. *$\mathcal{A}$ is given access to an oracle $\mathsf{KeyGen}(\mathsf{mk}, \cdot)$, subject to the constraint that $F(m_0, k) = F(m_1, k)$ for all k queried to the oracle.*
4. *$\mathcal{A}$ outputs a bit, which we define as the output of this interaction.*

Define the advantage of the adversary $\mathcal{A}$ as:

$$\mathsf{AdvSel}(\Sigma, \mathcal{A}, \lambda) := \left| \Pr\left[\mathsf{SelSecExp}(\Sigma, \mathcal{A}, 1, \lambda) = 1\right] - \Pr\left[\mathsf{SelSecExp}(\Sigma, \mathcal{A}, 0, \lambda) = 1\right] \right|$$

We say that Σ is Selective-**secure** *if for all PPT $\mathcal{A}$ the function $\mathsf{AdvSel}(\Sigma, \mathcal{A}, \lambda)$ is negligible in λ.*

This Selective security definition is the selective relaxation of the definition for FE given in [BSW11a]. In the case that the FE scheme is a predicate scheme, the values m_0 and m_1 in the above experiment contain both the "payload" and the "attributes" associated with the ciphertext. Thus the above experiment combines both the standard notion of security (payload hiding) as well as the notion of (strongly) attribute-hiding [KSW08].[4]

Definition 2.2.3 (Sim_1 security)**.** *Let Σ be an FE scheme as above. Define the following interactions:*

$\underline{\mathsf{Real}(\Sigma, \mathcal{A}, m, \lambda)\text{:}}$

1. $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$
3. $z \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{mk}, \cdot)}(1^\lambda, \mathsf{pk}, c)$
4. *output $(z, k_1 \ldots, k_n)$, where $k_1, \ldots, k_n$ are the queries made to the oracle in the previous step.*

$\underline{\mathsf{Ideal}(\Sigma, \mathcal{S}, m, \lambda)\text{:}}$

1. $z \leftarrow \mathcal{S}^{F(m, \cdot)}(1^\lambda)$
2. *output $(z, k_1 \ldots, k_n)$, where $k_1, \ldots, k_n$ are the queries made to the oracle in the previous step.*

We say that Σ is Sim_1-**secure** *if for all PPT $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ such that for all m, the distribution ensembles $\{\mathsf{Real}(\Sigma, \mathcal{A}, m, \lambda)\}_\lambda$ and $\{\mathsf{Ideal}(\Sigma, \mathcal{S}, m, \lambda)\}_\lambda$ are indistinguishable.*

[4] Weak attribute hiding would correspond to the additional restriction that $F(m_0, k) = F(m_1, k) = \perp$, in this case.

Furthermore, if for all $\mathcal{A}$, the corresponding $\mathcal{S}$ treats $\mathcal{A}$ only in a straight-line, black-box manner, then we say that the scheme is **straight-line-Sim_1-secure**.

Simulation-based security implies that no PPT adversary can distinguish between encryptions of two different messages (m_0, m_1) without an evaluation token for a key $k \in \mathcal{K}$ such that $F(m_0, k) \neq F(m_1, k)$.

The simulation-based security definition given in [BSW11a] allows the adversary to request many ciphertexts, and also learn the value $F(m, \varepsilon)$ — the so-called "intentional leakage" of the ciphertext. Thus, in the case where $F(m, \varepsilon) = \varepsilon$ (*i.e.*, the functionality is "attribute hiding") and the adversary only requests one ciphertext, their definition collapses to the Sim_1 definition here.

Theorem 2.2.4. *Let Σ be a Sim$_1$-secure FE scheme as above. Then, Σ is also Selective-secure.*

Proof. We prove the contrapositive. Assume that there exists an adversary $\mathcal{A}^*$ such that $\mathsf{AdvSel}(\Sigma, \mathcal{A}^*, \lambda)$ is non-negligible (*i.e.*, Σ is not Selective-secure). Then, it suffices to show that there exists an adversary $\mathcal{A}^+$ that breaks the Sim_1-security of Σ: *i.e.*, $\exists \mathcal{A}^+ \forall \mathcal{S} \exists m$: $\mathsf{Real}(\Sigma, \mathcal{A}^+, m, \lambda) \not\approx \mathsf{Ideal}(\Sigma, \mathcal{S}, m, \lambda)$. We use $\mathcal{A}^*$ to construct $\mathcal{A}^+$.

Since $\mathsf{AdvSel}(\Sigma, \mathcal{A}^*, \lambda)$ is non-negligible, there exist messages m_0, m_1 such that $\mathcal{A}^*$ can distinguish with non-negligible probability between $\mathsf{Enc}(\mathsf{pk}, m_0)$ and $\mathsf{Enc}(\mathsf{pk}, m_1)$ using only oracle queries $k_1 \ldots k_n$ for which $F(m_0, k_i) = F(m_1, k_i)$. Without loss of generality, assume $\mathcal{A}^*$ always outputs the same m_0, m_1 on a given security parameter.

A minor syntactic modification of $\mathcal{A}^*$ yields the desired $\mathcal{A}^+$: define $\mathcal{A}^+$ to be a Turing machine that participates in the Sim_1 security interaction, but uses an internally emulated copy of $\mathcal{A}^*$ to determine its output. That is, when $\mathcal{A}^+$ receives a ciphertext $c = \mathsf{Enc}(\mathsf{pk}, m)$, it starts $\mathcal{A}^*$, discards the plaintexts chosen by $\mathcal{A}^*$ in the first step of the Selective security experiment, passes (pk, c) to $\mathcal{A}^*$ and outputs the bit and oracle queries generated by $\mathcal{A}^*$. Thus, we have $\mathsf{Real}(\Sigma, \mathcal{A}^+, m_0, \lambda) \not\approx \mathsf{Real}(\Sigma, \mathcal{A}^+, m_1, \lambda)$.

However, the restriction that $F(m_0, k_i) = F(m_1, k_i)$ for all oracle queries k_i implies that any $\mathcal{S}$ receives the same information regardless of whether its oracle is parameterized with m_0 or m_1. Thus, $\mathsf{Ideal}(\Sigma, \mathcal{S}, m_0, \lambda) \equiv \mathsf{Ideal}(\Sigma, \mathcal{S}, m_1, \lambda)$ for all $\mathcal{S}$.

Combining these equations implies that either $\mathsf{Real}(\Sigma, \mathcal{A}^+, m_0, \lambda) \not\approx \mathsf{Ideal}(\Sigma, \mathcal{S}, m_0, \lambda)$ or $\mathsf{Real}(\Sigma, \mathcal{A}^+, m_1, \lambda) \not\approx \mathsf{Ideal}(\Sigma, \mathcal{S}, m_1, \lambda)$, which means Σ is not Sim_1-secure.

□

2.2.4 Composability of FE schemes

The MO construction in Chapter 4 uses existing FE schemes to construct a mediated obfuscation of the class of point functions with logarithmic-length[5] output. Towards obfuscating functions with longer output, we show that a natural way of composing FE schemes preserves the two security notions defined above and results in a more expressive FE functionality.

Let Σ_A, Σ_B be distinct FE schemes for the functionalities F_A, F_B, respectively, where F_A and F_B share the same key space. Then, the "composition" of Σ_A and Σ_B, which we denote as $\Sigma_A \| \Sigma_B$, is the FE scheme defined as follows:

$\underline{\mathsf{Setup}^*(1^\lambda)}$:

run $(pk_A, mk_A) \leftarrow \mathsf{Setup}_A(1^\lambda)$
run $(pk_B, mk_B) \leftarrow \mathsf{Setup}_B(1^\lambda)$
output $(pk^*, mk^*) :=$
$((pk_A, pk_B), (mk_A, mk_B))$

$\underline{\mathsf{KeyGen}^*(mk^*, k)}$:

parse mk^* as (mk_A, mk_B)
run $sk_{kA} \leftarrow \mathsf{KeyGen}_A(mk_A, k)$
run $sk_{kB} \leftarrow \mathsf{KeyGen}_B(mk_B, k)$
output $sk_k^* := (sk_{kA}, sk_{kB})$

$\underline{\mathsf{Dec}^*(sk_k^*, c^*)}$:

parse sk_k^* as (sk_{kA}, sk_{kB}) and c^* as (c_A, c_B)
run $y_A \leftarrow \mathsf{Dec}_A(sk_{kA}, c_A)$
run $y_B \leftarrow \mathsf{Dec}_B(sk_{kB}, c_B)$
output $y^* := (y_A, y_B)$

$\underline{\mathsf{Enc}^*(pk^*, (m_A, m_B))}$:

parse pk^* as (pk_A, pk_B)
run $c_A \leftarrow \mathsf{Enc}_A(pk_A, m_A)$
run $c_B \leftarrow \mathsf{Enc}_B(pk_B, m_B)$
output $c^* := (c_A, c_B)$

$\Sigma_A \| \Sigma_B$ then supports the functionality $F_A \| F_B$, defined by:

$$(F_A \| F_B)((m_A, m_B), k) = F_A(m_A, k) \,\|\, F_B(m_B, k).$$

Thus, if F_A and F_B are point functions with single-bit output, then the composed functionality $F_A \| F_B$ has 2-bit output (*i.e.*, $F_A, F_B \to \{0,1\} \Rightarrow F_A \| F_B \to \{0,1\}^2$).

Theorem 2.2.5. *If Σ_A, Σ_B are Selective-secure (resp. straight-line-Sim$_1$-secure) FE schemes for the functionalities F_A, F_B, respectively, then $\Sigma_A \| \Sigma_B$ is Selective-secure (resp. straight-line-Sim$_1$-secure) for the functionality $F_A \| F_B$.*

Proof overview. The proofs use a series of hybrid arguments over the adversaries in the security interactions (Figure 2.1). More generally, if the composed scheme is NOT secure, then the adversary which breaks the security of $\Sigma_A \| \Sigma_B$ could be used as a subroutine

[5] With respect to the security parameter λ.

with internally generated inputs to break the security of either Σ_A or Σ_B, contradicting the assumption that both Σ_A and Σ_B are secure. We first prove the security of $\Sigma_A \| \Sigma_B$ in the Selective security model:

Proof. In the Selective security model, let $\mathcal{A}$ be an adversary that participates in the selective security experiment for the composed FE scheme $\Sigma_A \| \Sigma_B$. For $\mathcal{A}$, SelSecExp runs as follows:

- On input 1^λ, $\mathcal{A}$ submits two message pairs $(m_A, m_B)_0, (m_A, m_B)_1$.
- Compute $(pk^*, mk^*) \leftarrow \mathsf{Setup}^*(1^\lambda)$ and $c^* = \mathsf{Enc}(pk^*, (m_A, m_B)_\beta)$. Give $pk^* = (pk_A, pk_B)$ and $c^* = (c_A, c_B)$ to $\mathcal{A}$.
- $\mathcal{A}$ adaptively submits queries to a $\mathsf{KeyGen}^*(mk^*, \cdot)$ oracle, subject to the constraint that $(F_A \| F_B)((m_A, m_B)_0, k) = (F_A \| F_B)((m_A, m_B)_1, k)$ for all k queried to the oracle.
- $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$.

Let $\mathcal{S}_A$ (resp. $\mathcal{S}_B$) be an adversary that participates in the selective security experiment for Σ_A (resp. Σ_B) by internally emulating $\mathcal{A}$ as well as the FE algorithms for Σ_B (resp. Σ_A) on arbitrary inputs. Assume that $\mathcal{S}_A$ (resp. $\mathcal{S}_B$) outputs the same bit b as its emulated copy of $\mathcal{A}$. On input pk_A, $\mathcal{S}_A$ honestly generates pk_B and sends $pk^* = (pk_A, pk_B)$ to $\mathcal{A}$. $\mathcal{A}$ issues its KeyGen^* queries for $k_1, k_2, \ldots$, and receives responses $sk_k^* = (sk_{kA}, sk_{kB})$ where sk_{kB} is generated internally by $\mathcal{S}_A$ according to KeyGen_B and sk_{kA} is generated by sending k to an external KeyGen_A oracle. Then, $\mathcal{A}$ generates two messages $(m_A, m_B)_0, (m_A, m_B)_1$ for the composed functionality $F_A \| F_B$. Since no key query $(F_A \| F_B)(\cdot, k_i)$ distinguishes between $(m_A, m_B)_0$ and $(m_A, m_B)_1$, we can say that $F_A(\cdot, k)$ (resp. $F_B(\cdot, k)$) doesn't distinguish between the messages $(m_A)_0$ and $(m_A)_1$ (resp. $(m_B)_0$ and $(m_B)_1$). Thus, $(m_A)_0$ and $(m_A)_1$ are passed to the (external) challenger in the indistinguishability game for Σ_A, which responds by sending $\mathcal{S}_A$ an encryption $\mathsf{Enc}(pk_A, (m_A)_\beta)$.

By the construction of $\mathcal{S}_A$, $\Pr[\mathsf{SelSecExp}(\Sigma_A, \mathcal{S}_A, 0, \lambda) = 1] \equiv \Pr[\mathsf{SelSecExp}(\Sigma_A \| \Sigma_B, \mathcal{A}, 0, \lambda) = 1]$, and by the security of Σ_A, $\Pr[\mathsf{SelSecExp}(\Sigma_A, \mathcal{S}_A, 0, \lambda) = 1] \approx \Pr[\mathsf{SelSecExp}(\Sigma_A, \mathcal{S}_A, 1, \lambda) = 1]$.

Now consider the adversary $\mathcal{S}_B$ that participates in the indistinguishability game for Σ_B. Since both $\mathcal{S}_A$ and $\mathcal{S}_B$ work by internally emulating $\mathcal{A}$, and by construction the view of $\mathcal{A}$ (*i.e.*, its inputs and outputs) in $\mathcal{S}_A$ is distributed identically to the view of $\mathcal{A}$ in $\mathcal{S}_B$, we can say that $\Pr[\mathsf{SelSecExp}(\Sigma_A, \mathcal{S}_A, 1, \lambda) = 1] \equiv \Pr[\mathsf{SelSecExp}(\Sigma_B, \mathcal{S}_B, 0, \lambda) = 1]$.

By the construction of $\mathcal{S}_B$ (similar to the argument above), we also have $\Pr[\mathsf{SelSecExp}(\Sigma_B, \mathcal{S}_B, 1, \lambda) = 1] \equiv \Pr[\mathsf{SelSecExp}(\Sigma_A \| \Sigma_B, \mathcal{A}, 1, \lambda) = 1]$, and the security of Σ_B implies that $\Pr[\mathsf{SelSecExp}(\Sigma_B, \mathcal{S}_B, 1, \lambda) = 1] \approx \Pr[\mathsf{SelSecExp}(\Sigma_B, \mathcal{S}_B, 0, \lambda) = 1]$. Thus,

by transitivity we have

$$\Pr\left[\mathsf{SelSecExp}(\Sigma_A \| \Sigma_B, \mathcal{A}, 0, \lambda) = 1\right] \approx \Pr\left[\mathsf{SelSecExp}(\Sigma_A \| \Sigma_B, \mathcal{A}, 1, \lambda) = 1\right]$$

which implies that $\Sigma_A \| \Sigma_B$ is Selective-secure. □

Next, we prove Theorem 2.2.5 for Sim_1-secure FE schemes:

Proof. Let Σ_A, Σ_B be Sim_1-secure FE schemes for the functionalities F_A, F_B respectively. Consider the adversary $\mathcal{A}^{\mathsf{KeyGen}^*(mk^*,\cdot)}$ which attacks the composed scheme $\Sigma_A \| \Sigma_B$. On input $c^* = \mathsf{Enc}^*(pk^*, (m_A, m_B))$, $\mathcal{A}$ submits queries to a KeyGen^* oracle and eventually outputs a bit. Since KeyGen^* operates by individually running the KeyGen algorithms from Σ_A and Σ_B, we can use $\mathcal{A}$ to construct an adversary $\mathcal{A}_A^{\mathsf{KeyGen}_A(mk_A,\cdot)}$ that internally emulates $\mathcal{A}^{\mathsf{KeyGen}^*(mk^*,\cdot)}$ along with all algorithms and inputs for Σ_B, and outputs the same bit as $\mathcal{A}$. Since Σ_A is black-box Sim_1-secure, we can then say that $\mathcal{A}_A^{\mathsf{KeyGen}_A(mk_A,\cdot)} \approx \mathcal{S}_A^{F_A(m_A,\cdot)}$ for some $\mathcal{S}_A$ that internally emulates $\mathcal{A}_A^{\mathsf{KeyGen}_A(mk_A,\cdot)}$ as a straight-line subroutine, given that $\mathcal{A}_A$ and $\mathcal{S}_A$ issue the same oracle queries as the emulated subroutine for $\mathcal{A}$.

Recall that $\mathcal{A}_A$ internally generates inputs to Σ_B. Using this fact, we can construct an adversary $\mathcal{A}_B^{\mathsf{KeyGen}_B(mk_B,\cdot)}$ whose output is identically distributed to $\mathcal{S}_A$, but who receives $\mathsf{Enc}_B(pk_B, m_B)$ externally and queries an external KeyGen_B oracle instead of generating these inputs internally. This construction implies $\mathcal{S}_A^{F_A(m_A,\cdot)} \equiv \mathcal{A}_B^{\mathsf{KeyGen}_B(mk_B,\cdot)}$.

Invoking the straight-line, black-box Sim_1-security of Σ_B then yields a simulator $\mathcal{S}_B$ that internally emulates $\mathcal{S}_A^{F_A(m_A,\cdot)}$ as a subroutine, such that $\mathcal{A}_B^{\mathsf{KeyGen}_B(mk_B,\cdot)} \approx \mathcal{S}_B^{F_B(m_B,\cdot)}$ when $\mathcal{A}_B$ and $\mathcal{S}_B$ issue the same oracle queries as $\mathcal{A}$. Finally, recall that the original (emulated) adversary $\mathcal{A}$ runs as a subroutine within $\mathcal{S}_B$, and every KeyGen^* query made by $\mathcal{A}$ on input k is answered by an emulated oracle for $F_A(m_A, k)$ as well as an external oracle for $F_B(m_B, k)$. This combination of two separate oracles for F_A and F_B operating on the same input k can then be replaced with one query to an oracle for $(F_A \| F_B)((m_A, m_B), \cdot)$, which means we can construct $\mathcal{S}^{(F_A \| F_B)((m_A,m_B),\cdot)} \equiv \mathcal{S}_B^{F_B(m_B,\cdot)}$. Then, by transitivity

$$\mathcal{A}^{\mathsf{KeyGen}^*} \approx \mathcal{S}^{(F_A \| F_B)((m_A,m_B),\cdot)}$$

which implies that $\Sigma_A \| \Sigma_B$ is Sim_1-secure. □

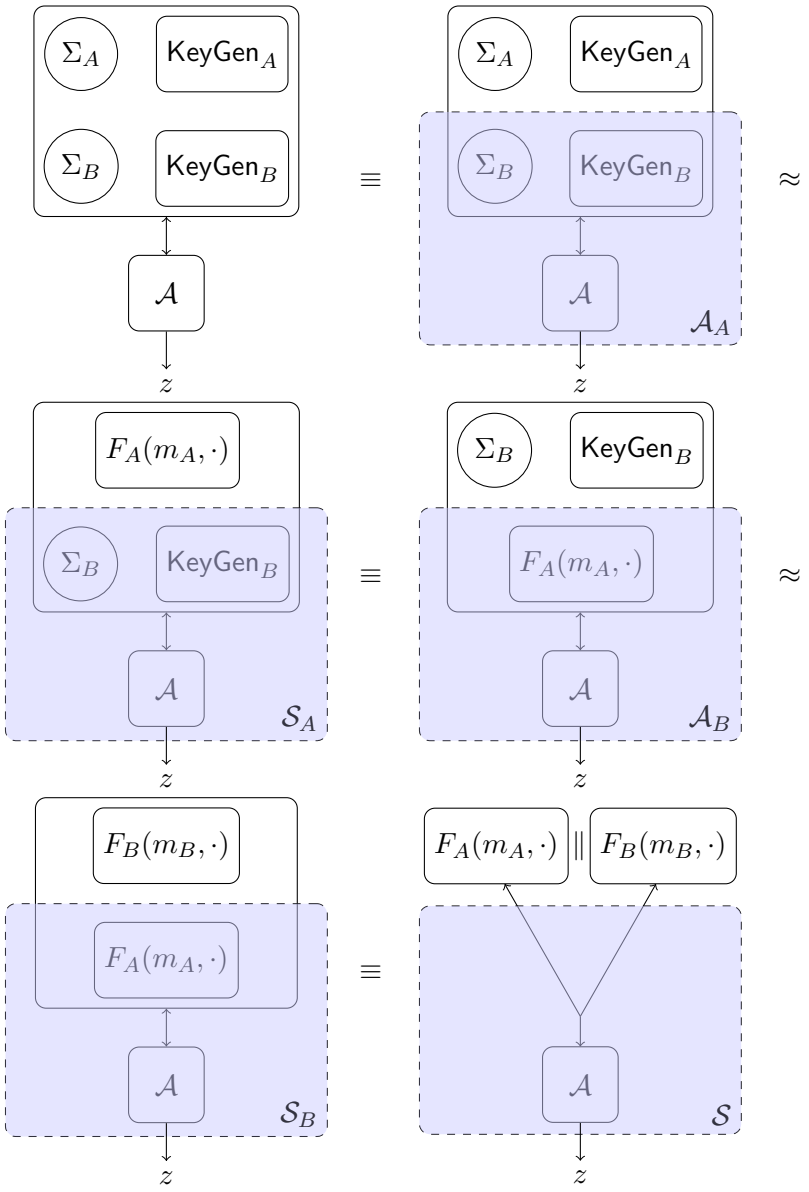

Figure 2.1: Steps in the proof of the composition theorem for straight-line-Sim_1-security.

2.3 Obfuscation

A program f is **obfuscated** if it cannot be "reverse-engineered" — that is, the source code of f reveals no more information than can be learned from oracle access to f. Obfuscation has a wide range of applications, most obviously in the realm of software protection.

Definition 2.3.1. *An algorithm $\mathcal{O}_f$ is an obfuscation of the function (Turing machine) f if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ there exists an oracle PPT simulator $\mathcal{S}$ such that the quantity*

$$\left|\Pr\left[\mathcal{A}(\mathcal{O}_f) = 1\right] - \Pr\left[\mathcal{S}^f(1^{|f|}) = 1\right]\right| \text{ is a negligible function of } |f|.$$

Essentially, Definition 2.3.1 implies that nothing meaningful can be learned by examining the description of an obfuscated program: at best, an obfuscated program $\mathcal{O}_f$ is indistinguishable from an idealized, inviolable "oracle" that does nothing but accept inputs x and return $f(x)$.

The study of program obfuscation in the context of provable security was initiated by Barak *et al.* [BGI+01], where it was famously shown that general-purpose obfuscation is impossible, even for relatively weak definitions of (static) obfuscation. This result is achieved by constructing a class of Turing machines (TM) for functions that can neither be obfuscated nor deduced via oracle queries alone. The unobfuscatable TM from [BGI+01] is defined in terms of its two "subroutine" Turing machines summarized here:

First, for strings $x, m \in \{0,1\}^k$ define the TM that computes a point function with output

$$I_{x,m}(y) := \begin{cases} m & \text{if } x = y \\ 0^k & \text{otherwise} \end{cases}$$

Then define a Turing machine J that when given the description of a Turing machine $\langle K \rangle$ determines whether K computes the same function[6] as $I_{x,m}$ or whether K computes $I_{x',m'}$ for any $(x', m') \neq (x, m)$

[6] As presented, J is uncomputable. However, it suffices for J to consider whether $K = I_{x,m}$ *in a polynomial number of steps*.

$$J_{x,m}(\langle K \rangle) := \begin{cases} 1 & \text{if } K(x) = m \\ 0 & \text{otherwise} \end{cases}$$

Now consider an adversary $\mathcal{A}$ that is given obfuscations $\mathcal{O}_{I_{x,m}}, \mathcal{O}_{J_{x,m}}$ of the functions $I_{x,m}$ and $J_{x,m}$. $\mathcal{A}$ can simply run $\mathcal{O}_{J_{x,m}}$ on input $\langle \mathcal{O}_{I_{x,m}} \rangle$, and the correctness property of obfuscation implies that the result of $\mathcal{O}_{J_{x,m}}(\langle \mathcal{O}_{I_{x,m}} \rangle) = 1$, that is

$$\Pr\left[\mathcal{A}(\mathcal{O}_{J_{x,m}}, \mathcal{O}_{I_{x,m}}) = 1\right] = 1$$

Since $I_{x,m}$ returns a non-zero value on only one point, and $J_{x,m}$ is non-zero on negligibly few points in the space of all Turing machines, any polynomial-time algorithm $\mathcal{S}$ with oracle access to $I_{x,m}$, $J_{x,m}$ has a negligibly small probability of querying either oracle on an input where the value of the function is not zero. Thus, even with the additional information yielded by oracle access to the function $I_{x,m}$, it is highly unlikely that any simulator will be able to generate a TM description $\langle M \rangle$ such that $J_{x,m}(\langle M \rangle) = 1$. Formally:

$$\forall \mathcal{S} : \Pr\left[\langle M \rangle \leftarrow \mathcal{S}^{I_{x,m}, J_{x,m}}(1^k) : J_{x,m}(\langle M \rangle) = 1\right] \text{ is negligible.}$$

Thus, the "virtual black-box" notion of obfuscation from Definition 2.3.1 is impossible for arbitrary functions. This construction formalizes the intuitive notion that access to the source code of a program is a significant advantage to algorithms that operate in polynomial time.

Positive results for obfuscations have mostly been limited to point functions [LPS04, Wee05, CD08], yet even point functions require some necessary relaxations of the original definitions (cf. [Wee05]). Hofheinz *et al.* [HMLS10] proposed a very natural relaxation of the security definitions of [BGI+01]; namely, they considered obfuscating functions which are drawn at random from a class (average-case obfuscation). By comparison, the previous definition required security to hold equally for every function in the class (worst-case obfuscation). Average-case obfuscation is a natural choice when obfuscating a cryptographic primitive, where the obfuscation needs only to hide the hard-coded secret key that was sampled honestly.[7] Several expressive cryptographic primitives have features

[7] The model proposed by Hofheinz *et al.* also considers the problem of obfuscating randomized functions.

that can be understood as being average-case obfuscations of certain functionalities: *e.g.*, proxy-re-encryption [HRsV11] and encrypted signatures [Had10].

Our definitions for mediated obfuscation require that client and mediator engage in a one-round protocol in order for the client to evaluate the obfuscated program. In our main feasibility result, we use existing constructions for one-round secure computation [CCKM00, IKO+11], as well as other highly powerful cryptographic tools like fully homomorphic [Gen09] and targeted non-malleable [BSW11b] encryption schemes which naturally lend themselves to one-round protocols.

2.4 Universally Composable Security

The Universally Composable (UC) framework is a system for establishing and analyzing the security of cryptographic protocols [Can00]. In this work, the UC framework is used to demonstrate the general feasibility of mediated obfuscation schemes based on secure 1-round protocols for non-interactive computation.

Interactive Turing machines (ITMs) form the basis of the UC framework's computational model. Interactive Turing machines are Turing machines with a special "shared tape" which can be read and modified (written into) by other ITMs [GMR89]. In the UC framework, the execution of a protocol in an arbitrary network environment is modeled with a system of ITMs that are appropriately connected via their shared tapes.

The main entities in the UC model are outlined below:

Functionalities ($\mathcal{F}$) represent desirable cryptographic tasks; in the UC model, a functionality is a trusted entity that receives (possibly empty) inputs from parties $P_0, P_1, \ldots, P_n$, performs some computational task on those inputs, and returns appropriate outputs to the parties.

Protocols ($\pi, \rho, \phi, \ldots$) define methods for interacting with functionalities. Generally, an *honest party* is a uniquely-identified ITM interacting with a functionality $\mathcal{F}$ by running the code of protocol π; *corrupt parties* interact with $\mathcal{F}$ but run arbitrary code.

Environments ($\mathcal{Z}$) model all network activity external to the parties executing π (or otherwise interacting with $\mathcal{F}$). The environment interacts with all parties and

the adversary, and outputs a single bit on the completion of protocol execution.

Adversaries $(\mathcal{A}, \mathcal{S})$ attempt to extract secret information from the honest parties or otherwise manipulate protocol execution by corrupting one or more honest parties. "Corruption" of an honest party is modeled by allowing the adversary total control over the corrupt party's interaction with the functionality. The adversary also communicates with the environment.

The UC framework uses the *trusted third-party* paradigm to establish the security of cryptographic protocols. For any given cryptographic task, an *ideal functionality* $\mathcal{F}$ is defined which represents a set of instructions that tell a trusted party how to carry out that task. As with the other entities in the UC framework, the ideal functionality is modeled as an ITM.

Interactions with ideal functionalities are carried out according to an *ideal protocol* via a set of *dummy parties* $\{\tilde{P}_0, \tilde{P}_1, \ldots, \tilde{P}_n\}$. Dummy parties simply relay messages between the ideal functionality and the other entities in the UC model.. The ideal protocol, denoted $\text{IDEAL}_{\mathcal{F}}$, represents an incorruptible means to achieve a task $\mathcal{F}$ and is necessarily impossible to realize in the real world.

Security guarantees in the UC framework are established relative to the ideal protocol. A real-world cryptographic protocol π is said to *securely realize* an ideal functionality $\mathcal{F}$ if the protocol can be mathematically shown to emulate the ideal protocol $\text{IDEAL}_{\mathcal{F}}$.

Security proofs in the UC framework are derived by considering a model that captures both the influence of the adversary on protocol execution and the impact of protocol execution on the environment. This model consists of a network of ITMs representing the protocol π, the adversary $\mathcal{A}$, and the environment $\mathcal{Z}$. Proofs in this model typically yield statements like, "protocol π emulates protocol ρ," or "protocol π securely realizes functionality $\mathcal{F}$ because it emulates $\text{IDEAL}_{\mathcal{F}}$."

Protocol emulation is established by considering the interaction between the protocol and the adversary *from the point of view of the environment*. Informally, a protocol π is said to emulate another protocol ρ if, for any given environment $\mathcal{Z}$, it is possible to construct a *simulator* $\mathcal{S}$ such that interactions between π and the adversary $\mathcal{A}$ "look the same" as interactions between ρ and the simulator $\mathcal{S}$.

More formally, let $\text{EXEC}[\pi, \mathcal{A}, \mathcal{Z}]$ be the random variable representing the environment $\mathcal{Z}$'s output after adversary $\mathcal{A}$ interacts with protocol π. Then, protocol π emulates protocol

ρ if:

$$\forall \mathcal{A} \exists \mathcal{S} \forall \mathcal{Z} : \text{EXEC}[\mathcal{Z}, \mathcal{A}, \pi] \approx \text{EXEC}[\mathcal{Z}, \mathcal{S}, \rho]$$

where $\approx$ denotes computational indistinguishability. If ρ is the ideal protocol $\text{IDEAL}_{\mathcal{F}}$, then π is said to *UC-realize* functionality $\mathcal{F}$.

Furthermore, the UC framework can be extended to admit secure protocols for essentially all computable functions by assuming a priori access to an appropriate set-up functionality. The UC framework augmented with black-box access to a functionality $\mathcal{G}$ is called the $\mathcal{G}$*-hybrid* model.

The most extensively studied set-up assumption is the *common reference string* functionality $\mathcal{F}_{\text{CRS}}$ (Figure 2.2) [CF01], which is parameterized by a distribution D and set of parties $\mathcal{P}$ participating in a protocol instance. The $\mathcal{F}_{\text{CRS}}$-hybrid model assumes that all protocol instances have access to a unique trusted party that can provide, upon request, a common string chosen uniformly at random from the uniform distribution over strings of some length.

- On input REQUEST from party P: verify that party P is a participant in the protocol instance. If $P \notin \mathcal{P}$, abort.
- If there is no random string r already recorded, choose and record $r \leftarrow D$.
- Send public output r to P.

Figure 2.2: The common reference string functionality $\mathcal{F}_{\text{CRS}}$ for a distribution D and a set of parties $\mathcal{P}$.

Chapter 3

Mediated Obfuscation

In this chapter, we present the mediated obfuscation (MO) model, security definitions, and general feasibility based on non-malleable secret sharing and fully homomorphic encryption. Mediated obfuscation is a natural relaxation of static obfuscation that employs a limited amount of interaction to securely evaluate a function.

3.1 Model

In the mediated obfuscation (MO) model (Figure 3.1), a **vendor** wants to provide a **client** with the ability to evaluate a function f without revealing f and without remaining online for the client's convenience. To accomplish this, the vendor employs a trusted **mediator** to interact with the client.

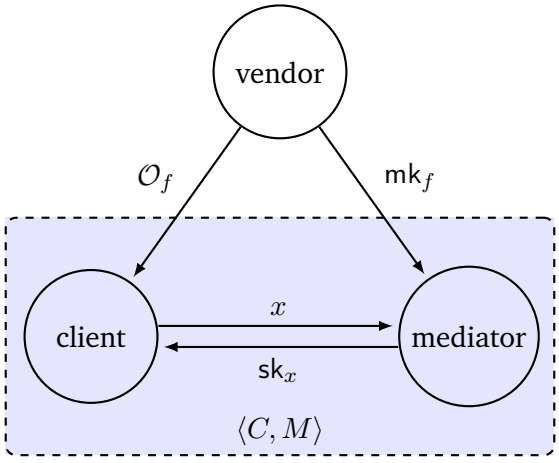

Figure 3.1: The mediated obfuscation model.

The security requirements are modeled via an ideal functionality $\mathcal{F}_{\text{MO}}$ (Figure 3.2), parameterized by a class of functions $\mathcal{C}$. For simplicity, the roles of the three parties are fixed in the description of $\mathcal{F}_{\text{MO}}$. In an interaction with $\mathcal{F}_{\text{MO}}$, the client has only oracle access to the function f. The mediator simply guards this access, possibly based on the inputs being sent.

- On input f from the vendor, where $f \in \mathcal{C}$: if any value f' is previously recorded, then abort. Otherwise, internally record f and send OK to the client and mediator. If both the client and mediator are corrupt, also send f to both parties.
- On input x from the client, abort if no value f was previously recorded. Otherwise, send the value INPUT to the mediator and internally record x.
- On input OK from the mediator, abort if no value f was previously recorded, or if no value x was previously recorded. Otherwise, compute $y = f(x)$ and give output y to the client and output OK to the mediator.

Figure 3.2: The $\mathcal{F}_{\text{MO}}$ functionality for mediated obfuscation of a class of functions $\mathcal{C}$.

3.2 Syntax

The functionality $\mathcal{F}_{\text{MO}}$, along with the following syntactic constraints, defines the syntax of MO:

Definition 3.2.1. *A* **mediated obfuscation (MO) scheme** $\Delta = (\mathsf{Obfu}, \langle C, M \rangle)$ *for the class of functions $\mathcal{C}$ is a protocol for the $\mathcal{F}_{\text{MO}}$ functionality described in Figure 3.2, where* Obfu *is the vendor's protocol function and $\langle C, M \rangle$ is a 1-round, 2-party interactive protocol between the client and mediator. The function* Obfu *is defined as follows:*

- $(\mathcal{O}_f, \mathsf{mk}_f) \leftarrow \mathsf{Obfu}(1^\lambda, f)$*: accepts a security parameter and a description of a function (or circuit) $f \in \mathcal{C}$, and outputs an obfuscated version of the function along with a master key for the function.*
- *The vendor sends $\mathcal{O}_f$ to the client and mk_f to the mediator.*

Once $\mathcal{O}_f$ and mk_f are distributed, the vendor's role in the protocol is finished: all further interaction takes place between the client and the mediator. When the client wants to evaluate f on input x, she engages in the 2-party protocol $\langle C, M \rangle$ with the mediator. The joint execution of

the client, holding inputs $\mathcal{O}_f, x$, and the mediator, holding input mk_f, results in an output $f(x)$ for the client assuming mk_f is a valid key for f. We write this as $f(x) \leftarrow \langle C[\mathcal{O}_f, x], M[\mathsf{mk}_f]\rangle$. We require that $\langle C, M\rangle$ is 1-round (send and receive one message) and stateless across different values of x.

3.3 Security Definitions

Next, we define the security conditions for MO. These definitions capture the intuitive notion that the client shouldn't learn anything more than she would with oracle access to the function f.

Definition 3.3.1. *We define the following levels of security for an MO scheme:*

1. **Full security:** *The MO scheme is a secure protocol for the $\mathcal{F}_{\text{MO}}$ functionality (Figure 3.2).*

2. **Relaxed security:** *The MO scheme is a secure protocol for the $\mathcal{F}_{\text{MOX}}$ functionality (Figure 3.3) which gives x to the mediator.*

   - On input f from the vendor, where $f \in \mathcal{C}$: if any value f' is previously recorded, then abort. Otherwise, internally record f and send OK to the client and mediator. If both the client and mediator are corrupt, also send f to both parties.
   - On input x from the client, abort if no value f was previously recorded. Otherwise, send INPUT and x to the mediator and internally record x.
   - On input OK from the mediator, abort if no value f was previously recorded, or if no value x was previously recorded. Otherwise, compute $y = f(x)$ and give output y to the client and output OK to the mediator.

   Figure 3.3: The $\mathcal{F}_{\text{MOX}}$ functionality for mediated obfuscation of a class of functions $\mathcal{C}$.

3. **Semantic security:** *Let Δ denote the MO scheme, and define the following interaction:*

   $\underline{\mathsf{MOSSExp}(\Delta, \mathcal{A}, \beta, \lambda)}$:

   (a) Give 1^λ to $\mathcal{A}$, who then outputs (descriptions of) two functions $f_0, f_1 \in \mathcal{C}$.

   (b) Compute $(\mathcal{O}_{f_\beta}, mk_\beta) \leftarrow \mathsf{Obfu}(f_\beta, 1^\lambda)$. Give $(1^\lambda, \mathcal{O}_{f_\beta})$ to $\mathcal{A}$.

   (c) $\mathcal{A}$ is allowed to repeatedly engage in the protocol with an honest party running $M[\mathsf{mk}_{f_\beta}]$. If M ever aborts, or outputs x such that $f_0(x) \neq f_1(x)$, the entire interaction halts.

   (d) $\mathcal{A}$ outputs a bit, which we define as the output of this interaction.

Define the advantage of the adversary $\mathcal{A}$ as:

$$\mathsf{MOSSAdv}(\Delta, \mathcal{A}, \lambda) := \left| \begin{array}{l} \Pr\left[\mathsf{MOSSExp}(\Delta, \mathcal{A}, 1, \lambda) = 1\right] \\ \quad - \Pr\left[\mathsf{MOSSExp}(\Delta, \mathcal{A}, 0, \lambda) = 1\right] \end{array} \right|$$

Then Δ is **semantically secure** *if relaxed security holds when the security condition against a corrupt client is replaced with the following condition: for all PPT $\mathcal{A}$ the function $\mathsf{MOSSAdv}(\Delta, \mathcal{A}, \lambda)$ is negligible in λ.*

We point out that for certain classes of functions (*e.g.*, collision-resistant hash functions), this definition of semantic security is somewhat vacuous since it may be computationally infeasible for the adversary to find two functions $f_0 \neq f_1$ and an input x which satisfy the requirement that $f_0(x) = f_1(x)$. An analogous drawback to semantic-security-style definitions for functional encryption was observed by O'Neill [O'N10].

3.4 General Feasibility

We outline two constructions of mediated obfuscation schemes that achieve full security and support obfuscation of *arbitrary* functions. However, these constructions make use of encryption primitives that are both *powerful* and *inefficient*, leaving room for the simpler construction based on functional encryption presented in Chapter 4.

3.4.1 Using one-round secure computation

A non-malleable secret sharing (NMSS) scheme [IPS08] consists of two algorithms, Split and Join. We require that if $(\alpha, \beta) \leftarrow \mathsf{Split}(x)$, then the marginal distributions of α and β are each independent of x, but that $\mathsf{Join}(\alpha, \beta) = x$. The non-malleability property of the scheme is that, for all x and PPT adversaries $\mathcal{A}$:

$$\Pr\left[(\alpha, \beta) \leftarrow \mathsf{Split}(x); \beta' \leftarrow \mathcal{A}(\beta, x) : \beta' \neq \beta \wedge \mathsf{Join}(\alpha, \beta') \neq \bot\right] \text{ is negligible.}$$

Using NMSS as a building block, the MO construction is as follows:

$\underline{\mathsf{Obfu}(f, 1^\lambda)}$:

1. Sample $r \leftarrow \{0,1\}^\lambda$ and send r to both the client and mediator.
2. Generate $(\alpha, \beta) \leftarrow \mathsf{Split}(\langle f \rangle)$. Set $\mathcal{O}_f := \alpha$ and $\mathsf{mk}_f := \beta$.

3. Send α to the client and β to the mediator.

$\langle C, M \rangle$ protocol (client has input $(\mathcal{O}_f, x)$; mediator has input mk_f):

1. Run a one-round, two-party protocol for the $\mathcal{F}_{\text{JOIN}}$ functionality (Figure 3.4).

- On input $(\mathcal{O}_f, x)$ from the client, record $(\mathcal{O}_f, x)$ and send the value INPUT to the mediator.
- On input mk_f from the mediator, record mk_f. Abort if no value $(\mathcal{O}_f, x)$ was previously recorded.
- Set $f := \mathsf{Join}(\mathcal{O}_f, \mathsf{mk}_f)$; if $f = \perp$, then abort.
- Give $f(x)$ to the client.
- Erase all previously recorded values.

Figure 3.4: The $\mathcal{F}_{\text{JOIN}}$ functionality for the $\langle C, M \rangle$ subprotocol using NMSS.

Several works show how the $\langle C, M \rangle$ protocol can be carried out in a single round of interaction [CCKM00, IKO+11]. In particular, Ishai *et al.* [IKO+11] give UC-secure, 1-round protocols in the $\mathcal{F}_{\text{CRS}}$-hybrid model (Figure 2.2). In our setting, the vendor can honestly sample and provide a common reference string to both parties, eliminating the need for trusted setup.

Theorem 3.4.1. *The NMSS-MO construction above is a fully-secure MO scheme for the class of functions whose description length is bounded by λ.*

Proof. In this construction, we can apply the UC security of the two-party subprotocol, so that the MO protocol is operating in a hybrid model providing ideal access to the $\mathcal{F}_{\text{JOIN}}$ functionality defined in Figure 3.4. We first consider security against a corrupt client. The simulator in the $\mathcal{F}_{\text{MO}}$-hybrid model is as follows: First, the simulator chooses a random α and gives it to the client. By the secret sharing property of the NMSS, the simulated α is distributed identically to the real interaction. When the client gives (α', x) to the hybrid functionality, the simulator checks whether $\alpha \stackrel{?}{=} \alpha'$. If not, then the simulator aborts; otherwise, the simulator sends x to $\mathcal{F}_{\text{MO}}$. By the non-malleability property of the NMSS scheme, we have that an honest mediator would have aborted with overwhelming probability if $\alpha \neq \alpha'$, so the simulation is sound.

Next, we consider security against a corrupt mediator. The simulation is extremely similar to the previous case. The simulator provides a random β. When the mediator sends

β to the hybrid functionality, the simulator sends OK to $\mathcal{F}_{\text{MO}}$. By similar reasoning to above, this simulation is sound. □

3.4.2 Using (derivatives of) fully-homomorphic encryption

A targeted-malleable homomorphic encryption scheme [BSW11b] supports a limited set of homomorphic ciphertext operations as features; all others ways of generating related ciphertexts are infeasible.

Definition 3.4.2 (Targeted-malleability)**.** *Let* $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a public-key encryption scheme. Define the following interactions:*

$\underline{\mathsf{Real}(\Pi, \mathcal{A}, \lambda)}$*:*

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(\mathcal{M}, \text{STATE}_1, \text{STATE}_2) \leftarrow A_1(\mathsf{pk}, 1^\lambda)$
3. $(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
4. $c_i^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_i)$ *for all* $i \in \{1, \ldots, r\}$
5. $(c_1, \ldots, c_q) \leftarrow A_2(c_1^*, \ldots, c_r^*, \text{STATE}_2, 1^\lambda)$
6. *For every* $j \in \{1, \ldots, q\}$ *let*
$$d_j := \begin{cases} \text{COPY}_i & \text{if } c_j = c_i^* \\ \mathsf{Dec}_{\mathsf{sk}}(c_j) & \text{otherwise} \end{cases}$$
7. *Output* $(\text{STATE}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$

$\underline{\mathsf{Ideal}(\Pi, \mathcal{S}, \lambda)}$*:*

1. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
2. $(\mathcal{M}, \text{STATE}_1, \text{STATE}_2) \leftarrow S_1(\mathsf{pk}, 1^\lambda)$
3. $(m_1, \ldots, m_r) \leftarrow \mathcal{M}$
4. $(c_1, \ldots, c_q) \leftarrow S_2(\text{STATE}_2, 1^\lambda)$
5. *For every* $j \in \{1, \ldots, q\}$ *let*
$$d_j := \begin{cases} \text{COPY}_i & \text{if } c_j = \text{COPY}_i \\ f(m_i) & \text{if } c_j = (i, f_1, \ldots, f_l) \text{ where } i \in \{1, \ldots, r\}, \\ & l \le t, f_1, \ldots, f_l \in \mathcal{F}, \\ & \text{and } f = f_1 \circ \cdots \circ f_l \\ \mathsf{Dec}_{\mathsf{sk}}(c_j) & \text{otherwise} \end{cases}$$
6. *Output* $(\text{STATE}_1, m_1, \ldots, m_r, d_1, \ldots, d_q)$.

We say that Π *is* non-malleable against chosen-plaintext attacks with respect to a class of functions $\mathcal{F}$ *if for all PPT* $\mathcal{A} = (A_1, A_2)$ *there exists a PPT simulator* $\mathcal{S} = (S_1, S_2)$ *such that the distribution ensembles* $\{\mathsf{Real}(\Pi, \mathcal{A}, \lambda)\}_\lambda$ *and* $\{\mathsf{Ideal}(\Pi, \mathcal{S}, \lambda)\}_\lambda$ *are indistinguishable.*

Let $\mathcal{E}(\langle C \rangle, x) := C(x)$ denote a universal circuit. Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a targeted-malleable encryption scheme supporting homomorphic operations $\mathsf{Enc}(y) \mapsto \mathsf{Enc}(\mathcal{E}(y, x) \oplus s)$ for all x, s. Then the MO construction is as follows:

$\underline{\mathsf{Obfu}(f, 1^\lambda)}$:

1. Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and set $c = \mathsf{Enc}(\mathsf{pk}, \langle f \rangle)$.
2. Set $\mathcal{O}_f := (\mathsf{pk}, c)$ and $\mathsf{mk}_f := \mathsf{sk}$.

$\underline{\langle C, M \rangle \text{ protocol (client has input } (\mathcal{O}_f, x); \text{ mediator has input } \mathsf{mk}_f):}$

1. The client chooses random string s and uses the homomorphic properties of the scheme to obtain $c' = \mathsf{Enc}(\mathcal{E}(\langle f \rangle, x) \oplus s)$. He sends c' to the mediator.
2. The mediator decrypts c' and sends the result to the client. The client strips s and obtains $f(x)$ as the result.

Theorem 3.4.3. *A targeted-malleable homomorphic encryption scheme can be used to construct a relaxed-secure MO scheme for the class of functions whose description length is bounded by λ.*

Proof. We first consider security against a corrupt client. The simulator provides the client with an honestly generated key pk and a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{pk}, 0^\lambda)$. The definition of targeted non-malleability implies that there is a simulator which can "extract" the homomorphic operation applied to a ciphertext. Thus when the client provides a ciphertext c', our simulator extracts the parameters x and s identifying the valid homomorphic operation that was applied to c. If this extraction is successful, the simulator sends x to $\mathcal{F}_{\text{MOX}}$. Upon receiving $y = f(x)$ from $\mathcal{F}_{\text{MOX}}$, the simulator hands the client an encryption $\mathsf{Enc}(\mathsf{pk}, y \oplus s)$, where s was the value extracted previously.

For security against a *semi-honest* mediator, we simply observe that by the homomorphic properties of the scheme, and the random choice of mask s chosen by the client, the mediator sees only a random-looking encryption of a random value in each interaction of the protocol. □

Chapter 4

Functional Encryption as Mediated Obfuscation

In this chapter, we demonstrate that secure mediated obfuscations can be constructed from existing functional encryption systems. First, we present a purely generic MO construction based on functional encryption syntax. Then, we show that the Okamoto-Takashima IBE scheme described in the full version of [OT10, §G.3] actually satisfies Sim_1-security for AIBE, provided that the message space is polynomial in size. Thus, this scheme yields a mediated obfuscation for the class of point functions with logarithmic-length output, which can be concatenated via Theorem 2.2.5 to provide mediated obfuscations of arbitrary functions.

4.1 Generic Construction from Functional Encryption

Given a functional encryption scheme $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the functionality $F(m, k)$, a mediated obfuscation scheme $\Delta = (\mathsf{Obfu}, \langle M, C \rangle)$ for the class of functions $\mathcal{C} = \{F(m, \cdot) \mid m\}$ can be naturally constructed as follows:

$\mathsf{Obfu}(f, 1^\lambda)$:

1. Run $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$, where m is such that $f(\cdot) \equiv F(m, \cdot)$.
2. Set $\mathcal{O}_f := c$ and $\mathsf{mk}_f := \mathsf{mk}$. Output $(\mathcal{O}_f, \mathsf{mk}_f)$.

$\langle C, M \rangle$ protocol (client has input $(\mathcal{O}_f, x)$; mediator has input mk_f):

1. The client sends x to the mediator.
2. The mediator runs $sk_x \leftarrow \mathsf{KeyGen}(\mathsf{mk}, x)$ and sends sk_x to the client.

3. The client runs $\mathsf{CheckKey}(\mathsf{pk}, \mathsf{sk}_x, x)$, and aborts if CheckKey returns 0.
4. The client computes and outputs $y \leftarrow \mathsf{Dec}(\mathsf{sk}_x, \mathcal{O}_f)$. The FE correctness properties ensure that $y = F(m, x)$.

Theorem 4.1.1. *If Σ is a Sim_1-secure (resp. Selective-secure) FE scheme with verifiable keys for functionality F, then the above construction Δ is a relaxed-secure (resp. semantically-secure) MO scheme for class of functions $\mathcal{C} = \{F(m, \cdot) \mid m\}$.*

Proof overview. Correctness of our construction follows from the correctness properties of the FE scheme. Security against a corrupt (semi-honest) mediator follows from the fact that such a mediator's view consists only of mk_f and the queries $x_1, \ldots, x_k$ of the client. mk_f is distributed independently of m (the index of $F(m, \cdot) \in \mathcal{C}$), and the client's queries are also available to the mediator in the ideal interaction with $\mathcal{F}_{\text{MOX}}$.

Proof. For security against a corrupt (malicious) client, we first consider the case where Σ is Sim_1-secure. The client receives a ciphertext $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ and then, via the $\langle C, M \rangle$ protocol, effectively uses the mediator as an oracle for $\mathsf{KeyGen}(\mathsf{mk}, \cdot)$. Thus the client is a valid adversary in the Sim_1 security experiment[1] for Σ. There exists a simulator $\mathcal{S}$ that simulates the view of the client, given only black-box access to $F(m, \cdot)$. This simulation can thus be carried out in the $\mathcal{F}_{\text{MOX}}$-ideal model, since $\mathcal{F}_{\text{MOX}}$ provides the simulator access to a (guarded) oracle for $F(m, \cdot)$. This establishes full security for the MO scheme against a corrupt client.

Similarly, in the semantic security game for MO against a corrupt client, the client first specifies two functions $f_0, f_1 \in \mathcal{C}$ and then receives an obfuscation of f_β. The client is further allowed to repeatedly interact with an honest mediator subject to the constraint that the mediator reports an input x for the client such that $f_0(x) = f_1(x)$. As the mediator is simply acting as an oracle to $\mathsf{KeyGen}(\mathsf{mk}_f, \cdot)$, it can easily be seen that the MO semantic security game can be carried out within the FE Selective-security game in a straight-forward manner. □

4.1.1 On full vs. relaxed security

The above construction achieves relaxed MO security (meaning that the mediator learns x). In some settings involving software rental/leasing, it is in fact desirable for the mediator

[1] The Sim_1 game also gives pk to the adversary, but it is not needed for our MO construction.

to obtain the client's inputs for accounting or policy-enforcement purposes. However, if leakage of x is undesirable, then a straight-forward transformation can convert this construction to full MO security (in which the mediator may be malicious but does not learn x) as follows:

First, the vendor generates a sufficiently long random string s and gives it to both the client and mediator. Then the client and mediator can treat s as a common random string and run a UC-secure subprotocol for evaluating the KeyGen function (with the client providing x and the mediator providing mk, and only the client receiving output). As above, this can be done in a single round of interaction [IKO+11]. By the UC-security of the subprotocol, whatever the client can learn in this modified protocol can be learned with access to a $\mathsf{KeyGen}(\mathsf{mk}, \cdot)$ oracle, as the mediator provides in the simpler protocol above. If the FE scheme satisfies the notion of key verifiability given in Definition 2.2.1, then this transformation achieves full MO security.

Remark 4.1.2. *Without loss of generality, we can assume in the MO scheme that the same query is not given twice to* KeyGen. *This can be enforced by letting the mediator keep track of queries, or by having the mediator evaluate a version of* KeyGen *that has been derandomized with a pseudorandom function.*

4.2 Implementing Mediated Obfuscation

The generic construction presented in Section 4.1 suggests that implementing MO with existing FE schemes should be relatively straightforward. We outline the requirements for meeting the security definitions given in Section 2.2.3, and then demonstrate that the AIBE instantiation of the general scheme from [OT10] satisfies key-verifiability and the definitions of both semantic security and strong simulation-based security.

4.2.1 Achieving semantic security

Modern FE schemes have focused on *predicate encryption*. That is, a "payload" encrypted with parameter y can be decrypted by the key corresponding to parameter x only if $P(x, y) = 1$. Our application requires a notion of "attribute-hiding" — namely, that the ciphertext hides parameter y in addition to the payload. The current state of the art for such predicate encryption schemes is for *inner-product* predicates; using the notation above,

these schemes use the predicate $P(x, y) = [\langle x, y\rangle \stackrel{?}{=} 0]$, where x and y are interpreted as vectors.

To achieve semantic-security for our MO scheme instantiation, we require only a Selective-secure (but attribute-hiding) FE scheme. Katz, Sahai, and Waters constructed such a Selective-secure inner-product predicate encryption scheme [KSW08]. Okamoto & Takashima [OT11] have recently published a fully-secure, attribute-hiding scheme as well.[2]

Predicate-encryption schemes encrypt a ciphertext with a fixed payload m, and the decryption is *all-or-nothing*, resulting in either m or an error $\perp$. Thus, these schemes can be used to obfuscate a function with a single bit of output (using a public, fixed m and interpreting m as 1, $\perp$ as 0). Then multiple FE schemes can be combined/concatenated (as outlined in Section 2.2.4) to obfuscate a function with longer output. Thus, to obfuscate a function f in our model, it suffices to obfuscate $f_1, \ldots, f_k$, where f_i is the function which gives the i-th bit of f's output.

Furthermore, we note that polynomial evaluation ("does $p(x) = 0$?"), CNF, and DNF formulas can all be expressed using inner products [KSW08]. Thus a very wide class of functions can be obfuscated in our model with semantic security.

Theorem 4.2.1. *A secure attribute-hiding predicate encryption scheme Σ which implements the predicate $P(x, y) := [\langle x, y\rangle \stackrel{?}{=} 0]$ can be used to evaluate CNF/DNF formulæ.*

Proof overview. Polynomials can be evaluated using inner-product relations, and CNF and DNF formulæ can be written as multivariate polynomials. We demonstrate that such an encoding reveals no information about the Boolean expression to be evaluated (though keys do reveal the values tested in the logical formula).

Proof. To evaluate a polynomial in m variables with total degree n, first order the $\binom{n+m}{n}$ terms lexicographically (shown here with $m = 2$, ordered by variable and then exponent):

$$p(x, y) = a_1 x^n y^0 + a_2 x^{n-1} y^1 + a_3 x^{n-1} y^0 + a_4 x^{n-2} y^2 + a_5 x^{n-2} y^1 + a_6 x^{n-2} y^0 + \cdots + a_{\binom{n+2}{n}} x^0 y^0$$

Associating the vector of coefficients $\vec{a} := (a_1, a_2, \ldots)$ with a ciphertext and specific variable assignments $\vec{v} := (x^n y^0, x^{n-1} y^1, \ldots, x^0 y^0)$ with a message, we see that $\langle \vec{a}, \vec{v}\rangle = p(x, y)$.

[2] Lewko *et al.* [LOS+10] construct a scheme that is fully (not selectively) secure but which satisfies only a weaker notion of attribute-hiding. It is not discussed whether the scheme satisfies selective attribute-hiding.

To encode logical expressions, we proceed recursively: for the base case, we construct a simple single-variable polynomial to test equality:

$$P_{x \stackrel{?}{=} I}(x) := x - I$$

Then, define the AND and OR operators in terms of logical expressions ϕ encoded as polynomials P_ϕ:

$$P_{\phi_1 \vee \phi_2}(\vec{x}_1, \vec{x}_2) := P_{\phi_1}(\vec{x}_1) \cdot P_{\phi_2}(\vec{x}_2)$$

$$P_{\phi_1 \wedge \phi_2}(\vec{x}_1, \vec{x}_2) := r \cdot P_{\phi_1}(\vec{x}_1) + P_{\phi_2}(\vec{x}_2)$$

for random $r \in \mathbb{Z}_q$. The proofs that $x - I = 0 \iff x = I$ and $x_1 \cdot x_2 = 0 \iff (x_1 = 0) \vee (x_2 = 0)$ are straightforward, and with all but negligible probability over choice of r it holds that $r \cdot x_1 + x_2 = 0 \iff (x_1 = 0) \wedge (x_2 = 0)$. Thus, for all ϕ, $P_\phi = 0$ if and only if ϕ evaluates to TRUE.

To build an appropriate polynomial for an arbitrary logical expression, *e.g.*

$$\phi = \Big((x_1 = I_1) \vee (x_2 = I_2)\Big) \wedge \Big((x_3 = I_3) \vee (x_4 = I_4)\Big)$$

we begin with the polynomials encoding the equality tests $P_{\phi_1} = x_1 - I_1$, $P_{\phi_2} = x_2 - I_2$, $P_{\phi_3} = x_3 - I_3$, and $P_{\phi_4} = x_4 - I_4$. Combining these via the definitions of $P_{\phi_1 \wedge \phi_2}$ and $P_{\phi_1 \vee \phi_2}$ above yields:

$$\begin{aligned} P_{\phi_5} &= P_{\phi_1 \vee \phi_2}(x_1, x_2) = P_{\phi_1}(x_1) \cdot P_{\phi_2}(x_2) \\ P_{\phi_6} &= P_{\phi_3 \vee \phi_4}(x_3, x_4) = P_{\phi_3}(x_3) \cdot P_{\phi_4}(x_4) \\ P_\phi = P_{\phi_5 \wedge \phi_6}\big((x_1, x_2), (x_3, x_4)\big) &= r \cdot P_{\phi_5}\big((x_1, x_2)\big) + P_{\phi_6}\big((x_3, x_4)\big) \\ = r \cdot \Big((x_1 - I_1) \cdot (x_2 - I_2)\Big) &+ \Big((x_3 - I_3) \cdot (x_4 - I_4)\Big) \end{aligned}$$

which can be expanded and ordered lexicographically to generate an appropriate inner-product relation. We associate polynomial coefficients (which encapsulate I_1, I_2, I_3, I_4, r) with ciphertexts; thus, security of this procedure follows directly from the attribute-hiding security of Σ.

□

4.2.2 Achieving strong security for point functions with AIBE

In [OT10] the scheme is presented as an inner-product encryption scheme. A decryption key associated with vector $\vec{v}$ can decrypt a ciphertext associated with vector $\vec{x}$ if and only if $\vec{x} \cdot \vec{v} = 0$. To obtain AIBE, we instantiate this general construction with vectors of dimension 2. To encrypt to identity $\mathsf{id} \in \mathbb{Z}_q$, use ciphertext vector $\vec{x} = (1, \mathsf{id})$; the decryption key for identity id is the one associated with vector $\vec{v} = (\mathsf{id}, -1)$.

In [OT10] the scheme is proven to be adaptively secure and weakly-attribute-hiding. However, the "weak" property of the weakly attribute hiding security definition is not relevant in the case of IBE (in the IBE instantiation, for each $\vec{v}$ there is a *unique* $\vec{x}$ such that $\vec{v} \cdot \vec{x} = 0$). Thus the same security proof for the scheme works in the context of the Sim_1-security definition, with only one additional step needed.

4.2.3 Mathematical background

The scheme presented in [OT10] is based on cyclic groups of large prime order augmented with a *bilinear pairing operation*.[3] Pairing-based cryptography has become increasingly popular in the past decade [DBS04]. Such schemes are typically based on a mapping between two well-chosen cryptographic groups which allows for a problem in one group to be reduced to a different (typically easier) problem in the other group.

Bilinear Pairings

Let $\mathbb{G}$ be a cyclic (additive) group of prime order q with generator G, and let $\mathbb{G}_T$ be a cyclic (multiplicative) group with the same order q. A **bilinear pairing** $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties:

1. **Bilinearity:** $e(sG, tG) = e(G, G)^{st}$ for all $s, t \in \mathbb{Z}_q$.
2. **Non-Degeneracy:** $e(G, G)$ is not the identity element of $\mathbb{G}_T$.
3. **Computability:** $e(sG, tG)$ is computable in polynomial time for all $s, t \in \mathbb{Z}_q$.

The Decisional Linear (DLIN) Assumption

The decisional linear problem in $\mathbb{G}$ is to determine whether $a+b = c$ given $U, V, H, aU, bV, cH \in \mathbb{G}$. More formally, we define the advantage of an adversary $\mathcal{A}$ in deciding the DLIN problem

[3] The canonical examples of bilinear pairings are the Weil and Tate pairings over elliptic curves.

as:

$$\mathsf{AdvDLIN}(\mathcal{A},\mathbb{G},\lambda) := \left| \begin{array}{c} \Pr\left[U,V,H \leftarrow \mathbb{G}; a,b \leftarrow \mathbb{Z}_q : \mathcal{A}(U,V,H,aU,bV,(a+b)H) = 1\right] - \\ \Pr\left[U,V,H,R \leftarrow \mathbb{G}; a,b \leftarrow \mathbb{Z}_q : \mathcal{A}(U,V,H,aU,bV,R) = 1\right] \end{array} \right|$$

The DLIN assumption is that $\mathsf{AdvDLIN}(\mathcal{A},\mathbb{G},\lambda)$ is negligible in λ for every PPT adversary $\mathcal{A}$. It is believed that the Decision Linear Problem is hard even in bilinear groups where Decisional Diffie-Hellman is easy [BBS04].

Dual Pairing Vector Spaces

Let $\mathbb{G}$ be a cyclic group with generator G and bilinear pairing operation $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. We write $\mathbb{G}$ additively and $\mathbb{G}_T$ multiplicatively, so that $e(sG,tG) = e(G,G)^{st}$.

Let $\mathbb{V}$ be the vector space $\mathbb{G}^n$. We define a bilinear map $\boldsymbol{e} : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$ as $\boldsymbol{e}(\boldsymbol{x},\boldsymbol{y}) = \prod_{i=1}^n e(X_i,Y_i)$, where $\boldsymbol{x} = (X_1,\ldots,X_n)$ and $\boldsymbol{y} = (Y_1,\ldots,Y_n)$. The space $\mathbb{V}$ has a canonical basis $\mathbb{A} = (\boldsymbol{a}_1,\ldots,\boldsymbol{a}_n)$, where $\boldsymbol{a}_i = (\underbrace{0,\ldots,0}_{i-1},G,\underbrace{0,\ldots,0}_{n-i})$.

A **dual pairing vector space** is such a vector space $\mathbb{V}$ along with dual orthonormal bases $\mathbb{B}$ and $\mathbb{B}^*$. We define the following DPVS setup algorithm, which chooses the dual bases at random:

$\mathsf{DPVSSetup}(1^\lambda, n)$**:** Choose a cyclic group $\mathbb{G}$ of prime order q, equipped with bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, for which the DLIN problem is hard with security parameter λ. Let G be a generator of $\mathbb{G}$. Define $\mathbb{V} := \mathbb{G}^n$ with canonical basis $\mathbb{A}$ as described above.

Choose $\psi \leftarrow \mathbb{Z}_q^*$ and $X = (x_{ij}) \leftarrow \mathrm{GL}(n,\mathbb{Z}_q)$. Set $g_T := e(G,G)^\psi$, and $Y = (y_{ij}) := \psi(X^T)^{-1}$. For $i \in \{1,\ldots,n\}$, set $\boldsymbol{b}_i := \sum_j x_{ij}\boldsymbol{a}_j$ and $\boldsymbol{b}_i^* := \sum_j y_{ij}\boldsymbol{a}_j$.

Return $(\mathsf{params} := (\mathbb{G},\mathbb{G}_T,e,g_T,n,q), \mathbb{B} := (\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n), \mathbb{B}^* := (\boldsymbol{b}_1^*,\ldots,\boldsymbol{b}_n^*))$.

Let $\mathbb{B} = (\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n)$, and let $a_1,\ldots,a_n \in \mathbb{Z}_q$. Then we define the notation $(a_1,\ldots,a_n)_\mathbb{B} = \sum_{i=1}^n a_i\boldsymbol{b}_i$. In other words, $(a_1,\ldots,a_n)_\mathbb{B}$ is the vector whose coordinates with respect to basis $\mathbb{B}$ are $(a_1,\ldots,a_n)$. When $(\mathbb{B},\mathbb{B}^*)$ are dual orthonormal bases, we have $\boldsymbol{e}\Big((a_1,\ldots,a_n)_\mathbb{B},(c_1,\ldots,c_n)_{\mathbb{B}^*}\Big) = (g_T)^{\vec{a}\cdot\vec{c}}$.

4.2.4 The AIBE construction

For completeness, we describe the fully-secure FE scheme from [OT10]. Let DPVSSetup be the DPVS setup routine described above. Then the scheme is given by the following algorithms. Note that identities are elements of $\mathbb{Z}_q$, and payloads are elements of $\mathbb{G}_T$.

Setup(1^λ)**:** Run $(\mathsf{params}, \mathbb{B}, \mathbb{B}^*) \leftarrow \mathsf{DPVSSetup}(1^\lambda, 8)$. Set $\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_8)$ and $\widehat{\mathbb{B}}^* := (\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*, \boldsymbol{b}_6^*, \boldsymbol{b}_7^*)$. Arbitrarily select a plaintext space $\mathcal{M} \subseteq \mathbb{G}_T$, where $|\mathcal{M}| = \mathsf{poly}(\lambda)$. Output $(\mathsf{pk} := (\mathsf{params}, \mathcal{M}, \widehat{\mathbb{B}}), \mathsf{sk} := \widehat{\mathbb{B}}^*)$.

KeyGen$(\mathsf{sk}, \mathsf{id})$**:** Choose $\sigma, \eta_1, \eta_2 \leftarrow \mathbb{Z}_q$ uniformly at random. Return

$$\boldsymbol{k} := (1, \sigma\,\mathsf{id}, -\sigma, 0, 0, \eta_1, \eta_2, 0)_{\mathbb{B}^*}$$

Enc$(\mathsf{pk}, \mathsf{id}, m \in \mathcal{M})$**:** Choose $\zeta, \omega, \varphi \leftarrow \mathbb{Z}_q$ uniformly at random. Return $(\boldsymbol{c}, c')$ where:

$$\boldsymbol{c} := (\zeta, \omega, \omega\,\mathsf{id}, 0, 0, 0, 0, \varphi)_{\mathbb{B}} \qquad\qquad c' := m(g_T)^\zeta$$

Dec$(\boldsymbol{k}, (\boldsymbol{c}, c'))$**:** Compute $m := c'/\boldsymbol{e}(\boldsymbol{c}, \boldsymbol{k})$. If $m \notin \mathcal{M}$ then output $\perp$; otherwise output m.

The original Okamoto-Takashima scheme [OT10] does not include the restriction on the plaintext space having polynomial size. Instead, their decryption algorithm simply returns m, which is random "gibberish" when a ciphertext is decrypted with an incorrect key. Note that the decrypter cannot compare the key's identity to that of the ciphertext, since the ciphertext hides its associated identity. So to make the output of Dec explicitly match the behavior of the IBE functionality, we must ensure that an error indicator $\perp$ is returned when the identities do not match. It is for this reason that we restrict $\mathcal{M}$ to have polynomial size and perform a consistency check in Dec. Furthermore, we also use this restriction for technical reasons in the security proof (briefly, the simulator must be able to determine the discrete logarithm of the plaintext).

Theorem 4.2.2. *This scheme is a straight-line-Sim$_1$-secure (Definition 3.4.2) AIBE scheme, under the DLIN assumption.*

Proof overview. We start by considering the real-world interaction of the Sim$_1$-security definition, in which the adversary first receives a ciphertext Enc$(\mathsf{pk}, \mathsf{id}^*, m^*)$ and then is

allowed to make queries to $\mathsf{KeyGen}(\mathsf{mk}, \cdot)$. By Remark 4.1.2, we assume that the adversary never makes two queries to KeyGen for the same identity. Then, through a series of indistinguishable hybrids we obtain an interaction in which the ciphertext is distributed independently of id^* and m^*. In fact, the values id^* and m^* are not needed until (and unless) the adversary queries KeyGen on id^*. Thus this final interaction serves as the required simulation — it first gives the "dummy" ciphertext to the adversary. Then, whenever the adversary queries KeyGen on id, the simulator queries $F((\mathsf{id}^*, m^*), \cdot)$ on id (here, F is the IBE functionality). From the output of this query, the simulator can deduce whether $\mathsf{id} = \mathsf{id}^*$, and if so, learn the value of m^*. Then the simulator can appropriately generate a simulated decryption key for identity id.

The sequence of hybrids is very similar to that used in the proof of weak-attribute-hiding from [OT10]. However, in that security definition the adversary is not allowed to request a decryption key for either of the challenge identities. In our security definition, the adversary is free to request a decryption key on id^*. Importantly, the adversary can request at most one such key. Our main technical contribution in the security proof is to show that the argument of [OT10] can withstand simulating at most one such key while still letting the simulated ciphertext be distributed independently of the identity and plaintext.

Proof. We prove security in a sequence of hybrid interactions. We begin with a simulator which has oracle access to the IBE functionality $F((\mathsf{id}^*, m^*), \cdot)$ but also receives id^* and m^* as input. Thus this simulator can perfectly simulate the real-interaction of the Sim_1-security definition. Finally by the last hybrid, the simulator will not require id^* and m^* until these are deduced from an oracle query.

Interaction 0: The simulator runs the Setup procedure of the scheme and generates the ciphertext honestly as follows. It chooses $\zeta, \omega, \varphi \leftarrow \mathbb{Z}_q$ and sets:

$$\boldsymbol{c} := (\zeta, \omega, \omega\,\mathsf{id}^*, 0, 0, 0, 0, \varphi)_{\mathbb{B}}; \qquad c' := m^*(g_T)^{\zeta}$$

When the adversary queries its KeyGen oracle on identity id, the simulator queries its oracle on id. If the output of the oracle is $\perp$, then the simulator marks this identity as an **incorrect** identity; otherwise it marks the identity as the **correct** identity. There is at most one correct identity query. It then generates the corresponding key as follows. It chooses $\sigma, \eta_1, \eta_2 \leftarrow \mathbb{Z}_q$

and sets:

$$\boldsymbol{k}_{\mathsf{id}} := (1, \sigma\, \mathsf{id}, -\sigma, 0, 0, \eta_1, \eta_2, 0)_{\mathbb{B}^*}$$

The view of the adversary is thus identical to the real-world interaction of Sim_1-security definition.

Interaction 1: Same as the previous interaction, except the simulator also chooses random $r_1, r_2 \leftarrow \mathbb{Z}_q$ and generates the ciphertext as:

$$\boldsymbol{c} := (\zeta, \omega, \omega\, \mathsf{id}^*, \underline{r_1}, \underline{r_2}, 0, 0, \varphi)_{\mathbb{B}}; \qquad c' = m^*(g_T)^{\zeta}$$

We call this ciphertext a **semi-functional** ciphertext. This interaction is indistinguishable from the previous by the same reasoning as in [OT10], from the security of their "Problem 1" which follows from the DLIN assumption.

Interaction $(2, h)$*,* for $h \in \{0, \ldots, N\}$, where N is an upper bound on the number of KeyGen queries made by the adversary. Interaction $(2, 0)$ is defined to be the same as Interaction 1. For $h > 0$, Interaction $(2, h)$ is defined to be the same as Interaction $(2, h-1)$ except that when the adversary makes its hth *incorrect* identity query to KeyGen, the simulator generates the corresponding decryption key as follows. It chooses random $w_1, w_2 \leftarrow \mathbb{Z}_q$ and sets:

$$\boldsymbol{k}_{\mathsf{id}} := (1, \sigma\, \mathsf{id}, -\sigma, \underline{w_1}, \underline{w_2}, \eta_1, \eta_2, 0)_{\mathbb{B}^*}$$

We call keys of this form **semi-functional** keys. This interaction is indistinguishable from the previous by the same reasoning as in [OT10], from the security of their "Problem 2" which follows from the DLIN assumption. Since we also apply similar reasoning in a later step, we give more details here:

Define $\vec{x}_{\mathsf{id}} = (1, \mathsf{id})$ and $\vec{v}_{\mathsf{id}} = (\mathsf{id}, -1)$. The proof in [OT10] defines a "Problem 2" game that involves the adversary distinguishing between two interactions (*i.e.*, guessing a random choice bit $\beta \leftarrow \{0, 1\}$). Relevant to the IBE scheme's security proof, they show that an adversary participating in the Problem 2 game can generate normal decryption keys, semi-functional keys for any incorrect identity, as well as the following distributions (for

the semi-functional ciphertext and the semi-functional h-th incorrect-identity key):

when $\beta = 0$ in the game	when $\beta = 1$ in the game
$\boldsymbol{c}_{\mathsf{id}^*} := (\zeta, \quad \omega\,\vec{x}_{\mathsf{id}^*}, \quad \vec{r}, \quad \vec{0}, \quad \varphi)_{\mathbb{B}}$	$\boldsymbol{c}_{\mathsf{id}^*} := (\zeta, \quad \omega\,\vec{x}_{\mathsf{id}^*}, \quad \underline{\rho U \vec{x}_{\mathsf{id}^*}}, \quad \vec{0}, \quad \varphi)_{\mathbb{B}}$
$\boldsymbol{k}_{\mathsf{id}} := (1, \quad \sigma\,\vec{v}_{\mathsf{id}}, \quad \vec{0}, \quad \vec{\eta}, \quad 0)_{\mathbb{B}^*}$	$\boldsymbol{k}_{\mathsf{id}} := (1, \quad \sigma\,\vec{v}_{\mathsf{id}}, \quad \underline{\tau Z \vec{v}_{\mathsf{id}}}, \quad \vec{\eta}, \quad 0)_{\mathbb{B}^*}$

Here, $\vec{r} = (r_1, r_2)$ and $\vec{\eta} = (\eta_1, \eta_2)$ are uniform in $\mathbb{Z}_q^2$; ρ, τ are random in $\mathbb{Z}_q$; and U and Z are random 2×2 matrices subject to the constraint that $U^{-1} = Z^T$. Now, since $\mathsf{id} \neq \mathsf{id}^*$, we have that $\vec{v}_{\mathsf{id}} \cdot \vec{x}_{\mathsf{id}^*} \neq 0$. As such, the vectors $\rho U \vec{x}_{\mathsf{id}^*}$ and $\tau Z \vec{v}_{\mathsf{id}}$ are random subject to their inner product being nonzero. Now, it is with negligible probability that independently random vectors have inner product zero; thus $\boldsymbol{k}_{\mathsf{id}}$ above is distributed statistically close to a true semi-functional key (which has independent randomness instead of $\tau Z \vec{v}_{\mathsf{id}}$).

Thus an adversary participating in the Problem 2 game can induce a view statistically close to that of Interaction $(2, h - 1 + \beta)$. Since it is infeasible to guess β in Problem 2 with nonnegligible advantage, we have that Interactions $(2, h - 1)$ and $(2, h)$ are indistinguishable.

Interaction 3: The same as Interaction $(2, N)$ except that if the adversary ever queries KeyGen on the *correct* identity id^*, then the simulator generates the corresponding key as follows. It chooses random $w_1, w_2 \leftarrow \mathbb{Z}_q$ subject to the constraint that (w_1, w_2) and (r_1, r_2) are orthogonal, where r_1, r_2 are the values from the ciphertext. It sets:

$$\boldsymbol{k}_{\mathsf{id}^*} := (1, \sigma\,\mathsf{id}^*, -\sigma, \underline{w_1}, \underline{w_2}, \eta_1, \eta_2, 0)_{\mathbb{B}^*}$$

The indistinguishability of this step follows from the same reasoning as above. However, when applying the same reduction to Problem 2 as before, we have that $\vec{v}_{\mathsf{id}} \cdot \vec{x}_{\mathsf{id}^*} = 0$. Thus the vectors $\rho U \vec{x}_{\mathsf{id}^*}$ and $\tau Z \vec{v}_{\mathsf{id}}$ are random subject to their inner product being *zero*. Indeed, this is exactly the distribution required by Interaction 3. Importantly, it is possible to generate *semi-functional* keys for *incorrect* identities within the Problem 2 game. It is not clear whether semi-functional keys for the *correct* identity can be generated within the game. Thus, this interaction must be arranged last in the sequence of hybrids, and we can only withstand one key for a *correct* identity.[4]

[4]Similarly, the approach here does not appear to extend to higher-dimensional inner product encryptions,

Let $\vec{r} = (r_1, r_2)$ and $\vec{w} = (w_1, w_2)$. Note that in this interaction, the purported plaintext in $\mathsf{Dec}(\boldsymbol{k}_{\mathsf{id}^*}, (\boldsymbol{c}, c'))$ is computed as $c'/\boldsymbol{e}(\boldsymbol{c}, \boldsymbol{k}_{\mathsf{id}^*}) = c'/(g_T)^{\zeta + \vec{r}\cdot\vec{w}} = m^*/(g_T)^{\vec{r}\cdot\vec{w}}$. Thus we can equivalently say that the value $\vec{w}$ chosen for $\boldsymbol{k}_{\mathsf{id}^*}$ is distributed uniformly subject to the constraint that $\mathsf{Dec}(\boldsymbol{k}_{\mathsf{id}^*}, (\boldsymbol{c}, c')) = m^*$.

Interaction 4: The same as Interaction 3, except the ciphertext is generated as follows. The simulator chooses $\zeta', y_1, y_2 \leftarrow \mathbb{Z}_q$ and then sets:

$$\boldsymbol{c} := (\underline{\zeta'}, \underline{y_1}, \underline{y_2}, r_1, r_2, 0, 0, \varphi)_{\mathbb{B}}; \qquad c' = m^*(g_T)^{\zeta}$$

As in the previous interaction, the values w_1, w_2 in $\boldsymbol{k}_{\mathsf{id}^*}$ are chosen randomly subject to the constraint that $\mathsf{Dec}(\boldsymbol{k}_{\mathsf{id}^*}, (\boldsymbol{c}, c')) = m^*$. We claim that interactions 3 & 4 are in fact distributed identically. To see why, let $S = (s_{ij})$ denote a randomly chosen 2×3 matrix, and consider the following change of basis:

$$\begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_8 \end{pmatrix} = \begin{pmatrix} I_3 & & \\ S & I_2 & \\ & & I_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_8 \end{pmatrix}; \qquad \begin{pmatrix} \boldsymbol{d}_1^* \\ \vdots \\ \boldsymbol{d}_8^* \end{pmatrix} = \begin{pmatrix} I_3 & -S^T & \\ & I_2 & \\ & & I_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1^* \\ \vdots \\ \boldsymbol{b}_8^* \end{pmatrix}$$

Then $\mathbb{D} := (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_8)$ and $\mathbb{D}^* := (\boldsymbol{d}_1^*, \ldots, \boldsymbol{d}_8^*)$ are dual orthonormal bases. $\mathbb{D}$ differs from $\mathbb{B}$ only in coordinates 4 & 5. These coordinates of $\mathbb{B}$ are not included as part of the public key, so from the adversary's point of view both $(\mathbb{D}, \mathbb{D}^*)$ and $(\mathbb{B}, \mathbb{B}^*)$ are equally consistent with the public key.

---

when $\vec{x}$ and $\vec{v}$ have high dimension. In that setting, there can be several distinct choices of $\vec{v}$ such that $\vec{x} \cdot \vec{v} = 0$.

With respect to the basis pair $(\mathbb{D}, \mathbb{D}^*)$, the ciphertext component $\boldsymbol{c}$ can be written as:

$$\begin{aligned}
\boldsymbol{c} &:= (\zeta, \omega, \omega\,\mathsf{id}^*, r_1, r_2, 0, 0, \varphi) \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_8 \end{pmatrix} \\
&= (\zeta, \omega, \omega\,\mathsf{id}^*, r_1, r_2, 0, 0, \varphi) \begin{pmatrix} I_3 & & \\ -S & I_2 & \\ & & I_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_8 \end{pmatrix} \\
&= (\underbrace{\zeta - s_{1,1}r_1 - s_{2,1}r_2}_{\zeta'}, \underbrace{\omega - s_{1,2}r_1 - s_{2,2}r_2}_{y_1}, \underbrace{\omega\,\mathsf{id}^* - s_{1,3}r_1 - s_{2,3}r_2}_{y_2}, r_1, r_2, 0, 0, \varphi) \begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_8 \end{pmatrix}
\end{aligned}$$

Since the entries of S are chosen uniformly at random, each of ζ', y_1, y_2 are uniform in $\mathbb{Z}_q$, as desired.

Similarly, the key corresponding to id can be expressed in the new basis as:

$$\begin{aligned}
\boldsymbol{k}_{\mathsf{id}} &:= (1, \sigma\,\mathsf{id}, -\sigma, w_1, w_2, \eta_1, \eta_2, 0) \begin{pmatrix} \boldsymbol{b}_1^* \\ \vdots \\ \boldsymbol{b}_8^* \end{pmatrix} \\
&= (1, \sigma\,\mathsf{id}, -\sigma, w_1, w_2, \eta_1, \eta_2, 0) \begin{pmatrix} I_3 & S^T & \\ & I_2 & \\ & & I_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_1^* \\ \vdots \\ \boldsymbol{d}_8^* \end{pmatrix} \\
&= (1, \sigma\,\mathsf{id}, -\sigma, \underbrace{w_1 + s_{1,1} + s_{1,2}\sigma\,\mathsf{id} - s_{1,3}\sigma}_{w_1'}, \underbrace{w_2 + s_{2,1} + s_{2,2}\sigma\,\mathsf{id} - s_{2,3}\sigma}_{w_2'}, \eta_1, \eta_2, 0)_{\mathbb{D}^*}
\end{aligned}$$

Now, when $\mathsf{id} \neq \mathsf{id}^*$ (*i.e.,* the simulator is servicing a query for an incorrect identity), w_1, w_2 are chosen uniformly in $\mathbb{Z}_q$ (independent of everything else). Thus w_1', w_2' are uniform as desired. When $\mathsf{id} = \mathsf{id}^*$, (w_1, w_2) are chosen uniformly subject to the constraint that $\mathsf{Dec}(\boldsymbol{k}, (\boldsymbol{c}, c')) = m^*$. Since we are changing only the basis, $\boldsymbol{k}$ and $\boldsymbol{c}$ are exactly as before. The change in basis has only affected the coordinates of $\mathbb{B}^*/\mathbb{D}^*$ involving w_1, w_2; thus these values are still uniform subject to the same constraint, as desired.

This final interaction defines our simulator. It first generates the ciphertext $(\boldsymbol{c}, c')$ according to the distribution in Interaction 4. Note that despite m^* appearing in the description of how c' is generated, the ciphertext is in fact distributed independently of m^* and id^*, as required. Then, when the adversary queries KeyGen on id, the simulator queries its oracle $F((\mathsf{id}^*, m^*), \cdot)$ and generates the resulting key accordingly, depending on whether $\mathsf{id} = \mathsf{id}^*$. In the case that $\mathsf{id} = \mathsf{id}^*$, the simulator must choose w_1, w_2 values (in $\boldsymbol{k}_{\mathsf{id}^*}$) according to a particular constraint. To do so, the simulator must know the discrete log of m^* (the values in the constraint all appear "in the exponent"). For this reason, we require the scheme's plaintext space to have polynomial size. Finally, we note that the simulator uses the adversary in a straight-line, black-box manner. □

Theorem 4.2.3. *This AIBE scheme is key verifiable (Definition 2.2.1).*

Proof overview. Recall that $\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_8)$, and $\widehat{\mathbb{B}} \in \mathsf{pk}$. Knowledge of these basis vectors allows the construction of four "test ciphertexts" that together ensure a secret key exhibits the correct properties when paired with a valid ciphertext.

Proof. Our proof of correctness need only consider honestly-generated ciphertexts, that is, ciphertexts of the form $(\boldsymbol{c}, c')$ where

$$\boldsymbol{c} := (\zeta, \omega, \omega\,\mathsf{id}, 0, 0, 0, 0, \varphi)_{\mathbb{B}} \qquad\qquad c' := m(g_T)^{\zeta}$$

To extract m, the decryption operation computes $c'/\boldsymbol{e}(\boldsymbol{c}, \widetilde{\mathsf{sk}})$. Thus, to verify the validity of $\widetilde{\mathsf{sk}}$ for the identity id, we must show that $\boldsymbol{e}(\boldsymbol{c}, \widetilde{\mathsf{sk}}) = (g_T)^{\zeta}$ with overwhelming probability over the randomness used to generate $\boldsymbol{c}$. We write $\boldsymbol{c} = (c_1, \ldots, c_8)_{\mathbb{B}}$ and $\widetilde{\mathsf{sk}} = (s_1, \ldots, s_8)_{\mathbb{B}^*}$, and recall that $\boldsymbol{e}\Big((c_1, \ldots, c_8)_{\mathbb{B}}, (s_1, \ldots, s_8)_{\mathbb{B}^*}\Big) = (g_T)^{\boldsymbol{c}\cdot\widetilde{\mathsf{sk}}}$. Thus, it suffices to show that the inner product $\boldsymbol{c} \cdot \widetilde{\mathsf{sk}} = \sum_{i=1}^{8} c_i s_i = \zeta$. Substituting an honestly-generated ciphertext for $\boldsymbol{c}$ yields:

$$\zeta s_1 + \omega s_2 + \omega\mathsf{id} s_3 + \varphi s_8 = \zeta$$

for all $\zeta, \omega, \varphi \in \mathbb{Z}_q$.

Thus we need only verify that $\widetilde{\mathsf{sk}}$ has appropriate values for s_1, s_2, s_3, and s_8. Since the public key pk includes the basis vectors $\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3, \boldsymbol{b}_8)$, we can construct ciphertexts to check these values.

We construct a $\mathsf{CheckKey}$ algorithm that can be used to verify decryption keys:

$\mathsf{CheckKey}(\mathsf{pk}, \widetilde{\mathsf{sk}}, \mathsf{id})$**:**

Verify that $\widetilde{\mathsf{sk}} \in \mathbb{V}$. If not, abort and output 0. Then, ensure the following constraints hold:

$$\begin{aligned}
\boldsymbol{e}((1,0,0,0,0,0,0,0)_{\mathbb{B}}, \widetilde{\mathsf{sk}}) &= (g_T)^1 \\
\boldsymbol{e}((0,0,0,0,0,0,0,1)_{\mathbb{B}}, \widetilde{\mathsf{sk}}) &= (g_T)^0 \\
\boldsymbol{e}((0,1,\mathsf{id},0,0,0,0,0)_{\mathbb{B}}, \widetilde{\mathsf{sk}}) &= (g_T)^0 \\
\boldsymbol{e}((0,0,1,0,0,0,0,0)_{\mathbb{B}}, \widetilde{\mathsf{sk}}) &\neq (g_T)^0
\end{aligned}$$

If any of the constraints fail, abort and ouput 0. Otherwise return 1.

□

Chapter 5

Connections to Static Obfuscation

In this chapter, we use mediated obfuscation to connect worst-case and average-case *static* obfuscation. Briefly, an average-case (static) obfuscation of the KeyGen algorithm of a suitable functional encryption scheme Σ yields a worst-case (static) obfuscation for Σ's functionality F. We use this connection to derive new impossibility results for average-case (static) obfuscation.

5.1 Static Obfuscation

Definition 5.1.1 ([BGI+01, HMLS10])**.** $(\mathsf{Obfu}, \mathsf{Eval})$ *is a* static obfuscation *scheme for a class of functions $\mathcal{C}$ if for all $f \in \mathcal{C}$ and all x, we have* $\mathsf{Eval}(\mathsf{Obfu}(f, 1^\lambda), x) = f(x)$ *with overwhelming probability over the coins of* Obfu*. The scheme is* **virtual-black-box (VBB) secure** *if for all PPT $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for all f:*

$$\left| \Pr\left[\mathcal{O}_f \leftarrow \mathsf{Obfu}(f, 1^\lambda); \mathcal{A}(\mathcal{O}_f, 1^\lambda) = 1\right] - \Pr\left[\mathcal{S}^f(1^\lambda) = 1\right] \right| \textit{ is negligible in } \lambda.$$

The probability is over the coins of Obfu*, $\mathcal{A}$, and $\mathcal{S}$. If the security condition holds only for a random choice of $f \leftarrow \mathcal{C}$, then the scheme is* **on-average-VBB-secure.**

If the security condition holds when both $\mathcal{A}$ and $\mathcal{S}$ are given an additional arbitrary input z (which may depend on f), then the scheme is secure **in the presence of auxiliary input.**

In our generic construction for a mediated obfuscation of functions $\mathcal{C}$, the client receives a static ciphertext from the vendor and then essentially uses the mediator for oracle access to the KeyGen algorithm of a functional encryption scheme. Now, if the KeyGen algorithm

admits a *static* obfuscation (in the sense of [BGI+01, HMLS10]), then the client has no use for the mediator. Intuitively, the ciphertext together with an appropriate static obfuscation for KeyGen constitute a static obfuscation for $\mathcal{C}$. In fact, the obfuscation of KeyGen need only be secure on average (*i.e.*, for a random choice of mk).

Theorem 5.1.2. *Let* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a* Sim_1*-secure FE scheme with functionality* F *and define* $\mathcal{C} = \{F(m, \cdot) \mid m\}$*. If there is a static obfuscation scheme for the class*[1] *of functions* $\{\mathsf{KeyGen}(\mathsf{mk}, \cdot) \mid \mathsf{mk}\}$ *that is on-average-VBB-secure in the presence of auxiliary input, then there is a VBB-secure static obfuscation scheme for the class* $\mathcal{C}$*.*

Proof. The static obfuscation for $\mathcal{C}$ is as follows:

- Obfu: Given m (the index of the function $F(m, \cdot)$ to be evaluated) and security parameter λ, generate $(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. Also generate $\mathcal{O}$ as a static obfuscation of $\mathsf{KeyGen}(\mathsf{mk}, \cdot)$. Output $(c, \mathcal{O})$.
- Eval: To evaluate such an obfuscation $(c, \mathcal{O})$ on input x, first evaluate $k \leftarrow \mathcal{O}(x)$, then output the result of $\mathsf{Dec}(k, c)$.

Correctness of this construction follows immediately from the correctness properties of the functional encryption scheme and static obfuscation for KeyGen.

To show VBB security of this construction, let $\mathcal{A}$ be an arbitrary PPT adversary, and consider an interaction in which $\mathcal{A}$ receives input $(c, \mathcal{O}, 1^\lambda)$. View its input $(\mathcal{O}, 1^\lambda)$ as an obfuscation of a randomly chosen member of the class $\{\mathsf{KeyGen}(\mathsf{mk}, \cdot) \mid \mathsf{mk}\}$, and the input c as dependent auxiliary input. By the security of $\mathcal{O}$, there exists a simulator $\mathcal{S}_0$ such that for all m:

$$\Big| \Pr\left[(c, \mathcal{O}) \leftarrow \mathsf{Obfu}(\langle F(m, \cdot)\rangle, 1^\lambda); \mathcal{A}(c, \mathcal{O}, 1^\lambda) = 1\right] - \Pr\left[(\mathsf{pk}, \mathsf{mk}) \leftarrow \mathsf{KeyGen}(1^\lambda); \mathcal{S}_0^{\mathsf{KeyGen}(\mathsf{mk}, \cdot)}(\mathsf{Enc}(\mathsf{pk}, m), 1^\lambda) = 1\right] \Big| \text{ is negligible in } \lambda.$$

Now by the Sim_1 security of the FE scheme, we have that there exists another simulator $\mathcal{S}_1$

[1] Technically, this class contains randomized functions, which are supported in the definitions for static obfuscation of Hofheinz *et al.* [HMLS10] that we use here.

such that for all m:

$$\Big|\Pr\left[(\mathsf{pk},\mathsf{mk}) \leftarrow \mathsf{KeyGen}(1^\lambda); \mathcal{S}_0^{\mathsf{KeyGen}(\mathsf{mk},\cdot)}(\mathsf{Enc}(\mathsf{pk},m),1^\lambda)=1\right] - \Pr\left[\mathcal{S}_1^{F(m,\cdot)}(1^\lambda)=1\right]\Big| \text{ is negligible in } \lambda.$$

By transitivity, $\mathcal{S}_1$ is our desired simulator for $\mathcal{A}$. □

5.2 Implications

Theorem 5.1.2 can be interpreted as a reduction from average-case (static) obfuscation to the standard, worst-case (static) obfuscation. As a corollary, we see that an impossibility result relating to an FE scheme's functionality F can be "lifted" to an impossiblity result for the scheme's KeyGen function, as long as the FE scheme is Sim_1-secure.

In Section 4.2.2 we have given an AIBE scheme that indeed satisfies the Sim_1 security required by Theorem 5.1.2. Thus a suitable static obfuscation of its KeyGen procedure would imply the existence of a static obfuscation of point functions with multi-bit output. Obfuscation of point functions has been thoroughly studied [Can97, LPS04, Wee05, CD08, CKVW10, BC10]. However, all of these results use either very strong computational assumptions, or weak security definitions for obfuscation (or both). In fact, several impossibility results are known for obfuscating point functions, which can be used to derive an impossibility for our KeyGen algorithm.

The above definitions of obfuscation consider only adversaries whose output is a single bit. Wee [Wee05] shows that point-function (static) obfuscation is impossible against adversaries with arbitrary-length output. Our Sim_1 security definition for FE does allow adversaries with arbitrary-length output, and the above proof goes through for arbitrary-length output if the obfuscation of KeyGen allows it. Our AIBE construction in Section 4.2.2 also has a black-box simulator, so when the simulator for the KeyGen-obfuscation is also black-box, the resulting simulation in the proof (for static obfuscations of $\mathcal{C} = \{F(m,\cdot) \mid m\}$) is black-box. Again, Wee [Wee05] showed that it is impossible to obfuscate point-functions with a black-box simulator. Thus:

Corollary 5.2.1. *The* KeyGen *algorithm of our construction (Section 4.2.2) has no static obfuscation that is on-average-VBB-secure and either view-simulating or black-box, in the*

presence of the public key.

Recall that the KeyGen algorithm of an IBE scheme is a signing algorithm in a natural digital signature scheme [BF01],[2] and the auxiliary input we consider (an encryption of (m, id)) can be derived from the verification key of this signature scheme. Thus it is quite natural to consider the problem of statically obfuscating KeyGen algorithms in the presence of such auxiliary input.

Roughly speaking, these results imply that the problem of obfuscating signature schemes (with an on-average security guarantee that is highly natural for such settings) is related to the problem of obfuscating point functions (with a more demanding worst-case security guarantee). Progress in the former will be contingent on progress in the latter.

[2] Observed by Naor but described in a paper by Boneh & Franklin.

Chapter 6

Conclusion & Open Problems

Mediated obfuscation is a theoretical framework for program obfuscation that uses interaction with a semi-trusted third-party to circumvent the general impossibility of static obfuscation.

As demonstrated in Chapter 4, mediated obfuscation for point functions arises naturally from fully attribute-hiding (*i.e.*, anonymous) identity-based encryption, and the FE scheme in [OT10] was shown to admit a relaxed-secure mediated obfuscation with relatively minor modifications. Further study of the MO paradigm will benefit from instantiations based on other fully attribute-hiding schemes (*e.g.* [LOS+10]) and pave the way toward the development of theoretically optimal mediated obfuscation.

Furthermore, the development of *strongly* simulation-secure MO (*i.e.*, schemes in which the mediator learns nothing about the client's input x, and security holds against entirely corrupt mediators) would represent a major advance. Fully-secure MO would enable vendors to outsource obfuscation without delegating total trust to the mediator, and guarantee correctness of any value $f(x)$ returned (to the client) by the mediator.

Finally (and perhaps most significantly), it remains to be seen if the mediated obfuscation model can be extended to randomized functionalities. Such an advance would likely coincide with the development of security definitions for FE that capture randomized functionalities; thus far, no such definitions exist.

Bibliography

[ABV+11] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Fuzzy identity based encryption from lattices. Cryptology ePrint Archive, Report 2011/414, 2011. http://eprint.iacr.org/.

[AI09] Nuttapong Attrapadung and Hideki Imai. Conjunctive broadcast and attribute-based encryption. In *Proceedings of the 3rd International Conference Palo Alto on Pairing-Based Cryptography*, Pairing '09, pages 248–265, Berlin, Heidelberg, 2009. Springer-Verlag.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.

[BC10] Nir Bitansky and Ran Canetti. On strong simulation and composable point obfuscation. In Rabin [Rab10], pages 520–537.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Kilian [Kil01], pages 213–229.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [Kil01], pages 1–18.

[BH08] Dan Boneh and Michael Hamburg. Generalized identity based and broadcast encryption schemes. In Josef Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 455–470. Springer, 2008.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 2007 IEEE Symposium on Security and*

Privacy, SP '07, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.

[BSW11a] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Ishai [Ish11], pages 253–273.

[BSW11b] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: Homomorphic encryption for restricted computations. *IACR Cryptology ePrint Archive*, 2011:311, 2011.

[BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[Can97] Ran Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 455–469. Springer, 1997.

[Can00] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2000.

[CCKM00] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller. One-round secure computation and secure autonomous mobile agents. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 512–523. Springer, 2000.

[CD08] Ran Canetti and Ronny Ramzi Dakdouk. Obfuscating point functions with multibit output. In Smart [Sma08], pages 489–508.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. *IACR Cryptology ePrint Archive*, 2001:55, 2001.

[CKVW10] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 2010.

[DBS04] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols : A survey. Cryptology ePrint Archive, Report 2004/064, 2004. http://eprint.iacr.org/.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 445–464. Springer Berlin / Heidelberg, 2006.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18:186–208, February 1989.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[Had10] Satoshi Hada. Secure obfuscation for encrypted signatures. In Gilbert [Gil10], pages 92–112.

[HMLS10] Dennis Hofheinz, John Malone-Lee, and Martijn Stam. Obfuscation for cryptographic purposes. *J. Cryptology*, 23(1):121–168, 2010.

[HPR12] Robert Hooker, Manoj Prabhakaran, and Mike Rosulek. Functional encryption as mediated obfuscation. 2012.

[HRsV11] Susan Hohenberger, Guy N. Rothblum, abhi shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.

[IKO+11] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 406–425. Springer, 2011.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

[Ish11] Yuval Ishai, editor. *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*. Springer, 2011.

[Kil01] Joe Kilian, editor. *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Smart [Sma08], pages 146–162.

[LOS+10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert [Gil10], pages 62–91.

[LPS04] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 20–39. Springer, 2004.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Rabin [Rab10], pages 191–208. Full version published on Cryptology ePrint Archive, Report 2010/563, http://eprint.iacr.org/2010/563.

[OT11] Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. Cryptology ePrint Archive, Report 2011/543, 2011. http://eprint.iacr.org/.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[Sma08] Nigel P. Smart, editor. *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*. Springer, 2008.

[SW04] Amit Sahai and Brent Waters. Fuzzy identity based encryption. Cryptology ePrint Archive, Report 2004/086, 2004. http://eprint.iacr.org/.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[Wat08] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008. http://eprint.iacr.org/.

[Wee05] Hoeteck Wee. On obfuscating point functions. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 523–532. ACM, 2005.

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 354–363, New York, NY, USA, 2004. ACM.