Functional Encryption: Constructions and Lower Bounds

by

Sergey Gorbunov

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Functional Encryption: Constructions and Lower Bounds

Sergey Gorbunov
Master of Science
Graduate Department of Computer Science
University of Toronto
2012

Functional encryption is an emerging paradigm for public-key encryption that enables fine-grained control of access to encrypted data. Given a secret key for a circuit C and an encryption of an input message x, a user should be able to learn the output $C(x)$, but nothing else about the input x. Moreover, security should hold against collusions amongst "key holders", namely, a collusion of users that hold secret keys for circuits $C_1, \ldots, C_q$ and an encryption of x should be able to learn $C_1(x), \ldots, C_q(x)$, but nothing else about x. In this work, we address the question of constructing functional encryption for all polynomial-size circuits. Our main contributions are as follows:

- We show that a general functional encryption for all circuits for unbounded collusions is impossible, under a weak simulation-based security definition. Furthermore, we show that the size of the ciphertext in a functional encryption scheme must grow with the number of collusions.

- We construct a functional encryption scheme secure against an a-priori bounded polynomial number of collusions for all polynomial-size circuits. Our constructions require only semantically secure public-key encryption schemes and pseudorandom generators computable by small-depth circuits. The constructions are secure under a strong adaptive simulation-based security notion.

Acknowledgements

First and foremost, I would like to thank my advisor Vinod Vaikuntanathan. I thank him for the countless hours of discussions, numerous advices, his inspiration and patience. Our meetings are always full of excitement and fascinating science. I am grateful to be his student and learn from him.

I thank Charles Rackoff for introducing me into the field and inspiring me to study it. Charlie is always happy to discuss any "randomly chosen" topic and explain new concepts to me. I thank him for these unconditional discussions.

I would like to acknowledge and thank my collaborators: Shweta Agrawal, Vinod Vaikuntanathan and Hoeteck Wee. The work in this thesis is mainly a result of my collaboration with them [GVW12], [AGVW12].

I thank my family, parents and grandparents for their support.

Contents

Chapter 1

Introduction

Traditional notions of public-key cryptography [DH76] allow only *all-or-nothing* access to data: users who possess the secret key can recover the entire message from a ciphertext, whereas those who do not know the secret key learn nothing at all. One of the advantages of a typical public-key encryption scheme is that it is non-interactive. That is, a user who holds a public-key typically sends a single message corresponding to a ciphertext. Also, a user who does not have the corresponding secret key should not be able to learn anything about the message. On the other hand, Multi-Party Computation (MPC) [Yao82] protocols allow users to compute and learn arbitrary functions of their joint messages, without revealing anything except for the final answer. This gives a fine-grained access control to the data, at the cost of interaction between the users. A functional encryption scheme combines the benefits of the two worlds. In particular, for certain classes of problems, it provides:

1. fine-grained access control to the data along with a

2. low communication round complexity mechanism.

Boneh, Sahai and Waters [BSW11] recently formalized the notion of functional encryption towards this end, building on and generalizing a number of previous constructs including (anonymous) identity-based encryption (IBE) [Sha84, BF01, Coc01, BW06], fuzzy IBE [SW05], attribute-based encryption (ABE) [GPSW06, LOS+10], and predicate encryption [KSW08, LOS+10]. Informally, a functional encryption scheme for a circuit family $\mathcal{C}$ associates secret keys SK_C with every circuit $C \in \mathcal{C}$, and ciphertexts CT with every input x. The owner of the secret key SK_C and the ciphertext CT should be able to obtain $C(x)$, but learn nothing else about the input message x itself.[1] Moreover, security should hold against collusions amongst "key holders", namely, a collusion of users that hold secret keys $\mathsf{SK}_{C_1}, \ldots, \mathsf{SK}_{C_q}$ and an encryption of x should learn nothing else about x apart from $C_1(x), \ldots, C_q(x)$.

Where does functional encryption come from? From one perspective, functional encryption transparently captures as special cases a number of familiar notions of encryption, such as identity-based encryption (IBE), anonymous IBE, fuzzy IBE, attribute-based encryption and so forth. For example, an identity-based encryption scheme can be seen as a functional

[1] We do not require the circuit C to be secret throughout this work, and in most literature on functional encryption. For the singular exception, see the work of Shi, Shen and Waters [SSW09].

encryption scheme for the following family of circuits parametrized by the identity:

$$C_{\mathsf{id}'}(\mathsf{id}, \mu) = \begin{cases} (\mathsf{id}, \mu) & \text{if } \mathsf{id} = \mathsf{id}' \\ (\mathsf{id}, \perp) & \text{otherwise} \end{cases}$$

In this case, id denotes the public identity since the encryption does not hide it. In contrast, in anonymous IBE (which is supposed to hide the user's identity) a functional encryption scheme would not output id explicitly. Another important special case of functional encryption is *predicate encryption with public index* (which is also called attribute-based encryption by some authors). This corresponds to a circuit family $\mathcal{C}$ parametrized by predicates g and defined as:

$$C_g(\mathsf{ind}, \mu) = \begin{cases} (\mathsf{ind}, \mu) & \text{if } g(\mathsf{ind}) = \mathsf{true} \\ (\mathsf{ind}, \perp) & \text{otherwise} \end{cases}$$

Here, ind is the so-called public index (since the encryption does not hide it), and μ is sometimes referred to as the payload message. In a similar vein, fuzzy IBE schemes correspond to a circuit that detects proximity between two strings. By itself, this "generalization standpoint" underestimates the full potential of functional encryption. In particular, a mechanism for allowing users to compute and learn arbitrary function of a message, without revealing the actual message, can be built from MPC. In addition, a question of minimizing complexity of MPC protocols was intensively studied in the community [FKN94, CCKM00, PR03, DI05]. So what does functional encryption capture and help us solve better than the existing techniques?

We present a practical motivating example, where functional encryption seems to give us the "best" solution. Consider a system gateway that checks incoming emails for spam and viruses[1]. It deletes the emails that do not pass the detection and forwards the rest to the user. We will assume that the code for the detection program is public and everyone can get a copy of it. Now, assuming all the incoming emails are encrypted, how can the gateway perform the check? We summarize some possible solutions below, all of which seem not practical.

- Users can send their secret keys to the gateway, in which case the gateway must be fully trusted.
- Each time an email comes in, the gateway and the user can initiate an MPC protocol. However, this requires the user to be online at that instance and communicate with the gateway.
- Using fully homomorphic encryption [Gen09b], the server can compute an encryption CT' of the result of its detection program over the ciphertext (that is, an encryption of 0 if the email is clean and an encryption of 1 otherwise.) It then sends CT' to the user, who decrypts and returns the result. The drawbacks of this approach are similar to the above MPC solution, except that fewer rounds of communication might be performed.[2]

A functional encryption provides us a mechanism to solve this problem most practically and efficiently. In particular, a user can get the code C for the detection program (which we assume is public and honestly generated) and send a special secret key SK_C to the gateway. Whenever the gateway receives a ciphertext CT of an email, it can run the functional encryption

[1] This example is borrowed from [KSW08].

[2] In fact, FHE gives us the most efficient MPC protocol to date [Gen09b].

decryption algorithm on the SK_C and CT to learn whether or not the email contains spam or viruses. It forwards the email to the users only if it is clean. Note, that this solution requires the user to compute SK_C once and for all, after which step the user no longer needs to be online. Furthermore, if the program code gets updated (with new spam/virus signatures) the user can come online, issue a new secret key to the gateway and go back offline.

Previous Results. In 1984, Shamir introduced the concept of identity-based encryption scheme [Sha85]. The first constructions for IBE were shown only in 2001 by Boneh and Franklin [BF01] and Cocks [Coc01]. Later, constructions with various parameters and security settings were presented in [CHK03, BB04a, BB04b, Wat05, Gen06]. More recently, lattice-based constructions were shown in [GPV08, CHKP10, ABB10]. Attributed-based encryption was first introduced by Sahai and Waters [SW05]. Public index ABE constructions were presented in [GPSW06, BSW07, LOS+10, GJPS08, Wat11]. Additional interesting special cases of functional encryption include hidden vector encryption and inner product encryption, which were constructed in [BW07] and [KSW08, LOS+10, AFV11], respectively. An interesting extension for functional encryption for unbounded length inputs was recently introduced by Waters and constructed for regular languages [Wat12]. Note, that most of the existing constructions satisfy only a public index security setting, described as above. A much fewer constructions satisfy the stronger secret index setting [BF01, BW06, KSW08].

The central and challenging open question in the study of functional encryption is:

Can we build a functional encryption scheme for the class of all poly-size circuits?

As mentioned above, all of the existing constructions are limited to predicate encryption schemes, where the predicate itself is computable by a "low complexity" class and the user learns the payload if and only if the predicate is satisfied. The "state-of-the-art" constructions are shown only for families of Boolean formulas and inner products over fields, both of which are computable in NC1. In particular, a large part of the difficulty in constructing functional encryption schemes lies in the fact that we typically require security against a-priori unbounded collusions, namely, adversaries who obtain secret keys for an unbounded number of circuits $C_1, \ldots, C_q$.

1.1 Our Results

Definitional Relations. We present two flavors of simulation-based security definitions and an indistinguishability-based definition largely based on works of [BSW11, O'N10]. Roughly speaking, simulation-based security definition states that whatever information an adversary is able to learn from a ciphertext and secret keys, a simulator can compute from the output values only. Indistinguishability-based security states that no adversary should be able to distinguish between two input messages that behave identically on all queries – that is, for all C, $C(x_0) = C(x_1)$. In the non-adaptive setting of these definitions the adversary can ask for secret keys only before getting the challenge ciphertext. We distinguish between security for a single input message and multiple messages. We show that security for a single input message is equivalent to security for many messages under our second (and stronger) simulation-based definition in the non-adaptive settings (Theorem 3.3.1). Note, the result of this form is only possible in the non-adaptive setting, since in the adaptive setting we present a positive result in Section 5.3 for a single input message and [BSW11] shows an impossibility result for many messages.

New Lower Bound: Impossibility for Simulation-based Definitions. Our first main result rules out general functional encryption under the weak one message secure, non-adaptive simulation definition (wNA-SIM).

Theorem 1.1.1 (Informal)**.** *There exists a circuit family $\mathcal{C}$ for which there is no* wNA-SIM*-secure function encryption scheme.*

Specifically, assuming the existence of a family of weak pseudo-random function $\mathsf{wPRF}(\cdot,\cdot)$, we show that there does not exist a functional encryption scheme for the family:

$$C_d(x) = \mathsf{wPRF}(x, d), \text{where the input message } x \text{ is the PRF seed}$$

We show that the ciphertext size in a functional encryption scheme realizing this circuit family must grow with the size of the collusion; this yields a contradiction, since the scheme must handle unbounded collusions. In fact, the result is unconditional since any non-trivial functional encryption scheme gives rise to a one-way function and thus pseudo-random functions.

The key observation is as follows. Suppose the adversary requests for q secret keys corresponding to random inputs $C_{d_1}, \ldots, C_{d_q}$ and then requests for an encryption of a random x. Then, the simulated ciphertext together with the q simulated secret keys constitute a description of the values $\mathsf{wPRF}(x, d_1), \ldots, \mathsf{wPRF}(x, d_q)$, which is essentially a sequence of q truly random bits via pseudo-randomness. By a standard information-theoretic argument, this means that the length of the ciphertext plus the secret keys must grow with q. To obtain a lower bound on the ciphertext size, we carefully exploit the fact that the simulator has to generate the secret keys before it sees the output of $\mathsf{wPRF}(x, \cdot)$. Then, the simulator has to generate a small ciphertext that "explains" all these pseudorandom values which is impossible using a compressibility argument. More generally, we show that (1) weak pseudo-random family is "incompressible", and (2) wNA-SIM-secure functional encryption only exists for "compressible" circuit families.

q-Bounded Functional Encryption for Circuits. To overcome the above impossibility result, we consider general functional encryption schemes for all circuits that depend on the number of secret keys that the adversary can obtain (i.e. number of collusions). Therefore, we consider a natural relaxation and initiate a systematic study of functional encryption for *bounded* collusions. We consider a notion of security where the adversary is given secret keys for an *a-priori bounded number of circuits* $C_1, \ldots, C_q$ of her choice (which can be made adaptively). This notion, which we call *q-bounded security* (or security against q collusions), is a natural relaxation of the strong definition above, and could be sufficient in a number of practical use-case scenarios. Our second main result is a construction of q-bounded secure functional encryption schemes for *arbitrary polynomial-size circuit families* under *mild cryptographic assumptions.*

The question of designing IBE schemes with bounded collusions has been considered in a number of works [DKXY02, CHH+07, GLW12]. The functional encryption setting presents us with a significantly richer landscape since (1) a secret key SK_C can be used to obtain (partial) information about many messages, as opposed to IBE where a secret key decrypts only ciphertexts for a single identity, and (2) the partial information is a result of a potentially complex computation on the message itself. Our constructions leverage interesting ideas from the study of (information-theoretic) multi-party computation [BGW88, BMR90, DI05] and randomized encodings [Yao86, IK00, AIK06].

We stress that q-bounded security does not restrict the system from issuing an unbounded number of secret keys. We guarantee security against any adversary that gets hold of at most

q keys. Specifically, our security definition achieves security against multiple "independent" collusions, as long as each collusion has size at most q. Indeed, it is not clear how to achieve such a security notion for general circuits even in the stateful setting where the system is allowed to maintain a counter while issuing secret keys (analogous to the early notion of stateful signatures). We note that our construction does not require maintaining any state.

The main result of this work is the construction of a q-query functional encryption scheme for the class of all polynomial-size circuits. Our construction is based on the existence of semantically secure public key encryption schemes, and pseudorandom generators (PRG) computable in NC1. The former is clearly a necessary assumption, and the latter is a relatively mild assumption which, in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

An important special case of functional encryption that we will be interested in is *predicate encryption with public index* defined as above. For the case of predicate encryption schemes with *public index*, our construction handles arbitrary polynomial-size circuits while relying *solely* on the existence of semantically secure public-key encryption schemes, which is clearly the minimal necessary assumption. In particular, we do not need the "bounded-depth PRG" assumption for this construction.

We will henceforth refer to a functional encryption scheme that supports arbitrary polynomial-size circuits as a *general functional encryption* scheme. Summarizing this discussion, we show:

Theorem 1.1.2 (Main Theorem, Informal)**.** *Let κ be a security parameter. Assuming the existence of semantically secure encryption schemes as well as PRGs computable in NC1, for every $q = q(\kappa)$, there exists a general functional encryption scheme secure against q secret key queries.*

Theorem 1.1.3 (Informal)**.** *Let κ be a security parameter. Assuming the existence of semantically secure encryption schemes, for every $q = q(\kappa)$, there exists a general predicate encryption scheme with public index secure against q secret key queries.*

1.2 Overview of the Thesis

In **Chapter 2**, we describe the preliminaries for our lower bounds and constructions. We also present the syntax and the correctness definition for functional encryption.

In **Chapter 3**, we describe two variants of simulation-based security definitions and an indistinguishability-based definition. We also present a few simple relations between the definitions and informally argue why we choose these definitions for our results.

In **Chapter 4**, we first describe a notion of incompressible circuits. We then show that weak pseudo-random functions are incompressible, and that functional encryption can only exist for compressible functions. This gives us our first main result: the (unconditional) lower bound that rules out a general non-adaptive simulation-secure functional encryption construction for all circuits.

In **Chapter 5**, we describe our second main result: the construction of q-bounded adaptive simulation-secure functional encryption for all poly-size circuits. Along the way, we present a

bootstrapping theorem that transforms an arbitrary functional encryption for NC1 circuits into functional encryption for all poly-size circuits. We present a second bootstrapping theorem that transforms functional encryption for NC1 circuits with special properties into a scheme for all poly-size circuits, assuming fully-homomorphic encryption.

In **Chapter 6**, we summarize and present a few open problems in the area.

Chapter 2

Preliminaries

Notations. Let $\mathcal{D}$ denote a distribution over some finite set S. Then, $x \leftarrow \mathcal{D}$ is used to denote the fact that x is chosen from the distribution $\mathcal{D}$. When we say $x \xleftarrow{\$} S$, we simply mean that x is chosen from the uniform distribution over S. Unless explicitly mentioned, all logarithms are to base 2. p.p.t. stands for a probabilistic polynomial-time algorithm. For $n \in \mathbb{N}$, let $[n]$ denote the set of numbers $1, \ldots, n$. Let κ denote the security parameter. We say a function is negligible in the security parameter if it decreases faster than an inverse of any polynomial. More formally, $f : \mathbb{N} \to \mathbb{R}$ is negligible if:

$$\forall c, \exists \kappa_0, \forall \kappa > \kappa_0, f(\kappa) < \frac{1}{n^c}$$

For simplicity, we often refer to such functions as $\mathrm{negl}(\kappa)$ without defining them explicitly.

2.1 Functional Encryption

Let $\mathcal{X} = \{\mathcal{X}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ denote ensembles where each $\mathcal{X}_\kappa$ and $\mathcal{Y}_\kappa$ is a finite set. Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ denote an ensemble where each $\mathcal{C}_\kappa$ is a finite collection of circuits, and each circuit $C \in \mathcal{C}_\kappa$ takes as input a string $x \in \mathcal{X}_\kappa = \{0,1\}^\kappa$ and outputs $C(x) \in \mathcal{Y}_\kappa$.[1] We are only interested in the families where testing the membership of $C \in \mathcal{C}_\kappa$ can be performed efficiently.

A functional encryption scheme $\mathcal{FE}$ for $\mathcal{C}$ consists of four algorithms $\mathcal{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ defined as follows.

- **Setup** $\mathsf{FE.Setup}(1^\kappa)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys $(\mathsf{MPK}, \mathsf{MSK})$.
- **Key Generation** $\mathsf{FE.Keygen}(\mathsf{MSK}, C)$ is a p.p.t. algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\kappa$ and outputs a corresponding secret key SK_C.
- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, x)$ is a p.p.t. algorithm that takes as input the master public key MPK and an input message $x \in \mathcal{X}_\kappa$ and outputs a ciphertext CT.
- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$ is a deterministic algorithm that takes as input the secret key SK_C and a ciphertext CT and outputs $C(x)$.

[1] Note that all circuits $C \in \mathcal{C}_\kappa$ take inputs of the same length.

Definition 2.1.1 (Correctness)**.** *A functional encryption scheme $\mathcal{FE}$ is correct if for all $C \in \mathcal{C}_\kappa$ and all $x \in \mathcal{X}_\kappa$,*

$$\Pr\left[\begin{array}{l}(\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa);\\ \quad \mathsf{FE.Dec}(\mathsf{FE.Keygen}(\mathsf{MSK},C),\mathsf{FE.Enc}(\mathsf{MPK},x)) \neq C(x)\end{array}\right] = \mathrm{negl}(\kappa)$$

where the probability is taken over the coins of FE.Setup*,* FE.Keygen*, and* FE.Enc*.*

Refer to Chapter 3 for the security definition. When we refer to a functional encryption scheme, we implicitly assume that it is correct.

2.2 Public Key Encryption

A public key encryption scheme $\mathcal{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, over message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$ where $\mathcal{M}_\kappa = \{0,1\}^\kappa$, is a triple of p.p.t. algorithms as follows.

- **Setup.** $\mathsf{PKE.Setup}(1^\kappa)$**:** takes a unary representation of the security parameter and outputs public and private secret keys $(\mathsf{PK},\mathsf{SK})$.
- **Encryption.** $\mathsf{PKE.Enc}_{\mathsf{PK}}(M)$**:** takes the public encryption key PK and a message $M \in \mathcal{M}_\kappa$ and outputs a ciphertext CT.
- **Decryption.** $\mathsf{PKE.Dec}_{\mathsf{SK}}(\mathsf{CT})$**:** takes the secret key SK and a ciphertext CT and outputs a message $M^* \in \mathcal{M}_\kappa$.

Correctness and security against chosen plaintext attacks are defined as follows.

Definition 2.2.1. *A public key encryption scheme $\mathcal{PKE}$ is correct if for all M,*

$$\Pr[(\mathsf{PK},\mathsf{SK}) \leftarrow \mathsf{PKE.Setup}(1^\kappa); \mathsf{PKE.Dec}_{\mathsf{SK}}(\mathsf{PKE.Enc}_{\mathsf{PK}}(M)) \neq M] = \mathrm{negl}(\kappa)\ ,$$

where the probability is over the coins of PKE.Setup*,* PKE.Enc*.*

Definition 2.2.2. *A public key encryption scheme $\mathcal{PKE}$ is (t,ϵ)-IND-CPA secure if for any adversary $\mathcal{A}$ that runs in time t it holds that*

$$\left|\Pr[\mathcal{A}^{\mathsf{PKE.Enc}_{\mathsf{PK}}(\cdot)}(1^\kappa,\mathsf{PK})=1] - \Pr[\mathcal{A}^{\mathsf{PKE.Enc}_{\mathsf{PK}}(0)}(1^\kappa,\mathsf{PK})=1]\right| \leq \epsilon\ ,$$

where the probability is over $(\mathsf{PK},\mathsf{SK}) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$, the coins of PKE.Enc *and the coins of the adversary $\mathcal{A}$.*

When we refer to a public key encryption scheme we implicitly assume that it is correct.

2.3 Shamir's Secret Sharing

We assume familiarity with Shamir's secret-sharing scheme [Sha79] which works as follows: Let $\mathbb{F}$ be a finite field and let $\mathbf{x} = (x_1,\ldots,x_n)$ be a vector of any distinct non-zero elements of $\mathbb{F}$, where $n < |\mathbb{F}|$. Shamir's t-out-of-n secret-sharing scheme works as follows:

- To share a secret $M \in \mathbb{F}$, the sharing algorithm $\mathsf{SS.Share}_{t,n}(M)$ chooses a random univariate polynomial $\mu(x)$ of degree t with constant coefficient M. The n shares are $\mu(x_1), \ldots, \mu(x_n)$.

  Note that any t or fewer shares look uniformly random.

- The reconstruction algorithm $\mathsf{SS.Reconstruct}$ takes as input $t+1$ shares and uses Lagrange interpolation to find a unique degree-t polynomial $\mu(\cdot)$ that passes through the share points. Finally, it computes $\mu(0)$ to recover the secret.

An important property of this scheme is that it permits computation on the shares, a feature used in many multi-party computation protocols starting from [BGW88]. In particular, adding shares gives us $\mu_1(i)+\mu_2(i) = (\mu_1+\mu_2)(i)$ and multiplying shares gives us $\mu_1(i)\mu_2(i) = (\mu_1\mu_2)(i)$ (where $\mu_1\mu_2$ denotes the product of the polynomials). Hence, given enough shares it is possible to reconstruct the resulting polynomial. The main catch is that the degree of the polynomial increases with the number of multiplications, requires more shares to recover the answer post multiplication. In other words, the scheme per se is multiplicatively homomorphic for a bounded number of multiplications (but an arbitrary number of additions).

2.4 Decomposable Randomized Encoding

Let $\mathcal{C}$ be a circuit that takes inputs $k \in \{0,1\}^\ell, x \in \{0,1\}^n$ and outputs $\mathcal{C}(k,x) \in \{0,1\}^m$. Intuitively, a randomizing encoding for $\mathcal{C}(k,x)$ allows a user to learn the value $\mathcal{C}(k,x)$ but nothing else about the inputs k and x. We distinguish between the inputs x and k since in our applications of the randomized encodings one of the users will hold the input x and the second user will hold the input k. Decomposability property assures that for all k, x, the encoding of $\mathcal{C}(\cdot, x)$ can be described as a list of pairs of strings, such that given one of the strings corresponding to a bit k_i for each pair a user can recover $C(k,x)$.

A decomposable randomized encoding scheme $\mathcal{RE}$ consists of two algorithms $(\mathsf{RE.Encode}, \mathsf{RE.Decode})$ satisfying the following properties:

1. **Decomposable Encoding.** $\mathsf{RE.Encode}(1^\kappa, \mathcal{C}, x, k)$**:** A p.p.t. algorithm takes as inputs a security parameter, a description of a circuit $\mathcal{C}$, inputs x and k generates a decomposable randomized encoding:
$$(\tilde{\mathcal{C}}_1(\cdot, x; R), \ldots, \tilde{\mathcal{C}}_\ell(\cdot, x; R))$$
where for $i \in [\ell]$ $\tilde{\mathcal{C}}_i(\cdot, x; R)$ is a pair of strings $\tilde{\mathcal{C}}_i(0, x; R)$ and $\tilde{\mathcal{C}}_i(1, x; R)$. The output of the algorithm is a list of strings $(\tilde{\mathcal{C}}_1(k_1, x; R), \ldots, \tilde{\mathcal{C}}_\ell(k_\ell, x; R))$.

2. **Decoding.** $\mathsf{RE.Decode}((\widetilde{y}_i)_{i=1}^\ell)$**:** On input of an encoding of a circuit $\tilde{y}_i = \tilde{\mathcal{C}}_i(k_i, x; R)$ for some $k = (k_1, \ldots, k_\ell)$ output $\mathcal{C}(k,x)$.

3. **Semantic Security.** We say decomposable randomized encoding $\mathcal{RE}$ is secure if there exists a p.p.t. simulator $\mathsf{RE.Sim}$, such that for every p.p.t. adversary A the outputs of the following two distributions are computationally indistinguishable:

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{RE},A}(1^\kappa)\textbf{:}}$

1: $(\mathcal{C}, k = (k_1, \ldots, k_\ell), x) \leftarrow A(1^\kappa)$
2: $\boxed{(\tilde{\mathcal{C}}_i(\cdot, x; R))_{i=1}^{\ell} \leftarrow \mathsf{RE.Encode}(1^\kappa, \mathcal{C}, x)}$
3: Output $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^{\ell}$

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{RE},\mathsf{RE.Sim}}(1^\kappa)\textbf{:}}$

1: $(\mathcal{C}, k = (k_1, \ldots, k_\ell), x) \leftarrow A(1^\kappa)$
2: $\boxed{(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^{\ell} \leftarrow \mathsf{RE.Sim}(1^\kappa, \mathcal{C}, \mathcal{C}(k, x))}$
3: Output $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^{\ell}$

Note that such a randomized encoding for arbitrary polynomial-size circuits follows from Yao's garbled circuit construction [Yao86, AIK06]. Furthermore, the construction based the garbled circuits gives decomposability property for both inputs k and x.

2.5 Weak Pseudo-Random Functions

Intuitively, a function is weakly pseudo-random if no adversary can distinguish between its output applied on a random secret key and random public inputs, and the output of a randomly chosen function mapping applied on public random inputs.

Definition 2.5.1 (wPRF)**.** *Let* $\mathsf{wPRF} = \{\mathsf{wPRF}_\kappa\}_{\kappa\in\mathbb{N}}$ *denote a family of efficiently computable functions where* $\mathsf{wPRF}_\kappa : \{0,1\}^{n(\kappa)} \times \{0,1\}^{m(\kappa)} \to \{0,1\}^{k(\kappa)}$*, the first argument of which is called the seed to the wPRF and the second argument is the input.*

For every probabilistic polynomial time oracle distinguisher Dist*, consider the following two experiments:*

- $\mathsf{Real}_{\mathrm{Dist}}(1^\kappa)$: *Choose* $x \xleftarrow{\$} \{0,1\}^{n(\kappa)}$ *and run* Dist *with access to a probabilistic oracle* $\mathcal{O}_{real}(x)$ *which, when invoked, chooses a uniformly random* $d \leftarrow \{0,1\}^{m(\kappa)}$ *and returns the pair* $(d, \mathsf{wPRF}_\kappa(x, d))$*. This experiment outputs whatever* Dist *outputs.*
- $\mathsf{Rand}_{\mathrm{Dist}}(1^\kappa)$: *Choose a uniformly random function* $R : \{0,1\}^{m(\kappa)} \to \{0,1\}^{k(\kappa)}$ *and run* Dist *with access to a probabilistic oracle* $\mathcal{O}_{rand}(R)$ *which, when invoked, chooses a uniformly random* $d \leftarrow \{0,1\}^{m(\kappa)}$ *and returns the pair* $(d, R(d))$*. This experiment outputs whatever* Dist *outputs.*

We say wPRF *is a weak pseudo-random function if for all p.p.t. distinguishers* Dist*,*

$$\left|\Pr[\mathsf{Real}_{\mathrm{Dist}}(1^\kappa) = 1] - \Pr[\mathsf{Rand}_{\mathrm{Dist}}(1^\kappa) = 1]\right| = \mathrm{negl}(\kappa)$$

where the probabilities are over the choice of x *and* R*, as well as the coin-tosses of* Dist *and the oracles* $\mathcal{O}_{real}$ *and* $\mathcal{O}_{rand}$*.*

In our impossibility result, we will use a weak pseudo-random function with seed length $n(\kappa) = \kappa$ and output length $k(\kappa) = 1$.

Chapter 3

Security of Functional Encryption

In this chapter, we describe simulation-based and indistinguishability-based definitions, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10]. We describe two flavors of simulation-based definitions: in Chapter 4 we use the first (and weaker) definition for proving the lower bounds; and in Chapter 5 we use the second (stronger) definition for proving positive results. In addition, we show relations between various flavors of the stronger simulation-based and indistinguishability definitions.

3.1 Simulation-based Definitions

First, we present a weak simulation-based definition that is used to prove our lower bounds for unbounded collusions (that is, the number of secret key queries that the adversary can request is independent of the setup.) The definition is referred to as weak because the simulator is allowed to "program" the setup parameters and the non-adaptive secret keys, and the security must hold for a single input message only.

Definition 3.1.1 (wNA-SIM- and wAD-SIM- Security)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a circuit family $\mathcal{C}$. Consider a p.p.t. adversary $A = (A_1, A_2)$ and a stateful p.p.t. simulator S*[1]*. Let $U_x(\cdot)$ denote a universal oracle, such that $U_x(C) = C(x)$. Consider the following two experiments:*

$\mathsf{Exp}^{\mathsf{real}}_{\mathcal{FE},A}(1^\kappa)$**:**	$\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{FE},S}(1^\kappa)$**:**
1: $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$	1: $\mathsf{MPK} \leftarrow S(1^\kappa)$
2: $(x, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$	2: $(x, st) \leftarrow A_1^{S(\cdot)}(\mathsf{MPK})$
3: $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x)$	3: $\mathsf{CT} \leftarrow S^{U_x(\cdot)}(1^\kappa, 1^{\lvert x\rvert})$
4: $\alpha \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK}, \mathsf{CT}, st)$	4: $\alpha \leftarrow A_2^{\mathcal{O}'(\cdot)}(\mathsf{MPK}, \mathsf{CT}, st)$
5: *Output* (x, α)	5: *Output* (x, α)

We distinguish between two cases of the above experiment:

1. The adaptive experiment*, where:*

[1] One can replace a stateful simulator by a regular (stateless) simulator that outputs a state st_s upon each invocation which is carried over to its next invocation.

- *the oracle* $\mathcal{O}(\mathsf{MSK},\cdot) = \mathsf{FE.Keygen}(\mathsf{MSK},\cdot)$ *and*
- *the oracle* $\mathcal{O}'(\cdot)$ *is the simulator, namely* $S^{U_x(\cdot)}(\cdot)$

We call a stateful simulator algorithm S admissible *if, on each input* C*,* S *makes just a single query to its oracle* $U_x(\cdot)$ *on* C *itself.*

The functional encryption scheme $\mathcal{FE}$ *is then said to be* weakly simulation-secure for one message against adaptive adversaries (wAD-SIM-secure, for short) *if there is an* admissible *stateful p.p.t. simulator* S *such that for every p.p.t. adversary* $A = (A_1, A_2)$*, the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{FE},A}(1^\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{FE},S}(1^\kappa)\right\}_{\kappa\in\mathbb{N}}$$

2. The non-adaptive experiment*, where the oracles* $\mathcal{O}(\mathsf{MSK},\cdot)$ *and* $\mathcal{O}'(\cdot)$ *are both the "empty oracles" that return nothing.*

   The functional encryption scheme $\mathcal{FE}$ *is then said to be* weakly simulation-secure for one message against non-adaptive adversaries (wNA-SIM-secure, for short) *if there is an* admissible *stateful p.p.t. simulator* S *such that for every p.p.t. adversary* $A = (A_1, A_2)$*, the two distributions above are computationally indistinguishable.*

Functional Encryption for Bounded Collusions. In Chapter 4 we show that the above definition is impossible to achieve even in the non-adaptive setting. Therefore, we initiate a systematic study of functional encryption for *bounded* collusions. We consider a relaxed notion of security where the adversary is given secret keys for an *a-priori bounded number of circuits* $C_1, \ldots, C_q$ of her choice (which can be made adaptively). In addition, we strengthen the definition by giving the simulator less power by running real setup and non-adaptive key generation algorithms in the ideal world.

Definition 3.1.2 (NA-SIM- and AD-SIM- Security)**.** *Let* $\mathcal{FE}$ *be a functional encryption scheme for a circuit family* $\mathcal{C}$*. For every p.p.t. adversary* $A = (A_1, A_2)$ *and a p.p.t. simulator* $S = (S_1, S_2)$*, consider the following two experiments:*

$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{FE},\ell,A}(1^\kappa)}$**:**

1: $(\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(x_1,\ldots,x_\ell,st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
3: $\boxed{\mathsf{CT}_i \leftarrow \mathsf{FE.Enc}(\mathsf{MPK},x_i)}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK},\mathsf{CT}_1,\ldots,\mathsf{CT}_\ell,st)}$
5: *Output* $(\alpha,x_1,\ldots,x_\ell)$

$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{FE},\ell,S}(1^\kappa)}$**:**

1: $(\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(x_1,\ldots,x_\ell,st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
   ▶ *Let* $(C_1,\ldots,C_q)$ *be* A_1*'s oracle queries*
   ▶ *Let* SK_i *be the oracle reply to* C_i
   ▶ *Let* $\mathcal{V} := \{y_{ij} = C_i(x_j), C_i, \mathsf{SK}_i\}$.
3: $\boxed{(\mathsf{CT}_1,\ldots,\mathsf{CT}_\ell,st') \leftarrow S_1(\mathsf{MPK},\mathcal{V},1^{|x_i|})}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}'(\mathsf{MSK},st',\cdot)}(\mathsf{MPK},\mathsf{CT}_1,\ldots,\mathsf{CT}_\ell,st)}$
5: *Output* $(\alpha,x_1,\ldots,x_\ell)$

We distinguish between two cases of the above experiment:

1. The adaptive case, *where:*

   - *the oracle* $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)$ *and*
   - *the oracle* $\mathcal{O}'(\mathsf{MSK}, st', \cdot)$ *is the second stage of the simulator, namely* $S_2^{U_x(\cdot)}(\mathsf{MSK}, st', \cdot)$*, where* $U_x(C) = C(x)$ *for any* $C \in \mathcal{C}$*.*

   The simulator algorithm S_2 *is stateful in that after each invocation, it updates the state* st' *which is carried over to its next invocation. We call a simulator algorithm* $S = (S_1, S_2)$ admissible *if, on each input* C*,* S_2 *makes just a single query to its oracle* $U_x(\cdot)$ *on* C *itself.*

   The functional encryption scheme $\mathcal{FE}$ *is then said to be* $(q, many)$-simulation-secure for many messages against adaptive adversaries ($(q, many)$-AD-SIM-secure, for short) *if there is an* admissible *p.p.t. simulator* $S = (S_1, S_2)$ *such that for every polynomial function* $\ell = \ell(\kappa)$ *and for every p.p.t. adversary* $A = (A_1, A_2)$ *that makes at most* q *queries, the following two distributions are computationally indistinguishable:*

$$\left\{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{FE},A}(1^\kappa)\right\}_{\kappa\in\mathbb{N}} \overset{c}{\approx} \left\{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{FE},S}(1^\kappa)\right\}_{\kappa\in\mathbb{N}}$$

   In the special case where $\ell(\kappa) = 1$*, we will call the scheme* (q, one)-AD-SIM-secure.

2. The non-adaptive case, *where the oracles* $\mathcal{O}(\mathsf{MSK}, \cdot)$ *and* $\mathcal{O}'(\mathsf{MSK}, st, \cdot)$ *are both the "empty oracles" that return nothing: the functional encryption scheme* $\mathcal{FE}$ *is then said to be* $(q, many)$-query simulation-secure for many messages against non-adaptive adversaries ($(q, many)$-NA-SIM-secure, for short) *if there is a p.p.t. simulator* $S = (S_1, \perp)$ *such that for every polynomial function* $\ell = \ell(\kappa)$ *for every p.p.t. adversary* $A = (A_1, A_2)$ *that makes at most* q *queries, the two distributions above are computationally indistinguishable. In the special case where* $\ell(\kappa) = 1$*, we will call the scheme* (q, one)-NA-SIM-secure.

Intuitively, simulation-based security definition states that any information that the adversary is able to learn from the ciphertext and secret keys, can be obtained by a simulator from the secret keys and the outputs of the circuit alone. A number of remarks on this definition are in order.

1. In the *non-adaptive* setting, the simulator
   (a) is *not* allowed to "program" the public parameters or the pre-ciphertext secret key queries (as opposed to the wNA-SIM and wAD-SIM);
   (b) given the real public parameters, adversary's oracle queries, corresponding real secret keys and circuit output values, is asked to produce a ciphertext indistinguishable from the real ciphertext.
2. In the *adaptive* setting, in addition to the above bullets the second stage simulator
   (c) is given the real MSK and is allowed to "program" the post-ciphertext secret keys.
3. Even if the the adversary does not request any secret keys, he learns the length of x and therefore, the simulator should be given this information to be on even ground with the adversary. This also ensures that the definition properly generalizes (regular) public-key encryption.

4. We remark that our definitions imply (and are stronger than) those presented in the work of Boneh, Sahai and Waters [BSW11], except we only consider a *single* ciphertext and impose an upper bound on the number of secret key queries. More formally, for the adaptive variant we can instantiate [BSW11] simulator (Sim_1, Sim_O, Sim_2) as follows.

   - Sim_1 runs $\mathsf{FE.Setup}$ and sets $pp := \mathsf{MPK}, \sigma := \mathsf{MSK}$.
   - Sim_O runs $\mathsf{FE.Keygen}$ algorithm on MSK and updates σ to include all oracle queries and replies (C_i, SK_i).
   - Sim_2 computes $y_i = U_x(\cdot)$ for all C_i using its oracle. Next, it runs our simulator $S_1(\mathsf{MPK}, \{y_i, C_i, \mathsf{SK}_i\})$ to obtain the ciphertext CT. It invokes A° on the ciphertext, and on any $\mathsf{FE.Keygen}$ call it uses our S_2 to obtain a secret key. Finally, output the same α as A°. The non-adaptive variant follows similarly.

Why do we prove positive results under this definition? First, as mentioned above, our definition is at least as strong as the definition presented in [BSW11]. In addition, we show the following relations between the definitions:

1. *Relations between simulation and indistinguishability:* We show that a *single* message simulation definition implies *single* message indistinguishability definition for both non-adaptive and adaptive worlds.

2. *Relations between* single *and* many *messages (simulation):* We show that a *single* message non-adaptive simulation implies *many* messages non-adaptive simulation definition. However, we cannot hope to achieve the same implication for adaptive world due to the impossibility results presented in [BSW11].

3. *Relations between* single *and* many *messages (indistinguishability):* Finally, we show that a *single* message indistinguishability implies *many* message indistinguishability definition in both the adaptive and non-adaptive worlds.

These definitional implications are summarized in Figure 3.1 and proved below. As a result of these definitional implications, we focus on proving that our constructions are secure under the *single* message adaptive simulation definition ((q, one)-$\mathsf{AD\text{-}SIM}$-security).

3.2 An Indistinguishability-Based Definition

Definition 3.2.1 ($\mathsf{NA\text{-}IND}$- and $\mathsf{AD\text{-}IND}$-Security)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a circuit family $\mathcal{C}$. For every function $\ell = \ell(\kappa)$, every p.p.t. adversary $A = (A_1, A_2)$, consider the following two experiments:*

[1] This proof was not explicitly given in [O'N10], but a similar proof for single message definitions can be easily extended.

[2] General functional encryption for this definition was shown impossible in [BSW11] when adversary makes just 2 $\mathsf{FE.Keygen}$ calls (2-bounded collusion). Since we show a secure construction satisfying $\mathsf{AD\text{-}SIM}_{one}$, this implication follows.

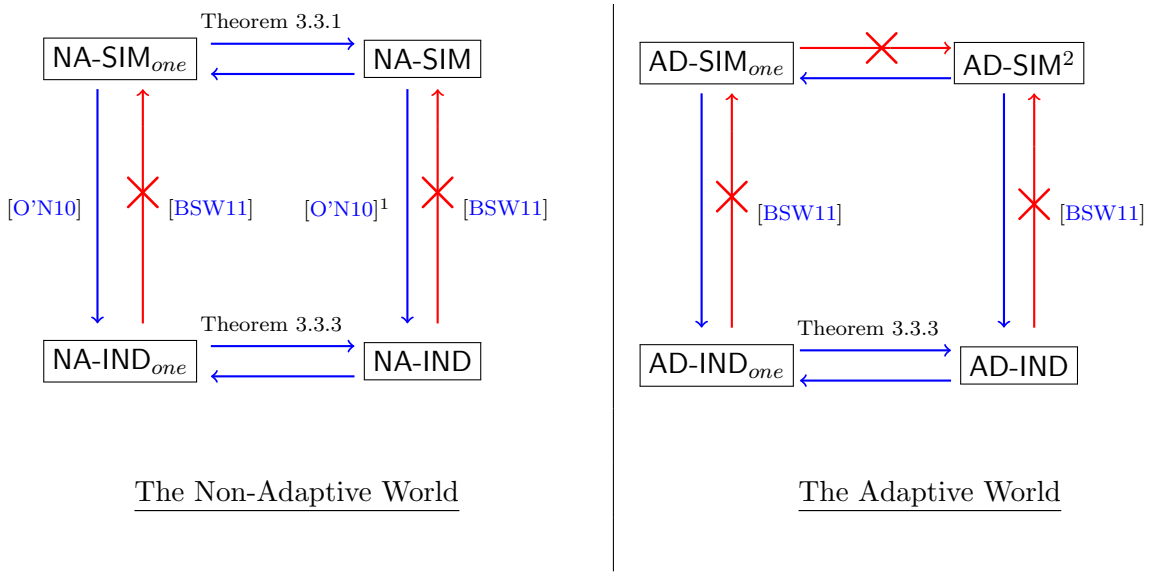

Figure 3.1: Relations between definitions of functional encryption in the non-adaptive and adaptive flavors. Regular blue arrows indicate an implication between the definitions, and a red arrow with a cross on it indicates a separation. The citations for all non-trivial implications and separations are also shown. Note that we omit writing q in the abbreviations above (i.e. AD-SIM:=$(q, many)$-AD-SIM, AD-SIM$_{one}$:=(q, one)-AD-SIM; similarly for the rest of the abbreviations.)

$\underline{\mathsf{Exp}^{(0)}_{\mathcal{FE},A}(1^\kappa)}$**:**

1: $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(\vec{x}_0, \vec{x}_1, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
   ▶ *where* $\vec{x}_0 = (x_0[1], \ldots, x_0[\ell])$
   ▶ *and* $\vec{x}_1 = (x_1[1], \ldots, x_1[\ell])$
3: $\boxed{\mathsf{CT}_i \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x_0[i]) \quad \forall i \in [\ell]}$
4: $b \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK}, \mathsf{CT}_1, \ldots, \mathsf{CT}_\ell, st)$
5: *Output* b

$\underline{\mathsf{Exp}^{(1)}_{\mathcal{FE},A}(1^\kappa)}$**:**

1: $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(\vec{x}_0, \vec{x}_1, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
   ▶ *where* $\vec{x}_0 = (x_0[1], \ldots, x_0[\ell])$
   ▶ *and* $\vec{x}_1 = (x_1[1], \ldots, x_1[\ell])$
3: $\boxed{\mathsf{CT}_i \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x_1[i]) \quad \forall i \in [\ell]}$
4: $b \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK}, \mathsf{CT}_1, \ldots, \mathsf{CT}_\ell, st)$
5: *Output* b

Define an *admissible adversary* $A = (A_1, A_2)$ as one which makes at most q oracle queries and $C(x_0[i]) = C(x_1[i])$ for each query C and every $i \in [\ell]$. *We distinguish between two cases of the above experiment:*

1. The adaptive case, *where the oracle* $\mathcal{O}(\mathsf{MSK}, \cdot) = \mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)$*: the functional encryption scheme* $\mathcal{FE}$ *is said to be* indistinguishable-secure for many messages against adaptive adversaries ($(q, many)$-AD-IND-secure, for short) *if for every polynomial function* $\ell = \ell(\kappa)$ *and every* admissible *p.p.t. admissible adversary* $A = (A_1, A_2)$*, the advantage of* A *defined as below is negligible in the security parameter* κ*:*

$$\mathsf{Adv}_{\mathcal{FE},\ell,A}(\kappa) := \left| \Pr[\mathsf{Exp}^{(0)}_{\mathcal{FE},\ell,A}(1^\kappa) = 1] - \Pr[\mathsf{Exp}^{(1)}_{\mathcal{FE},\ell,A}(1^\kappa) = 1] \right|$$

*where the probability is over the random coins of the algorithms of the scheme $\mathcal{FE}$ and that of A. In the special case where $\ell(\kappa) = 1$, we will call the scheme (q, one)-*AD-IND*-secure.*

2. The non-adaptive case, *where the oracle $\mathcal{O}(\mathsf{MSK}, \cdot)$ is the "empty oracle" that returns nothing: the functional encryption scheme $\mathcal{FE}$ is said to be* indistinguishable-secure for many messages against non-adaptive adversaries (($q, many$)-NA-IND-secure, for short) *if for every polynomial function $\ell = \ell(\kappa)$ and every* admissible *p.p.t. adversary $A = (A_1, A_2)$, the advantage of A defined as above is negligible in the security parameter κ.*

   *In the special case where $\ell(\kappa) = 1$, we will call the scheme (q, one)-*NA-IND*-secure.*

Note that this definition is identical to the definitions presented in [BSW11] and [O'N10], except that they define it for a single message only.

3.3 Relations Between Definitions of Functional Encryption

Theorem 3.3.1. *Let $\mathcal{FE}$ be (q, one)-*NA-SIM*-secure functional encryption scheme for a circuit family $\mathcal{C}$. Then, $\mathcal{FE}$ is also $(q, many)$-*NA-SIM*-secure.*

Proof. Let S_1 be the single message p.p.t. simulator. We construct a p.p.t. simulator S_m. Intuitively, the multiple message simulator will just invoke the single message simulator many times. Then, using the standard hybrid argument we can conclude that it produces output indistinguishable from the real. Let $\ell = \ell(k)$ be arbitrary polynomial function and let $A = (A_1, A_2)$ be arbitrary p.p.t. adversary.

On input $(\mathsf{MPK}, \{y_{ij} = C_i(x_j), C_i, \mathsf{SK}_i\})$ the simulator S_m proceeds as follows: For each j, let

$$V_j := \{y_{ij} = C_i(x_j), C_i, \mathsf{SK}_i\}_{i \in [q]}$$

The simulator computes and outputs the ciphertext[1]:

$$(\mathsf{CT}_1, \ldots, \mathsf{CT}_\ell), \text{ where } \mathsf{CT}_j \leftarrow S_1(\mathsf{MPK}, V_j)$$

Now, let D be the distinguisher between the real and ideal experiments. Then, by the hybrid argument D can distinguish between the experiments where A_2 is given

$$(\mathsf{CT}_1^r, \ldots, \mathsf{CT}_{i-1}^r, \mathsf{CT}_i^s, \ldots, \mathsf{CT}_\ell^s) \text{ vs } (\mathsf{CT}_1^r, \ldots, \mathsf{CT}_i^r, \mathsf{CT}_{i+1}^s, \ldots, \mathsf{CT}_\ell^s)$$

for some i, where CT^r's and CT^s's correspond to the real and simulated ciphertexts, respectively.

We now construct a single message adversary $B = (B_1, B_2)$ and a distinguisher D' as follows:

1. $B_1^{\mathsf{FE.Keygen}(\mathsf{MSK}, \cdot)}(\mathsf{MPK})$ runs A_1 and replies to its oracle queries appropriately to get $(x_1, \ldots, x_\ell, st)$. It outputs

$$(x_i, st' = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_\ell, st, (C_j, \mathsf{SK}_j)_{j \in [q]})$$

[1]Note, that this theorem does not extend to the adaptive definition. In particular, the proof breaks down when even trying to construct the multiple message simulator to "forge" the secret keys SK.

2. $B_2(\mathsf{MPK}, \mathsf{CT}, st')$ first runs the real encryption algorithm on input messages $x_1, \ldots, x_{i-1}$ to obtain $\mathsf{CT}_1^r, \ldots, \mathsf{CT}_{i-1}^r$. Then, for all $j \geq i+1$ it sets

$$V_j := \{y_{ij} = C_i(x_j), C_i, \mathsf{SK}_i\}_{i \in [q]}$$

and runs the single message simulator to get a ciphertext $\mathsf{CT}_j^s \leftarrow S_1(\mathsf{MPK}, V_j)$.

3. Finally, it invokes $A_2(\mathsf{MPK}, \mathsf{CT}_1^r, \ldots, \mathsf{CT}_{i-1}^r, \mathsf{CT}, \mathsf{CT}_{i+1}^s, \ldots, \mathsf{CT}_\ell^s)$ and outputs whatever it outputs.

4. The distinguisher D' is the same as D.

We showed that if there exists a distinguisher for many message simulator, then we can break the security for the single message simulator. This concludes the proof.

□

Theorem 3.3.2. *Let $\mathcal{FE}$ be (q, one)-*AD-SIM*-secure functional encryption scheme for a circuit family $\mathcal{C}$. Then, $\mathcal{FE}$ is also (q, one)-*AD-IND*-secure.*

Proof. Let $A = (A_1, A_2)$ be the admissible adversary such that $\mathsf{Adv}_{\mathcal{FE},\ell,A}$ is non-negligible. We construct adversary $B = (B_1, B_2)$ against (q, one)-AD-SIM-*security.*

- $B_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$: Run the adversary A_1 and reply to its oracle queries using its own oracle to obtain (x_0, x_1, st). Output $(x_b, st^* := (st, x_0, x_1)$, where $b \xleftarrow{\$} \{0,1\}$.
- $B_2^{\mathcal{O}'(\mathsf{MSK}, st', \cdot)}(\mathsf{MPK}, \mathsf{CT}, st)$: Run the adversary $A_2(\mathsf{MPK}, \mathsf{CT}, st)$ replying to its oracle queries using its own oracle to obtain b'. Output $\alpha := (b', st')$.

Now, in the real experiment $b = b'$ with probability $1/2 + \epsilon$ for some noticeable ϵ. In the ideal experiment since the simulator is admissible, it must make the same oracle queries to $U_x(\cdot)$ as B_2 makes, which are the same queries as A_2 makes. Hence, it must be the case that $C_j(x_0) = C_j(x_1)$ for all j. Therefore, information theoretically the simulator gets no information about the bit b and hence cannot produce the corresponding ciphertext with probability better than $1/2$. Hence, we can distinguish between the ideal and real experiment.

□

Theorem 3.3.3. *Let $\mathcal{FE}$ be (q, one)-*AD-IND/NA-IND*-secure functional encryption scheme for a circuit family $\mathcal{C}$. Then, $\mathcal{FE}$ is also $(q, many)$-*AD-IND/NA-IND*-secure, respectively.*

Proof. These proofs follow a standard hybrid argument. □

As a result, we focus on proving only (q, one)-NA-SIM and (q, one)-AD-SIM for our constructions.

Chapter 4

Impossibility Results for Functional Encryption

4.1 Overview of the Results

Our main result rules out general functional encryption under the weak one message secure, non-adaptive simulation definition (wNA-SIM). In particular, this rules out a general unbounded collusions construction for all circuit families for

- 1-message adaptive simulation-based security ((*unbounded*, *one*)-AD-SIM[1]) and
- *many*-message non-adaptive simulation-based security ((*unbounded*, *many*)-NA-SIM.

We compare the impossibility result from [BSW11] with ours in Figure 4.1.

Theorem 4.1.1 (Informal)**.** *There exists a circuit family $\mathcal{C}$ for which there is no* wNA-SIM*-secure function encryption scheme.*

Specifically, assuming the existence of a family of weak pseudo-random function $\mathsf{wPRF}(\cdot,\cdot)$, we show that there does not exist a functional encryption scheme for the family:

$$C_d(x) = \mathsf{wPRF}(x, d), \text{where the input message } x \text{ is the PRF seed}$$

We show that the ciphertext size in a wNA-SIM-secure scheme realizing this circuit family must grow with the size of the collusion; this yields a contradiction, since the scheme must handle unbounded collusions. In fact, the result is unconditional since any non-trivial functional encryption scheme gives rise to a one-way function and thus pseudo-random functions.

The key observation is as follows. Suppose the adversary requests for q secret keys corresponding to random inputs $C_{d_1}, \ldots, C_{d_q}$ and then requests for an encryption of a random x. Then, the simulated ciphertext together with the q simulated secret keys constitute a description of the values $\mathsf{wPRF}(x, d_1), \ldots, \mathsf{wPRF}(x, d_q)$, which is essentially a sequence of q truly random bits via pseudo-randomness. By a standard information-theoretic argument, this means that the length of the ciphertext plus the secret keys must grow with q. To obtain a lower bound on the ciphertext size, we carefully exploit the fact that the simulator has to generate the secret keys before it sees the output of $\mathsf{wPRF}(x, \cdot)$. Then, the simulator has to generate a

[1] "Unbounded" here refers to the schemes that must remain secure for any q polynomial number of collusions, independent of the setup.

	Our impossibility result (Theorem 5.4.1)	Boneh, Sahai and Waters ([BSW11, Theorem 2])
adaptive vs. non-adaptive	non-adaptive	adaptive
one vs. many messages	one message	many messages
one vs. many secret-key queries	many queries	one query
class of circuits	weak PRFs	IBE

Figure 4.1: A comparison between the BSW lower bound and ours for functional encryption. The underlines indicates the stronger result. For example, the first row says that our impossibility result rules out even a non-adaptive notion of security and is thus, stronger than the BSW result that rules out an adaptive notion.

small ciphertext that "explains" all these pseudorandom values which is impossible using a compressibility argument. More generally, we show that (1) weak pseudo-random family is "incompressible", and (2) wNA-SIM-secure functional encryption only exists for "compressible" circuit families. (In particular, the circuit family for all *public-index* predicate encryption is compressible.)

This idea is reminiscent of the obfuscation impossibility result of Goldwasser and Kalai [GK05], although the precise settings are quite different (in particular, functional encryption and program obfuscation seem incomparable, although related, objects).

Implications. The basic idea described above can be extended to a lower bound for even weaker forms of the simulation-based definition, including (a non-adaptive variant of) the definition of Boneh, Sahai and Waters [BSW11]. Here, we mention yet another implication of this idea.

In Chapter 5 we show (q, one)-AD-SIM-secure functional encryption scheme for all circuits, assuming that the adversary can only corrupt an a-priori bounded number of users (and thus, get the corresponding secret keys). One of the shortcomings of their bounded-collusion security notion as well as their construction is that the parameters of the system, and especially the size of the ciphertext depends on the collusion bound q. A natural question is whether their ciphertexts can be made to have size independent of q (or, at the very least, $o(q)$).[1] Indeed, in light of the results of Dodis, Katz, Xu and Yung [DKXY02] and most recently, Goldwasser, Lewko and Wilson [GLW12] in the context of bounded-collusion IBE, one might expect that achieving "short" ciphertexts is actually be possible in general.

Unfortunately, our techniques result in a strong negative answer to this question.

Corollary 4.1.2. *There exists a family of circuits $\mathcal{C}$ such that for every $q = q(\kappa)$, there are no q-collusion resistant $(unbounded, one)$-AD-SIM-secure (resp. -NA-SIM) functional encryption schemes with ciphertexts of size $o(q)$.*

In this section, we present our main lower bound for wNA-SIM-secure functional encryption. We begin with a notion of "incompressible" circuits. Then, we show that (1) weak pseudo-random functions are "incompressible", and (2) wNA-SIM-secure functional encryption only exists for "compressible" circuits. Putting the two together yields our lower bound.

[1] The previous lower bound for $(unbounded, many)$-AD-SIM IBE in [BSW11] (which says that the secret key size must grow with the number of challenge ciphertexts) is not applicable here as our construction considers only a single challenge ciphertext.

4.2 Incompressible Circuits

We first define a family of compressible circuits. Informally, we say that a family of circuits $\{\mathcal{G}_\kappa\}$ is (ℓ, t)-compressible if for a list of uniformly random circuit descriptions $G_1, \ldots, G_\ell \in \mathcal{G}_\kappa$ and a uniformly chosen input x, there is some efficiently computable description of $G_1(x), \ldots, G_\ell(x)$ of size t. Note that if there is no efficiency requirement, then any family is $(\ell, |s|)$-compressible.

Definition 4.2.1 (Incompressible Circuits)**.** *Let $\ell = \ell(\kappa)$ and $t = t(\kappa)$ be functions of the security parameter κ. A family of circuits $\mathcal{G} = \{\mathcal{G}_\kappa\}_{\kappa\in\mathbb{N}}$ is* (ℓ, t)-compressible *if there exist a family of (deterministic) compressor circuits $\{\mathbf{C}_\kappa\}_{\kappa\in\mathbb{N}}$ and a family of decompressor circuits $\{\mathbf{D}_\kappa\}_{\kappa\in\mathbb{N}}$ such that:*

- *(polynomial size) the circuits $\mathbf{C}_\kappa$ and $\mathbf{D}_\kappa$ have size $\mathsf{poly}(\kappa, \ell)$.*
- *(mild compression) for sufficiently large κ, $\left|\mathbf{C}_\kappa(G_1, \ldots, G_\ell, y_1, \ldots, y_\ell)\right| = t$, where $y_i = G_i(x)$.*
- *(correctness) there is a polynomial $p = p(\kappa)$ such that*

$$\Pr[x \xleftarrow{\$} \{0,1\}^\kappa, G_1, \ldots, G_\ell \xleftarrow{\$} \mathcal{G}_\kappa, y_i = G_i(x) : \mathbf{D}_\kappa(G_1, \ldots, G_\ell, \mathbf{C}_\kappa(G_1, \ldots, G_\ell, y_1, \ldots, y_\ell)) = (y_1, \ldots, y_\ell)] \geq 1/p(\kappa)$$

where the probability is taken over the choice of x as well as the circuits $G_1, \ldots, G_\ell$.

The family $\mathcal{G}$ is (ℓ, t)-incompressible *if it is not (ℓ, t)-compressible.*

We now give examples of (in)compressible circuits. First, consider the notion of pre-image samplable family of circuits introduced by O'Neill [O'N10] which requires that given $G_1(x), \ldots, G_\ell(x)$, there is a polynomial-time algorithm that returns an arbitrary x' such that $G_i(x') = G_i(x)$ for all i. In our language, this says that the family $\mathcal{G}$ is $(\ell, |x'|)$-compressible; the compression algorithm simply outputs x'.

Next, consider an arbitrary public-index circuit family parametrized by predicates P and given by:

$$G_P(\mathsf{ind}, \mu) = \begin{cases} (\mathsf{ind}, \mu) & \text{if } P(\mathsf{ind}) = 1 \\ (\mathsf{ind}, \perp) & \text{otherwise} \end{cases}$$

It is easy to see that this circuit family is $(\ell, |(\mathsf{ind}, \mu)|)$-compressible. On input

$$G_{P_1}(\mathsf{ind}, \mu), \ldots, G_{P_\ell}(\mathsf{ind}, \mu)$$

If $P_i(\mathsf{ind}) = 1$ for some i, then the compression algorithm outputs (ind, μ). If $P_i(\mathsf{ind}) = 0$ for all i, then the algorithm outputs $(\mathsf{ind}, \perp)$.

On the other hand, as we show below (see Lemma 4.2.1), any family of (weak) pseudo-random functions is incompressible in a strong sense. More precisely, consider a family of circuits $\mathcal{G} = \{G_{d_i}(\cdot) = \mathsf{wPRF}(\cdot, d_i)\}$ where d_i serves as the input to the pseudo-random function. Informally, the incompressibility is due to the fact that a sequence $(G_{d_1}(x), \ldots, G_{d_\ell}(x)) = (\mathsf{wPRF}(x, d_1), \ldots, \mathsf{wPRF}(x, d_\ell))$ is indistinguishable from a sequence of uniformly random bits, which are clearly incompressible.

Lemma 4.2.1 (weak PRFs are $(\ell, \ell - \kappa)$-incompressible)**.** *Let* $\mathsf{wPRF} = \{\mathsf{wPRF}_\kappa : \{0,1\}^\kappa \times \{0,1\}^{m(\kappa)} \to \{0,1\}\}_{\kappa \in \mathbb{N}}$ *be a family of weak pseudo-random functions, where* $m(\kappa) = \omega(\log \kappa)$*. Define* $G_d(x) = \mathsf{wPRF}(x, d)$*. Consider a family* $\mathcal{G} = \{\mathcal{G}_\kappa\}_{\kappa \in \mathbb{N}}$ *defined as*

$$\mathcal{G}_\kappa = \left\{G_d(\cdot) : |d| = m(\kappa)\right\}$$

Then, $\mathcal{G}$ *is* $(\ell, \ell - \kappa)$*-incompressible.*

Proof. Assume, for the sake of contradiction, that $\mathcal{G}$ is $(\ell, \ell - \kappa)$-compressible. Namely, there are families of compressor and decompressor circuits $(\mathbf{C}, \mathbf{D})$ that satisfy Definition 4.2.1. We show how to construct a distinguisher $\mathrm{Dist}^{\mathcal{O}}$ that distinguishes between the case where $\mathcal{O} = \mathsf{wPRF}(x, \cdot)$ is a pseudo-random oracle that outputs pairs $(d_i, y_i = \mathsf{wPRF}_\kappa(x, d_i))$ where d_i are uniformly random, and the case where $\mathcal{O}$ outputs strings $(d_i, y_i = R(d_i))$ where d_i and R are uniformly random strings and function, respectively. $\mathrm{Dist}^{\mathcal{O}}$ proceeds as follows.

- Choose a sufficiently large κ such that

$$\left|\mathbf{C}_\kappa(G_1, \ldots, G_\ell, y_1, \ldots, y_\ell)\right| = \ell - \kappa$$

- Query the oracle $\mathcal{O}$ to obtain pairs of strings of the form $(d_i \xleftarrow{\$} \{0,1\}^{m(\kappa)}, y_i)$. Define the circuit $G_{d_i}(\cdot) := \mathsf{wPRF}(\cdot, d_i)$.

- Run the compressor $\mathbf{C}_\kappa$ to get a string

$$\gamma \leftarrow \mathbf{C}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, y_1, \ldots, y_\ell)$$

- Outputs 1 if and only if

$$\mathbf{D}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, \gamma) = (y_1, \ldots, y_\ell)$$

We now show that the distinguisher succeeds with non-negligible advantage $1/p(\kappa) - 2^{-\kappa}$ in breaking the weak pseudo-random function family wPRF.

If $\mathcal{O}$ is the pseudo-random oracle, then the samples $\mathrm{Dist}^{\mathcal{O}}$ gets are of the form $(d_i, y_i \leftarrow \mathsf{wPRF}(x, d_i))$. Hence, by correctness of $\mathbf{C}$ and $\mathbf{D}$,

$$\mathbf{D}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, \mathbf{C}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, y_1, \ldots, y_\ell)) = (y_1, \ldots, y_\ell)$$

with probability at least $1/p(\kappa)$. Thus, the distinguisher in this case outputs 1 with probability at least $1/p(\kappa)$ as well.

On the other hand, if $\mathcal{O}$ outputs pairs of strings of the form $(d_i, y_i \leftarrow R(d_i))$ for a randomly chosen function mapping R, we now show that the distinguisher above outputs 1 with probability at most $2^{-\kappa}$. In the analysis below, we assume that $d_1, \ldots, d_\ell$ are distinct, for which we need to pay a price of an additive $\ell^2 \cdot 2^{-m(\kappa)} = \mathrm{negl}(\kappa)$ term in the distinguishing error.

$$
\begin{aligned}
&\Pr[\text{Dist}^{\mathcal{O}} \text{ outputs } 1] \\
&\leq \Pr_{\substack{d_1,\ldots,d_\ell \xleftarrow{\$} \{0,1\}^{m(\kappa)} \\ y_1,\ldots,y_\ell \xleftarrow{\$} \{0,1\}}} \left[\exists \gamma : |\gamma| = \ell - \kappa \text{ and } \mathbf{D}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, \gamma) = (y_1, \ldots, y_\ell)\right] \\
&\leq \sum_{\gamma \in \{0,1\}^{\ell-\kappa}} \Pr_{\substack{d_1,\ldots,d_\ell \xleftarrow{\$} \{0,1\}^{m(\kappa)} \\ y_1,\ldots,y_\ell \xleftarrow{\$} \{0,1\}}} \left[\mathbf{D}_\kappa(G_{d_1}, \ldots, G_{d_\ell}, \gamma) = (y_1, \ldots, y_\ell)\right] \quad \text{(via a union bound)} \\
&= \sum_{\gamma \in \{0,1\}^{\ell-\kappa}} 2^{-\ell} \quad \text{(since } y_1, \ldots, y_\ell \text{ are random and independent of } d_1, \ldots, d_\ell, \gamma) \\
&\leq 2^{\ell-\kappa} \cdot 2^{-\ell} = 2^{-\kappa}
\end{aligned}
$$

This yields the required contradiction to the security of wPRF. □

4.3 The Impossibility Result

We are now ready to state and prove our main theorem.

Theorem 4.3.1. *There exists a family of circuits $\mathcal{G}$ for which there are no* wNA-SIM*-secure functional encryption schemes.*

Proof. We consider two cases.

Case 1: Assume there exists a circuit family of weak pseudo-random functions

$$\mathsf{wPRF} = \{\mathsf{wPRF}_\kappa : \{0,1\}^\kappa \times \{0,1\}^{m(\kappa)} \to \{0,1\}\}_{\kappa \in \mathbb{N}}$$

where $m(\kappa) = \omega(\log \kappa)$. Let $G_d(x) = \mathsf{wPRF}(x, d)$ and consider a family $\mathcal{G} = \{\mathcal{G}_\kappa\}_{\kappa \in \mathbb{N}}$ defined as

$$\mathcal{G}_\kappa = \left\{G_d(\cdot) : |d| = m(\kappa)\right\}$$

Assume, for the sake of contradiction, there exist a wNA-SIM-secure function encryption scheme $\mathcal{FE}$ for $\mathcal{G}$, and let $|\mathsf{CT}|$ denote the length of a ciphertext in the scheme. Let $\ell = \ell(\kappa) = |\mathsf{CT}| + \kappa$.

From Lemma 4.2.1, we know that $\mathcal{G}$ is $(|\mathsf{CT}|+\kappa, |\mathsf{CT}|)$-incompressible. However, Lemma 4.3.2 below tells us that since there is a wNA-SIM secure scheme for $\mathcal{G}$, the family $\mathcal{G}$ is $(|\mathsf{CT}|+\kappa, |\mathsf{CT}|)$-compressible. This gives us the desired contradiction, and therefore, there cannot exist a wNA-SIM-secure functional encryption scheme for $\mathcal{G}$.

Case 2: Assume there does not exist a family of weak pseudo-random functions. Also, for the sake of contradiction, assume there exists a wNA-SIM-secure function encryption scheme for all families of circuits $\mathcal{G}$.

In particular, this means that there is a functional encryption scheme for the empty circuit family (namely, a family $\mathcal{G}$ that does not contain any circuits at all). A wNA-SIM-secure scheme $\mathcal{FE}$ for $\mathcal{G}$ is also a secure public-key encryption scheme. Since public-key encryption implies one-way functions, which in turn imply pseudo-random functions [GGM86, HILL99], we obtain the desired contradiction. □

Lemma 4.3.2 (wNA-SIM $\Rightarrow (\ell, |\mathsf{CT}|)$-compressibility)**.** *Let $\mathcal{G} = \{\mathcal{G}_\kappa\}_{\kappa\in\mathbb{N}}$ be a family of circuits. Suppose there exists a* wNA-SIM*-secure functional encryption scheme for the $\mathcal{G}$. Then, the family $\mathcal{G}$ is $(\ell, |\mathsf{CT}|)$-compressible for any polynomially bounded $\ell = \ell(\kappa)$, where $|\mathsf{CT}|$ denotes size of the encryption of input x.*

Informally, the compression algorithm works as follows: on input $G_1, \ldots, G_\ell$ and $G_1(x), \ldots, G_\ell(x)$, the output is the simulated ciphertext corresponding to an encryption of x. The decompression algorithm then evaluates the decryption algorithm, which is guaranteed to produce $G_1(x), \ldots, G_\ell(x)$.

Proof. Let $(\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ denote the encryption scheme for the family $\mathcal{G}$. Consider the adversary $A = (A_1, A_2)$ in the wNA-SIM security experiment that acts as follows:

- A_1 chooses $G_1, \ldots, G_\ell \xleftarrow{\$} \mathcal{G}$ independently at random and requests for the corresponding secret keys $\mathsf{SK}_1, \ldots, \mathsf{SK}_\ell$. In addition, it chooses $x \xleftarrow{\$} \{0,1\}^{m(\kappa)}$ and outputs x as the challenge message, and $(G_1, \ldots, G_\ell, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$ as the state.
- A_2 outputs α composed of the challenge ciphertext and the state $(G_1, \ldots, G_\ell, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$.

Let S denote the (admissible) stateful p.p.t. simulator guaranteed by wNA-SIM security. We show how to use the simulator to construct a family of (deterministic) compressor and decompressor circuits $\mathbf{C}_\rho$ and $\mathbf{D}_\rho$, indexed by a random string ρ corresponding to the random tape for the simulator:

- The compressor $\mathbf{C}_\rho$, on input $G_1, \ldots, G_\ell$ and $y_1, \ldots, y_\ell$ works as follows: first, compute $\mathsf{MPK} \leftarrow S(1^\kappa;\ \rho)$ and secret keys $\{\mathsf{SK}_i : \mathsf{SK}_i \leftarrow S(G_i;\ \rho)\}_{i\in[\ell]}$. Then compute and output CT as the compressed string, where queries $G_i(x)$ are answered with y_i:

$$\mathsf{CT} \leftarrow S^{U_x(\cdot)}(1^{|m(\kappa)|})$$

- The decompressor $\mathbf{D}_\rho$, on input $G_1, \ldots, G_\ell$ and CT first reconstructs the master public key $\mathsf{MPK} \leftarrow S(1^\kappa;\ \rho)$ and the set of secret keys:

$$\{\mathsf{SK}_i : \mathsf{SK}_i \leftarrow S(G_i;\ \rho)\}_{i\in[\ell]}$$

  Note that $\mathbf{D}_\rho$ has the same randomness ρ hard-wired, and so the secret keys SK_i are exactly the same as those used by $\mathbf{C}_\rho$. Finally, it computes and outputs:

$$\left\{y_i \leftarrow \mathsf{FE.Dec}(\mathsf{SK}_i, \mathsf{CT})\right\}_{i\in[\ell]}$$

Formally, we output $(\mathbf{C}_\rho, \mathbf{D}_\rho)$ for a random ρ, which is a pair of polynomial-size circuits. Clearly, we achieve $(\ell, |\mathsf{CT}|)$-compressibility, since the size of CT is determined by the functional encryption scheme and independent of ℓ. To establish correctness, it suffices to show that:

$$\Pr_{\rho, x, G_1, \ldots, G_\ell}[\mathbf{D}_\rho(G_1, \ldots, G_\ell, \mathbf{C}_\rho(G_1, \ldots, G_\ell, G_1(x), \ldots, G_\ell(x))) = (G_1(x), \ldots, G_\ell(x))] \geq 1 - \mathrm{negl}(\kappa)$$

Here, we will rely on the correctness of the functional encryption scheme as well as wNA-SIM-security. First, consider the distinguisher Dist that given the output $(x, \mathsf{CT}, G_1, \ldots, G_\ell, \mathsf{SK}_1, \ldots, \mathsf{SK}_\ell)$ of the adversary A_2 proceeds as follows:

Output 1 iff for all $i \in [\ell]$, $\mathsf{FE.Dec}(\mathsf{SK}_i, \mathsf{CT}) = G_i(x)$.

Observe that by correctness of the encryption scheme, Dist outputs 1 with probability $1-\mathrm{negl}(\kappa)$ given the output of the adversary A_2 in the wNA-SIM experiment. Therefore, by wNA-SIM-security, Dist also outputs 1 with probability $1 - \mathrm{negl}(\kappa)$ given the output of the (admissible) simulator, where the randomness is taken over the coin tosses ρ of the simulator, along with the random choices of $x, G_1, \ldots, G_\ell$.

This shows that the pair of circuits $(\mathbf{C}_\rho, \mathbf{D}_\rho)$ for a uniformly random ρ is a correct compressor-decompressor pair, establishing the lemma. □

We point out here that our lower bound extends to the setting where the simulator is not required to be admissible, by using a family of (standard) pseudo-random functions.

Finally, the argument here generalizes to showing that functional encryption secure against an a-priori bounded number $q = q(\kappa)$ of collusions is impossible if one insists on small ciphertexts (namely, ciphertexts with much fewer than q bits). This matches the recent result of [GVW12] who construct such functional encryption schemes with ciphertexts of size polynomial in q.

Corollary 4.3.3. *There exists a family of circuits $\mathcal{G}$ such that for every $q = q(\kappa)$, there are no q-collusion resistant* wNA-SIM*-secure functional encryption schemes with ciphertexts of size $o(q)$.*

4.4 Extensions: Impossibility of Weaker Simulation-based Definitions

The idea behind our impossibility result is robust enough to apply to various relaxations of the simulation-based security definition. In this section, we describe a number of such extensions of our result.

Impossibility for the selective and random-input definitions. In the selective model, the adversary is required to commit to the secret key queries $G_1, \ldots, G_q$ as well as the challenge input x before the setup phase. In particular, this means that the adversary will not be able to pick up the circuits or the challenge input depending on the system parameters. Variants of the selective security model are frequently considered in the literature as a relaxations of regular security notions (see, e.g., [BB11, GPSW06, AFV11]). Another relaxation one can consider is one where the adversary is not allowed to choose the circuits or the challenge, but instead, they are chosen uniformly at random.

Our lower bound easily extends to these weaker notions, simply because the adversary we consider in the proof of Lemma 4.3.2 chooses the circuits and the challenge uniformly at random, and independent of the system parameters.

Impossibility for the non-adaptive BSW Definition (the "Rewinding Definition"). The main difference between the definition proposed by [BSW11] and our definition 3.1.1 is that whereas our definition restricts the simulator to be "straight-line", the BSW definition allows the simulator to "rewind" the adversary and interact with it in order to generate the view.

The proof of Lemma 4.3.2 transparently extends to the BSW definition. The adversary A is the same as in the proof. The compressor $\mathbf{C}$ runs the simulator, executing the code of the designated adversary A to compute the response whenever the simulator queries ("rewinds") A. Also, since the simulator is admissible, the queries it makes are exactly the ones that the

compressor knows the answer to. As before, we can make the impossibility result work for even non-admissible simulators by appealing to regular (rather than weak) PRFs.

Impossibility for Secret-key Functional Encryption. In the setting of secret-key functional encryption (first considered by Shi, Shen and Waters [SSW09] in its predicate encryption variant), the encryption algorithm relies on the master secret key to produce the ciphertext for an input x.

All our impossibility results carry over to the setting of secret key functional encryption since in the proof of Lemma 4.3.2, neither the compressor nor the decompressor needs to run the encryption algorithm and generate ciphertexts.

Chapter 5

Functional Encryption for All Polynomial-size Circuits

5.1 Overview of the Construction

We proceed with an overview of our construction of a q-bounded general functional encryption scheme.

Starting point. The starting point of our constructions is the fact, observed by Sahai and Seyalioglu [SS10], that *general* functional encryption schemes resilient against a *single* secret-key query can be readily constructed using the beautiful machinery of Yao's "garbled circuits" [Yao86] (and in fact, more generally, from randomized encodings [IK00, AIK06]).[1] The construction given in [SS10] only achieves "selective, non-adaptive" security, where the adversary must specify the input message x before it sees the public key, and the single key query C before it sees the challenge ciphertext. We show how to overcome these limitations and achieve "full adaptive security" (for a single key query) by using techniques from non-committing encryption [CFGN96], while still relying only on the existence of semantically secure encryption schemes. All of our constructions henceforth also achieve full adaptive security.

Building on this, our construction proceeds in two steps.

Functional Encryption for NC1 Circuits. In the first step, we show how to construct a q-query functional encryption scheme for NC1 circuits starting from any 1-query scheme.

We denote a "degree" of a circuit C as the degree of the polynomial computing C in the variables of x. A degree of a circuit family denotes the maximum degree of a circuit in the family. Let D denote the degree of NC1 family. The complexity of our construction will be polynomial in both D and q, where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext. This step does not require any additional assumption (beyond semantically secure public key encryption).

The high level approach is as follows: we will run N independent copies of the 1-query scheme. To encrypt, we will encrypt the views of some N-party MPC protocol computing some functionality related to C (aka "MPC in the head" [IKOS07]). As the underlying MPC protocol, we will rely on the BGW semi-honest MPC protocol without degree reduction (c.f.

[1] We note that [SS10] is completely insecure for collusions of size two: in particular, given two secret keys SK_{0^ℓ} and SK_{1^ℓ}, an adversary can derive the SK_C for any other C, and moreover, completely recover x.

[DI05, Section 2.2]). We will exploit the fact that this protocol is *completely non-interactive* when used to compute *bounded-degree* functions.

We proceed to sketch the construction. Suppose the encryptor holds input $x = (x_1, \ldots, x_\ell)$, the decryptor holds circuit C, and the goal is for the decryptor to learn $C(x_1, \ldots, x_\ell)$. In addition, we fix t and N to be parameters of the construction.

- The public keys of the system consists of N independent public keys for the 1-query scheme for the same family $C(\cdot)$. The key generation algorithm associates the decryptor with a random subset $\Gamma \subseteq [N]$ of size $Dt + 1$ and generates secret keys for the public keys MPK_i for $i \in \Gamma$. (Note key generation is already a point of departure from previous q-bounded IBE schemes in [DKXY02, CHH+07] where the subset Γ is completely determined by C.)

- To encrypt x, the encryptor first chooses ℓ random polynomials $\mu_1, \ldots, \mu_\ell$ of degree t with constant terms $x_1, \ldots, x_\ell$ respectively. The encryptor computes CT_i to be the encryption of $(\mu_1(i), \ldots, \mu_\ell(i))$ under the i'th public key, and sends $(\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$.

- To decrypt, observe that since $C(\cdot)$ has degree at most D,

$$P(\cdot) := C(\mu_1(\cdot), \ldots, \mu_\ell(\cdot))$$

  is a univariate polynomial of degree at most Dt and whose constant term is $C(x_1, \ldots, x_\ell)$. Now, upon decrypting CT_i for each $i \in \Gamma$, the decryptor recovers $P(i) = C(\mu_1(i), \ldots, \mu_\ell(i))$. It can then recover $P(0) = C(x_1, \ldots, x_\ell)$ via polynomial interpolation.

The key question now is: what happens when q of the decryptors collude? Let $\Gamma_1, \ldots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two of these sets intersect, the adversary obtains two distinct secret keys for the same public key in the underlying one-query FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j. Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j, and learn a share of the encrypted message x.

In particular, to guarantee security, we require that sets $\Gamma_1, \ldots, \Gamma_q$ have *small pairwise intersections* which holds for a uniformly random choice of the sets under an appropriate choice of the parameters t and N. With small pairwise intersections, the adversary is guaranteed to learn at most t shares of the input message x, which together reveal no information about x.

For technical reasons, this is not sufficient to establish security of the basic scheme. The first issue, which already arises for a single key query, is that we need to randomize the polynomial P by adding a random share of 0; this is needed to ensure that the evaluations of P correspond to a random share of $C(x_1, \ldots, x_\ell)$, and indeed, the same issue also arises in the BGW protocol. More generally, we need to rerandomize the polynomial P for each of the q queries $C_1, \ldots, C_q$, in order to ensure that it is consistent with random shares of $C_i(x_1, \ldots, x_\ell)$, for $i = 1, 2, \ldots, q$. This can be done by having the encryptor hard-code additional randomness into the ciphertext. For more details, see Section 5.3.

Predicate encryption with public index. We point out that this construction also gives us for free a predicate encryption scheme with public index for *arbitrary polynomial-size* circuits (with no a-priori bound on the degree). In this setting, it suffices to realize the following family

of circuits parametrized by predicates g:

$$C_g(\mathsf{ind}, \mu) = \begin{cases} (\mathsf{ind}, \mu) & \text{if } g(\mathsf{ind}) = 1 \\ (\mathsf{ind}, 0) & \text{otherwise} \end{cases}$$

We can write C_g as:

$$C_g(\mathsf{ind}, \mu) = (\mathsf{ind}, \mu \cdot g(\mathsf{ind}))$$

Since ind is always part of the output, we can just publish ind "in the clear". Now, observe that for all ind, C_g, we have $C_g(\mathsf{ind}, \mu)$ is a degree one function in the input μ.

To obtain a *predicate encryption scheme with public index*, we observe that the construction above satisfies a more general class of circuits. In particular, if the input to the encryption algorithm is composed of a *public input* (that we do not wish to hide) and a *secret input* (that we do wish to hide), then the construction above only requires that the circuit C has small degree in the bits of the secret input. Informally, this is true because we do not care about hiding the public input, and thus, we will not secret share it in the construction above. Thus, the degree of the polynomial $P(\cdot)$ grows only with the degree of C in its secret inputs. The bottom line is that since predicate encryption schemes with *public index* deal with circuits that have very low degree in the secret input (degree 1, in particular), our construction handles arbitrary predicates.

A Bootstrapping Theorem and Functional Encryption for P. In the second step, we show a "bootstrapping theorem" for functional encryption schemes. In a nutshell, this shows how to generically convert a q-query secure functional encryption scheme for NC1 circuits into one that is q-query secure for arbitrary polynomial-size circuits, assuming in addition the existence of a pseudo-random generator (PRG) that can be computed with circuits of degree $\mathsf{poly}(\kappa)$. Such PRGs can be constructed based on most concrete intractability assumptions such as those related to factoring, discrete logarithms and lattices.

The main tool that enables our bootstrapping theorem is the notion of randomized encodings [Yao86, IK00, AIK06]. Instead of using the FE scheme to compute the (potentially complicated) circuit C, we use it to compute its randomized encoding $\widetilde{C}$ which is typically a much easier circuit to compute. In particular, secret keys are generated for $\widetilde{C}$ and the encryption algorithm for the bounded-degree scheme is used to encrypt the pair $(x; R)$, where R is a uniformly random string. The rough intuition for security is that the randomized encoding $\widetilde{C}(x; R)$ reveals "no more information than" $C(x)$ itself and thus, this transformation does not adversely affect the security of the scheme.

Unfortunately, intuitions can be misleading and so is this one. Note that in the q-query setting, the adversary obtains not just a single randomized encoding, but q of them, namely $\widetilde{C_1}(x; R), \ldots, \widetilde{C_q}(x; R)$. Furthermore, since all these encodings use *the same randomness* R, the regular notion of security of randomized encodings does not apply *as-is*. We solve this issue by hard-coding a large number of random strings (proportional to q) in the ciphertext and using a cover-free set construction, ensuring that the adversary learns q randomized encodings with independently chosen randomness. See Section 5.4 for more details.

Putting this construction together with a randomized encoding scheme for polynomial-size circuits (which follows from Yao's garbled circuits [Yao86, AIK06]) whose complexity is essentially the complexity of computing a PRG, we get our final FE scheme.

As a bonus, we show a completely different way to bootstrap q-query FE schemes for NC1

circuits into a q-query FE scheme for any polynomial-size circuits, using a fully homomorphic encryption scheme [Gen09a, BV11]. See Section 5.5 for more details.

5.2 Background Constructions

5.2.1 Adaptive, Singleton

Consider the following simple circuit family that consists of a single identity circuit $\mathcal{C} = \{C\}$, input space $\mathcal{X} = \{0,1\}$ and $C(x) = x$. We construct a $(1, one)$-AD-SIM-secure functional encryption for this circuit family, starting from any CPA-secure encryption $(\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. (The construction is inspired by techniques used in non-committing encryption [CFGN96, DN00, KO04].)

- **Setup** $\mathsf{BasicFE.Setup}(1^\kappa)$**:** Run $\mathsf{PKE.Setup}$ twice to generate independent master public-key/secret-key pairs

$$(\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{PKE.Setup}(1^\kappa) \qquad \text{for } i = 0, 1$$

  Output the master public/secret key pair

$$\mathsf{MPK} := (\mathsf{PK}_0, \mathsf{PK}_1) \quad \text{and} \quad \mathsf{MSK} := (\mathsf{SK}_0, \mathsf{SK}_1)$$

- **Key Generation** $\mathsf{BasicFE.Keygen}(\mathsf{MSK}, C)$**:** On input the master secret key MSK and a circuit C, pick a random bit $r \xleftarrow{\$} \{0,1\}$ and output the secret key

$$\mathsf{SK} := (r, \mathsf{SK}_r)$$

- **Encryption** $\mathsf{BasicFE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key MPK and an input message $x \in \{0,1\}$: output as ciphertext

$$\mathsf{CT} := (\mathsf{PKE.Enc}(\mathsf{PK}_0, x), \mathsf{PKE.Enc}(\mathsf{PK}_1, x))$$

- **Decryption** $\mathsf{BasicFE.Dec}(\mathsf{SK}, \mathsf{CT})$**:** On input a secret key $\mathsf{SK} = (r, \mathsf{SK}_r)$ and a ciphertext $\mathsf{CT} = (\mathsf{CT}_0, \mathsf{CT}_1)$, output

$$\mathsf{PKE.Dec}_{\mathsf{SK}_r}(\mathsf{CT}_r)$$

Correctness. Correctness is straight-forward.

Security. We prove that the scheme is $(1, one)$-AD-SIM-secure. We define a simulator $\mathsf{BasicFE.Sim}$ that proceeds as follows:

- If the adversary makes a secret key query before seeing the ciphertext, the simulator learns x and can therefore simulate the ciphertext perfectly via normal encryption.
- If the adversary requests for the ciphertext first, then the simulator picks a random bit $\beta \xleftarrow{\$} \{0,1\}$ and outputs as ciphertext:

$$\mathsf{CT} := (\mathsf{PKE.Enc}(\mathsf{PK}_0, \beta), \mathsf{PKE.Enc}(\mathsf{PK}_1, \overline{\beta}))$$

When the adversary then requests for a secret key, the simulator learns $\mathsf{MSK} = (\mathsf{SK}_0, \mathsf{SK}_1)$ and x, and outputs as the secret key:

$$\mathsf{SK} := (\beta \oplus x, \mathsf{SK}_{\beta \oplus x})$$

We establish security via a series of Games.

Game 0. Normal encryption.

Game 1. If the adversary requests for the ciphertext before making a secret key query, then we modify the ciphertext as follows:

$$\mathsf{CT} := (\mathsf{PKE.Enc}(\mathsf{PK}_0, x \oplus r), \mathsf{PKE.Enc}(\mathsf{PK}_1, x \oplus \overline{r}))$$

Game 2. Output of the simulator.

It is easy to see that the outputs of Games 0 and 1 are computationally indistinguishable by CPA security, and that the outputs of Games 1 and 2 are identically distributed.

Extension to larger $\mathcal{X}$. It is easy to see that this construction extends to $\mathcal{X} = \{0,1\}^\lambda$ via λ-wise repetition (that is, λ independent master public keys, etc). We can then prove the security by defining a series of hybrids H_i for $0 \leq i \leq \lambda$, where in hybrid H_i, i bits of the input message and secret keys are simulated and $\lambda - i$ bits and secret keys are real. If there exists a distinguisher between hybrids H_i and H_{i+1} for some i, we can then break the security of the public-key encryption similarly to the case of a single bit input message space.

5.2.2 Adaptive, "Brute Force"

Boneh, et. al [BSW11, Section 4.1] presented a AD-IND-secure scheme for any functionality where the circuit family has polynomial size, starting from any semantically secure public-key encryption scheme. For simplicity, we just write down the construction for a family of two circuits $\mathcal{C} = \{C_0, C_1\}$, which easily extends to any poly-size family. We show that if we replace the underlying encryption scheme with the previous $(1, one)$-AD-SIM-secure FE encryption for singleton circuit space $\mathcal{C}' = \{C^*\}$, then we obtain a $(1, one)$-AD-SIM-secure FE encryption for $\mathcal{C}$.

- **Setup** $\mathsf{BFFE.Setup}(1^\kappa)$**:** Run BasicFE.Setup twice to generate independent master public-key/secret-key pairs

  $$(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{BasicFE.Setup}(1^\kappa) \qquad \text{for } i = 0, 1$$

  Output $(\mathsf{MPK}_0, \mathsf{MPK}_1)$ as the master public key and $(\mathsf{MSK}_0, \mathsf{MSK}_1)$ as the master secret key.

- **Key Generation** $\mathsf{BFFE.Keygen}(\mathsf{MSK}, C_b)$**:** On input the master secret key MSK and a circuit $C_b \in \mathcal{C}$, output as secret key $\mathsf{SK}_b \leftarrow \mathsf{BasicFE.Keygen}(\mathsf{MSK}_b, C^*)$.

- **Encryption** $\mathsf{BFFE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key MPK and an input message $x \in \mathcal{X}$, output as ciphertext

$$\mathsf{CT} := (\mathsf{BasicFE.Enc}(\mathsf{MPK}_0, C_0(x)), \mathsf{BasicFE.Enc}(\mathsf{MPK}_1, C_1(x)))$$

- **Decryption** $\mathsf{BFFE.Dec}(\mathsf{SK}_b, \mathsf{CT})$**:** On input a secret key SK_b and a ciphertext $\mathsf{CT} = (\mathsf{CT}_0, \mathsf{CT}_1)$, output

$$\mathsf{BasicFE.Dec}_{\mathsf{SK}_b}(\mathsf{CT}_b)$$

Correctness. Correctness is straight-forward.

Security. We prove that the scheme is $(1, one)$-AD-SIM-secure. The simulator $\mathsf{BFFE.Sim}$ proceeds as follows:

- If the adversary makes a query C_b before seeing the ciphertext, the simulator learns $C_b(x)$ and then simulates the ciphertext as follows:

$$\mathsf{CT}_b \leftarrow \mathsf{BasicFE.Enc}(\mathsf{MPK}_b, C_b(x)) \quad \text{and} \quad \mathsf{CT}_{1-b} \leftarrow \mathsf{BasicFE.Sim}(\mathsf{MPK}_{1-b}, \emptyset, 1^{|x|})$$

  Output $\mathsf{CT} := (\mathsf{CT}_0, \mathsf{CT}_1)$

- If the adversary requests for the ciphertext first, then the simulator simulates the ciphertext as follows:

$$\mathsf{CT}_i \leftarrow \mathsf{BasicFE.Sim}(\mathsf{MPK}_i, \emptyset, 1^{|x|}), \qquad \text{for } i = 0, 1$$

  Output $\mathsf{CT} := (\mathsf{CT}_0, \mathsf{CT}_1)$. When the adversary then requests for a secret key C_b, the simulator learns $\mathsf{MSK} = (\mathsf{MSK}_0, \mathsf{MSK}_1)$ and $C_b, C_b(x)$ and outputs as secret key

$$\mathsf{SK}_b \leftarrow \mathsf{BasicFE.Sim}(\mathsf{MSK}_b, (C_b(x), C_b), 1^{|x|})$$

We establish security via a series of Games.

Game 0. Normal encryption.

Game 1. Roughly speaking, we will simulate on $\mathsf{MPK}_0, \mathsf{CT}_0$ and follow normal encryption on $\mathsf{MPK}_1, \mathsf{CT}_1$. More precisely, the simulator proceeds as follows:

- If the adversary makes a secret key query C_b before seeing the ciphertext, proceed as follows:
  - if $b = 0$, use the normal encryption for both CT_0 and CT_1.
  - if $b = 1$, follow $\mathsf{BFFE.Sim}$ (that is, generate CT_0 using $\mathsf{BasicFE.Sim}$).
- If the adversary requests for the ciphertext first, then the simulator simulates the ciphertext as follows:

$$\mathsf{CT}_0 \leftarrow \mathsf{BasicFE.Sim}(\mathsf{MPK}_0, \emptyset) \quad \text{and} \quad \mathsf{CT}_1 \leftarrow \mathsf{BasicFE.Enc}(\mathsf{MPK}_1, C_1(x))$$

  Output $\mathsf{CT} := (\mathsf{CT}_0, \mathsf{CT}_1)$. When the adversary then requests for a secret key C_b, the simulator proceeds as follows:

– if $b = 0$, follow $\mathsf{BFFE.Sim}$ (that is, generate SK_0 using $\mathsf{BasicFE.Sim}$);
– if $b = 1$, follow normal encryption (that is, generate SK_1 using $\mathsf{BasicFE.Keygen}$).

Game 2. Output of the simulator.

It is easy to see that the outputs of Games 0 and 1 are computationally indistinguishable by $(1, one)$-$\mathsf{AD\text{-}SIM}$ of the underlying scheme. The same applies to the outputs of Games 1 and 2.

5.2.3 One-Query General Functional Encryption from Randomized Encoding

Sahai and Seyalioglu [SS10] proved $(1, one)$-$\mathsf{NA\text{-}SIM}$; we observe the same "bootstrapping" construction works for $(1, one)$-$\mathsf{AD\text{-}SIM}$. Let $\mathcal{C}$ be an arbitrary family of poly-size circuits. We construct $\mathcal{ONEQFE}$ scheme for $\mathcal{C}$ as follows.

Let $\mathcal{BFFE}$ denote the brute-force construction defined above. In a high-level the idea is this: suppose we wish to construct an FE scheme for a polynomial-size circuit C and input x. Let $U(C, x)$ denote the universal circuit that output $C(x)$. Let $\widetilde{U}(C, x; R)$ denote a randomized encoding of $U(C, x)$ where for every x, R, $\widetilde{U}(\,\cdot\,, x; R)$ has small locality. Then, assuming C has length λ, we can write

$$\widetilde{U}(C, x; R) = (\widetilde{U}_1(C[1], x; R), \ldots, \widetilde{U}_\lambda(C[\lambda], x; R))$$

where $\widetilde{U}_i(\,\cdot\,, x; R)$ depends only on $C[i]$, the ith bit of circuit C. For each i, we can now use $\mathcal{BFFE}$ scheme for a family of two circuits:

$$\widetilde{U}_i := \{\widetilde{U}_i(0,\ \cdot\ ;\ \cdot\ ), \widetilde{U}_i(1,\ \cdot\ ;\ \cdot\ )\}$$

- **Setup** $\mathsf{FE.Setup}(1^\kappa)$**:** Run the brute-force setup algorithm λ times to generate independent master public-key/secret-key pairs

$$(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{BFFE.Setup}(1^\kappa) \qquad \text{for } \widetilde{U}_i \text{ and } i = 1, \ldots, \lambda$$

  Output $(\mathsf{MPK}_i)_{i=1}^\lambda$ as the master public key and $(\mathsf{MSK}_i)_{i=1}^\lambda$ as the master secret key.

- **Key Generation** $\mathsf{FE.Keygen}(\mathsf{MSK}, C)$**:** On input the master secret key MSK and a circuit $C \in \mathcal{C}$, compute

$$\mathsf{SK}_{C,i} \leftarrow \mathsf{BFFE.Keygen}(\mathsf{MSK}_i, \widetilde{U}_i(C[i],\ \cdot\ ;\ \cdot\ )) \qquad \text{for } i = 1, \ldots, \lambda$$

  Output as secret key

$$\mathsf{SK}_C := ((\mathsf{SK}_{C,i})_{i \in [\lambda]})$$

- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key MPK and an input message $x \in \mathcal{X}$, choose R and compute

$$\mathsf{CT}_i \leftarrow \mathsf{BFFE.Enc}(\mathsf{MPK}_i, (x; R)) \qquad \text{for } i = 1, \ldots, \lambda$$

  Output $(\mathsf{CT}_i)_{i=1}^\lambda$ as the ciphertext.

- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** On input a secret key $\mathsf{SK}_C = (\mathsf{SK}_{C,i})_{i \in [\lambda]})$ and a ciphertext $\mathsf{CT} = (\mathsf{CT}_i)_{i=1}^\lambda$, do the following:

1. Compute $\widetilde{y}_i \leftarrow \mathsf{BFFE.Dec}(\mathsf{MSK}_i, \mathsf{CT}_i) = \widetilde{U}_i(C[i], x; R)$ for $i = 1, \ldots, \lambda$;
2. Run the decoder to get $y \leftarrow \mathsf{RE.Decode}(\widetilde{y}_1, \ldots, \widetilde{y}_\lambda)$.

Output y.

Correctness. Correctness follows directly from the correctness of the brute-force FE construction and randomized encodings.

Security. We first prove that $\mathcal{ONEQFE}$ is $(1, one)$-NA-SIM-*secure* (See below on how to modify the proof to show $(1, one)$-AD-SIM-*security*). Recall that the simulator gets as input the following values:

1. The public key: $(\mathsf{MPK}_i)_{i=1}^{\lambda}$;
2. The query C and the corresponding secret key $\mathsf{SK}_C = (\mathsf{SK}_{C,i})_{i=1}^{\lambda}$;
3. The output of C: $C(x)$;

On the very high level, the security of the scheme follows from the fact that by the security of brute-force construction the adversary can only learn $\widetilde{y}_i$ for all i and by the security of the randomized encoding the adversary can only learn $y = C(x)$.

We establish security via a series of Games. Game 0 corresponds to the real experiment and Game $\lambda + 1$ corresponds to the ideal experiment where simulator S produced the ciphertext. The goal of the simulator S is to produce a ciphertext that is indistinguishable from the real ciphertext. Let $\mathsf{BFFE.Sim}$ and $\mathsf{RE.Sim}$ be the brute-force FE and randomized encoding simulators, respectively.

Game 0. Real encryption experiment.

Game i for $i \in \{1, \ldots, \lambda\}$. In Game i, i ciphertexts are encrypted properly using MPK_i and $\lambda - i$ ciphertexts are simulated. Formally, for all $1 \leq j \leq i$, let

$$\mathsf{CT}_i \leftarrow \mathsf{BFFE.Enc}(\mathsf{MPK}_i, (x; R))$$

For all $i < j \leq \lambda$, let

$$\mathsf{CT}_i \leftarrow \mathsf{BFFE.Sim}(\mathsf{MPK}_i, (\widetilde{U}_i(C[i], x; R), \widetilde{U}_i(C[i], \cdot\,; \cdot), \mathsf{SK}_{C,i}))$$

Output the ciphertext

$$\mathsf{CT} := (\mathsf{CT}_1, \ldots, \mathsf{CT}_\lambda)$$

Game $\lambda + 1$. Same as Game λ, except the randomized encoding is now produced by the $\mathsf{RE.Sim}$. Formally, the simulator S does the following.

1. Let
$$(\widetilde{U}_i(C[i], x; R))_{i=1}^{\lambda} \leftarrow \mathsf{RE.Sim}(1^\kappa, U, U(C, x)))$$
2. For all $i \in [\lambda]$, let
$$\mathsf{CT}_i \leftarrow \mathsf{BFFE.Sim}(\mathsf{MPK}_i, (\widetilde{U}_i(C[i], x; R), \widetilde{U}_i(C[i], \cdot\,; \cdot), \mathsf{SK}_{C,i}))$$

3. Output the ciphertext

$$\mathsf{CT} := (\mathsf{CT}_1, \ldots, \mathsf{CT}_\lambda)$$

Claim 5.2.0.1. *The outputs of Game* 0 *and Game* λ *are computationally indistinguishable.*

Proof. The only different between Games 0 and λ is that in the later the ciphertext produced by the simulator. If there is a distinguisher between the Games, then by we can distinguish between Games i and $i+1$ for some i, hence compromise the security of the underlying $\mathcal{BFFE}$ construction. □

Claim 5.2.0.2. *The outputs of Game* λ *and Game* $\lambda+1$ *are computationally indistinguishable.*

Proof. This claim follows directly from the security of the randomized encoding simulator. □

Therefore, we can conclude that the real experiment is indistinguishable from the ideal experiment.

We now sketch how to modify the above proof to show that $\mathcal{ONEQFE}$ is $(1, one)$-AD-SIM-*secure*. Construct the simulator $S = (S_1, S_2)$ as follows. The simulator S_1 is the same as in the non-adaptive case, except it passes the simulated decomposable randomized encoding $\widetilde{U}(C, x; R)$ as a part of the state to S_2. Now, assume the oracle query C comes after the challenge ciphertext (the other case is trivial). We invoke the single brute-force simulator $\mathsf{BFFE.Sim}$ many times for all MSK_i. For every oracle queries $\widetilde{U}_i(C[i], \cdot\,; \cdot)$ made by $\mathsf{BFFE.Sim}$ reply with $\widetilde{y}_i \leftarrow \widetilde{U}_i(C[i], x; R)$. Finally, output $(\mathsf{SK}_{C,i})_{i\in[\lambda]}$ as the secret key to the adversary.

5.3 A Construction for NC1 circuits

In this section, we construct a functional encryption scheme for all NC1 circuits secure against q secret-key queries, starting from one that is secure against *a single secret-key query*. Our construction will rely on any semantically secure public-key encryption scheme.

The Class of Circuits. We construct q-bounded FE scheme for a circuit family $\mathcal{C} := \mathrm{NC1}$. In particular, we consider polynomial representation of circuits C in the family. The input message space $\mathcal{X} = \mathbb{F}^\ell$ is an ℓ-tuple of field elements, and for every circuit $C \in \mathcal{C}$, $C(\cdot)$ is an ℓ-variate polynomial over $\mathbb{F}$ of total degree at most D. The complexity of our construction will be polynomial in both D and q, where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext.

5.3.1 Our Construction

Let $\mathcal{C} := \mathrm{NC1}$ be a circuit family with circuits of degree $D = D(\kappa)$ in its input, and let $q = q(\kappa)$ be a bound on the number of secret key queries. Our scheme is associated with additional parameters $S = S(\kappa)$, $N = N(\kappa)$, $t = t(\kappa)$ and $v = v(\kappa)$ (for an instantiation of the parameters, see Section 5.3.2).

We start by defining a new family $\mathcal{G}$ as follows:

$$G_{C,\Delta}(x, Z_1, \ldots, Z_S) := C(x) + \sum_{i\in\Delta} Z_i \tag{5.1}$$

where $\Delta \subseteq [S]$ and $Z_1, \ldots, Z_S \in \mathbb{F}$.

Let $(\mathsf{OneQFE.Setup}, \mathsf{OneQFE.Keygen}, \mathsf{OneQFE.Enc}, \mathsf{OneQFE.Dec})$ be a functional encryption scheme for $\mathcal{G}$ secure against a *single* secret key query. Our q-query secure encryption scheme $\mathcal{BDFE} = (\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ for $\mathcal{C}$ works as follows:

- **Setup** $\mathsf{BdFE.Setup}(1^\kappa)$**:** Run the one-query setup algorithm N times to generate independent master public-key/secret-key pairs

$$(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{OneQFE.Setup}(1^\kappa) \qquad \text{for } i = 1, \ldots, N$$

  Output $(\mathsf{MPK}_i)_{i=1}^N$ as the master public key and $(\mathsf{MSK}_i)_{i=1}^N$ as the master secret key.

- **Key Generation** $\mathsf{BdFE.Keygen}(\mathsf{MSK}, C)$**:** On input the master secret key MSK and a circuit $C \in \mathcal{C}$,

  1. Choose a uniformly random set $\Gamma \subseteq [N]$ of size $tD + 1$;
  2. Choose a uniformly random set $\Delta \subseteq [S]$ of size v;
  3. Generate the secret keys

$$\mathsf{SK}_{C,\Delta,i} \leftarrow \mathsf{OneQFE.Keygen}(\mathsf{MSK}_i, G_{C,\Delta}) \qquad \text{for every } i \in \Gamma$$

  Output as secret key $\mathsf{SK}_C := (\Gamma, \Delta, (\mathsf{SK}_{C,\Delta,i})_{i\in\Gamma})$.

- **Encryption** $\mathsf{BdFE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key $\mathsf{MPK} = (\mathsf{MPK}_i)_{i=1}^N$ and an input message $x = (x_1, \ldots, x_\ell) \in \mathcal{X}$:

  1. For $i = 1, 2, \ldots, \ell$, pick a random degree t polynomial $\mu_i(\cdot)$ whose constant term is x_i.
  2. For $i = 1, 2, \ldots, S$, pick a random degree Dt polynomial $\zeta_i(\cdot)$ whose constant term is 0.
  3. Run the one-query encryption algorithm $\mathsf{OneQFE.Enc}$ N times to produce ciphertexts

$$\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Enc}\big(\mathsf{MPK}_i, (\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i))\big) \qquad \text{for } i = 1, \ldots, N$$

  Output $(\mathsf{CT}_i)_{i=1}^N$ as the ciphertext.

- **Decryption** $\mathsf{BdFE.Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** On input a secret key $\mathsf{SK}_C = (\Gamma, \Delta, (\mathsf{SK}_{C,\Delta,i})_{i\in\Gamma})$ and a ciphertext $\mathsf{CT} = (\mathsf{CT}_i)_{i=1}^N$, do the following:

  1. Compute a degree Dt polynomial $\eta(\cdot)$ such that $\eta(i) = \mathsf{OneQFE.Dec}(\mathsf{SK}_{C,\Delta,i}, \mathsf{CT}_i)$ for all $i \in \Gamma$.
  2. Output $\eta(0)$.

Correctness

We show that the scheme above is correct. By correctness of the underlying single-query FE, we have that for all $i \in \Gamma$,

$$\begin{aligned} \eta(i) &= G_{C,\Delta}(\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i)) \\ &= C(\mu_1(i), \ldots, \mu_\ell(i)) + \sum_{a\in\Delta} \zeta_a(i) \end{aligned}$$

Since $|\Gamma| \geq Dt + 1$, this means that η is equal to the degree Dt polynomial

$$\eta(\cdot) = C(\mu_1(\cdot), \ldots, \mu_\ell(\cdot)) + \sum_{a \in \Delta} \zeta_a(\cdot)$$

Hence, $\eta(0) = C(x_1, \ldots, x_\ell) = C(x)$.

5.3.2 Setting the Parameters

We show how to set the parameters $S = S(\kappa)$, $N = N(\kappa)$ and $t = t(\kappa)$. These parameters govern the choice of the sets Γ and Δ during the key generation algorithm, and are required to satisfy the following two conditions:

Small Pairwise Intersections. Let $\Gamma_1, \ldots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two of these sets intersect, the adversary obtains two distinct secret keys for the underlying one-query secure FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j. Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j. In particular, for every such j, the adversary potentially learns a share of the encrypted input message x.

Thus, to guarantee security, we require that the union of the pairwise intersections of $\Gamma_1, \ldots, \Gamma_q$ is small. In particular, we require that $\left| \bigcup_{i \neq j} (\Gamma_i \cap \Gamma_j) \right| \leq t$. This ensures that the adversary learns at most t shares of the input message x, which together reveal no information about x.

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(t/q^2)}$) as long as $q^2 \cdot (Dt/N)^2 \cdot N \leq t/10$. In other words, we will set $t(\kappa) = \Theta(q^2\kappa)$ and $N(\kappa) = \Theta(D^2 q^2 t)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$. For details, we refer an interested reader to Section 5.6.1.

Cover-Freeness. Let $\Delta_1, \ldots, \Delta_q \subseteq [S]$ be the (uniformly random) sets of size v chosen for each of the q secret key queries of the adversary. The security proof relies on the condition that the polynomials $\sum_{a \in \Delta_j} \zeta_a(\cdot)$ are uniformly random and independent which is true if the collection of sets $\Delta_1, \ldots, \Delta_q$ is cover-free. That is, for every $i \in [q]$: $\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j \right) \neq \phi$.

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(q^2 v^2/S)}$) as long as $q^2 v^2 / S \leq v/100$. In other words, we will set $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$. For details, we refer an interested reader to Section 5.6.2.

We remark that in our construction, multiple secret key queries for the same $C \in \mathcal{C}$ result in different secret keys SK_C, essentially because of the different random choices of the sets Δ and Γ. Using a pseudorandom function (applied to C), it is possible to ensure that multiple secret key queries for the same C result in the same answer.

5.3.3 Proof of Security

Theorem 5.3.1. *Let $\mathcal{ONEQFE}$ be a $(1, one)$-$\mathsf{AD\text{-}SIM}$-secure (resp. $(1, one)$-$\mathsf{NA\text{-}SIM}$-secure) functional encryption scheme for any family of poly-size circuits. Then, for any circuit family $\mathcal{C}$ computable in NC1 the $\mathcal{BDFE}$ scheme described above is (q, one)-$\mathsf{AD\text{-}SIM}$-secure (resp. (q, one)-$\mathsf{NA\text{-}SIM}$-secure).*

We prove that the construction $\mathcal{BDFE}$ given in Section 5.3 is (q, one)-$\mathsf{AD\text{-}SIM}$-*secure* if we start out with a $(1, one)$-$\mathsf{AD\text{-}SIM}$-secure scheme. This subsumes the non-adaptive variant of the proof. By Theorem 3.3.1, this implies that $\mathcal{BDFE}$ is $(q, many)$-$\mathsf{NA\text{-}SIM}$-*secure*. However, it is only one message adaptive secure ((q, one)-$\mathsf{AD\text{-}SIM}$) (see Figure 3.1 for relations.)

We establish security by first defining the simulator and then arguing that its output is indistinguishable via a series of Games. For readability, we adopt the following convention: we use i to index over values in $[N]$, and we use j to index over the queries.

Overview. Suppose the adversary receives the challenge ciphertext after seeing $q^* \leq q$ queries. The simulator has to simulate the ciphertext and answer the remaining secret key queries. We may assume it already knows all of $\Gamma_1, \ldots, \Gamma_q, \Delta_1, \ldots, \Delta_q$. This is because:

- for $j \leq q^*$, the simulator gets Γ_j, Δ_j from SK_j;
- for $j > q^*$, the simulator gets to program Γ_j, Δ_j and could pick all these quantities in advance.

We first describe our strategy for simulating the ciphertext $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$ and the secret keys. Let $\mathcal{I}$ denote

$$\bigcup_{j \neq j'} (\Gamma_j \cap \Gamma_{j'})$$

We will consider two cases:

- $i \in \mathcal{I}$: Here, we may issue more than one secret key corresponding to $(\mathsf{MPK}_i, \mathsf{MSK}_i)$; therefore, we can no longer rely on the security of the underlying one-query FE scheme. Instead, we rely on the statistical security of the underlying MPC protocol and the fact that $|\mathcal{I}| \leq t$. Specifically, we can simulate CT_i and the secret keys honestly.
- $i \notin \mathcal{I}$: Here, we issue at most one secret key corresponding to $(\mathsf{MPK}_i, \mathsf{MSK}_i)$; this is because at most one of the sets $\Gamma_1, \ldots, \Gamma_q$ contains i. Suppose $i \in \Gamma_j$. We may now appeal to the security of the underlying one-query FE scheme. Specifically, we simulate CT_i computationally using the simulator for the underlying one-query FE scheme. If $j \leq q^*$, then we do not need to program secret keys at all. If $j > q^*$, upon receiving query C_j, we program the corresponding keys $\mathsf{SK}_{C_j, \Delta_j, i}$ using the one-query simulator.

We formally define the simulator $\mathsf{BdFE.Sim}$ as follows:

Simulating the ciphertext after query q^*. Here, the simulator knows $\Gamma_1, \ldots, \Gamma_q, \Delta_1, \ldots, \Delta_q$; the queries $C_1, \ldots, C_{q^*}$, the outputs $C_1(x), \ldots, C_{q^*}(x)$, and the secret keys $\mathsf{SK}_1, \ldots, \mathsf{SK}_{q^*}$.

1. Uniformly and independently sample ℓ random degree t polynomials $\mu_1, \ldots, \mu_\ell$ whose constant terms are all 0.

2. We sample the polynomials $\zeta_1, \ldots, \zeta_S$ as follows: let $\Delta_0 := \emptyset$. For $j = 1, 2, \ldots, q$:
   (a) by the cover-free property, fix some $a^* \in \Delta_j \setminus (\Delta_0 \cup \cdots \cup \Delta_{j-1})$;
   (b) for all $a \in (\Delta_j \setminus (\Delta_0 \cup \cdots \cup \Delta_{j-1})) \setminus \{a^*\}$, set ζ_a to be a uniformly random degree Dt polynomial whose constant term is 0;
   (c) if $j \leq q^*$, pick a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$; if $j > q^*$, pick random values for $\eta_j(i)$ for all $i \in \mathcal{I}$;
   (d) the evaluation of ζ_{a^*} on the points in $\mathcal{I}$ is defined by the relation:

$$\eta_j(\cdot) = C_j(\mu_1(\cdot), \ldots, \mu_\ell(\cdot)) + \sum_{a \in \Delta_j} \zeta_a(\cdot)$$

   Finally, for all $a \notin (\Delta_1 \cup \cdots \cup \Delta_q)$, set ζ_a to be a uniformly random degree Dt polynomial whose constant term is 0.

3. For each $i \in \mathcal{I}$, run the one-query encryption algorithm $\mathsf{OneQFE.Enc}$ to produce ciphertexts

$$\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Enc}\big(\mathsf{MPK}_i, (\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i))\big)$$

4. For each $i \notin \mathcal{I}$, run the one-query simulator $\mathsf{OneQFE.Sim}$ to produce ciphertexts CT_i as follows: at most one of $\Gamma_1, \ldots, \Gamma_q$ contains i.
   - If such a set exists, let j denote the unique set Γ_j that contains i (i.e. $i \in \Gamma_j$). If $j \leq q^*$, compute

$$\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Sim}\big(\mathsf{MPK}_i, (\eta_j(i), G_{C_j,\Delta_j}, \mathsf{SK}_{C_j,\Delta_j,i})\big)$$

     where $\mathsf{SK}_{C_j,\Delta_j,i}$ is provided as part of SK_j.
   - If no such set exist or $j > q^*$, then compute

$$\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Sim}\big(\mathsf{MPK}_i, \emptyset\big)$$

Output $(\mathsf{CT}_i)_{i=1}^N$ as the ciphertext.

Simulating secret key SK_j, for $j > q^*$. Here, the simulator gets $\mathsf{MSK} = (\mathsf{MSK}_1, \ldots, \mathsf{MSK}_N)$ and $C_j(x), C_j$ and needs to simulate $(\mathsf{SK}_{C_j,\Delta_j,i})_{i \in \Gamma_j}$.

1. For each $i \in \Gamma_j \cap \mathcal{I}$, pick $\mathsf{SK}_{C_j,\Delta_j,i} \leftarrow \mathsf{OneQFE.Keygen}(\mathsf{MSK}_i, G_{C_j,\Delta_j})$.
2. For each $i \in \Gamma_j \setminus \mathcal{I}$ (i.e, Γ_j is the only set that contains i),
   (a) pick a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$ and subject to the constraints on the values in $\mathcal{I}$ chosen earlier;
   (b) run $\mathsf{OneQFE.Sim}(\mathsf{MSK}_i, (\eta_j(i), G_{C_j,\Delta_j}))$ to obtain $\mathsf{SK}_{C_j,\Delta_j,i}$ so that CT_i decrypts to $\eta_j(i)$.

Output $(\mathsf{SK}_{C_j,\Delta_j,i})_{i \in \Gamma_j}$.

We establish security via a series of Games. The simulator is described above.

Game 1. We modify $\zeta_1, \ldots, \zeta_S, \eta_1, \ldots, \eta_q$ to be the same as that in the simulator.

Game 2. We simulate $(\mathsf{CT}_i)_{i \notin \mathcal{I}}$ and $\mathsf{SK}_j, j > q^*$ as in the simulator.

Game 3. The output of the simulator. That is, we modify how polynomials $\mu_1, \ldots, \mu_\ell$ are sampled.

Claim 5.3.1.1. *The outputs of Game* 0 *and Game* 1 *are identically distributed.*

Proof. In the normal encryption, ζ_{a^*} is chosen at random and $\eta_j(\cdot)$ is defined by the relation. From Step 2 in the ciphertext simulation and Step 2 in the secret keys simulation (for $j > q^*$) $\mathsf{BdFE.Sim}$, essentially, chooses $\eta_j(\cdot)$ at random which defines ζ_{a^*}. It is easy to see that reversing the order of how the polynomials are chosen produces the same distribution. □

Claim 5.3.1.2. *The outputs of Game* 1 *and Game* 2 *are computationally indistinguishable.*

Proof. Informally, this follows from the security of the underlying one-query FE scheme and the fact that for all $i \notin \mathcal{I}$, we run $\mathsf{OneQFE.Keygen}(\mathsf{MSK}_i, \cdot)$ at most once.

By a hybrid argument, it suffices to show that for all $i \notin \mathcal{I}$, the distribution of CT_i in Game 1 and 2 are computationally indistinguishable (given MPK_i and $\mathsf{SK}_1, \ldots, \mathsf{SK}_q$). Indeed, fix such a $i \notin \mathcal{I}$ and a corresponding unique j such that $i \in \Gamma_j$ (the case no such j exists is similar).

First, observe that amongst $\mathsf{SK}_1, \ldots, \mathsf{SK}_q$, only SK_j contains a key $\mathsf{SK}_{C_j,\Delta_j,i}$ that is generated using either $\mathsf{SK}_{C_j,\Delta_j,i} \leftarrow \mathsf{OneQFE.Keygen}(\mathsf{MSK}_i, G_{C_j,\Delta_j})$ (for the non-adaptive queries) or $\mathsf{SK}_{C_j,\Delta_j,i} \leftarrow \mathsf{OneQFE.Sim}(\mathsf{MSK}_i, (\eta_j(i), G_{C_j,\Delta_j}))$ (for the adaptive queries).

Case 1: Assume $j \leq q^*$. Observe that

$$\begin{aligned} \eta_j(i) &= C_j(\mu_1(i), \ldots, \mu_\ell(i)) + \sum_{a \in \Delta_j} \zeta_a(i) \\ &= G_{C_j,\Delta_j}(\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i)) \end{aligned} \tag{5.2}$$

which means that in both Games 1 and 2, CT_i decrypts to the same value. Now, note that in Game 1, CT_i is generated using

$$\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Enc}\big(\mathsf{MPK}_i, (\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i))\big)$$

By the security of the underlying FE scheme, this is computationally indistinguishable from

$$\mathsf{OneQFE.Sim}\big(\mathsf{MPK}_i, (G_{C_j,\Delta_j}(\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i)), G_{C_j,\Delta_j}, \mathsf{SK}_{C_j,\Delta_j,i})\big)$$

By the Equation 5.2, this is the same as

$$\mathsf{OneQFE.Sim}\big(\mathsf{MPK}_i, (\eta_j(i), G_{C_j,\Delta_j}, \mathsf{SK}_{C_j,\Delta_j,i})\big)$$

which is the distribution of CT_i in Game 2.

Case 2: Assume $j > q^*$. Then:

- $\mathsf{CT}_i \leftarrow \mathsf{OneQFE.Sim}\big(\mathsf{MPK}_i, \emptyset\big)$ and

- $\mathsf{SK}_{C_j,\Delta_j,i} \leftarrow \mathsf{OneQFE.Sim}(\mathsf{MSK}_i, (\eta_j(i), G_{C_j,\Delta_j}))$

Similarly, by the Equation 5.2 and by the security of the underlying one-query FE scheme this simulated pair of ciphertext and secret key is indistinguishable from the real. □

Claim 5.3.1.3. *The outputs of Game 2 and Game 3 are identically distributed.*

Proof. In Game 2, the polynomials $\mu_1, \ldots, \mu_\ell$ are chosen with constant terms $x_1, \ldots, x_\ell$, respectively. In Game 3, these polynomials are now chosen with 0 constant terms. This only affects the distribution of $\mu_1, \ldots, \mu_\ell$ themselves and polynomials $\zeta_1, \ldots, \zeta_S$. Moreover, only the evaluations of these polynomials on the points in $\mathcal{I}$ affect the outputs of the games. Now observe that:

- The distribution of the values $\{\mu_1(i), \ldots, \mu_\ell(i)\}_{i\in\mathcal{I}}$ are identical in both Game 2 and 3. This is because in both games, we choose these polynomials to be random degree t polynomials (with different constraints in the constant term), so their evaluation on the points in $\mathcal{I}$ are identically distributed, since $|\mathcal{I}| \leq t$.
- The values $\{\zeta_1(i), \ldots, \zeta_S(i)\}_{i\in\mathcal{I}}$ depend only on the values $\{\mu_1(i), \ldots, \mu_\ell(i)\}_{i\in\mathcal{I}}$.

The claim follows readily from combining these observations. □

5.4 A Bootstrapping Theorem for Functional Encryption

In this section, we show a "bootstrapping-type" theorem for functional encryption (FE). In a nutshell, this shows how to take a q-query functional encryption scheme for "bounded degree" circuits, and transform them into a q-query functional encryption scheme for arbitrary polynomial-size circuits. The transformation relies on the existence of a pseudorandom generator (PRG) that stretches the seed by a constant factor, and which can be computed by circuits of degree $\mathsf{poly}(\kappa)$. This is a relatively mild assumption, and in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

In a high-level the idea is this: Suppose we wish to construct an FE scheme for a family $\mathcal{C}$ of polynomial-size circuit. Let $C \in \mathcal{C}$ and x be some input. Then, let $\widetilde{C}(x; R)$ denote a randomized encoding of C that is computable by a constant-depth circuit with respect to the inputs x and R. By [AIK06, Theorem 4.14], we know that assuming the existence of a pseudorandom generator in $\oplus$L/poly, such a randomized encoding exists for every polynomial-size circuit C.

Consider a new family of circuits $\mathcal{G}$ defined as follows:

$$G_{C,\Delta}(x, R_1, \ldots, R_S) := \widetilde{C}\left(x; \bigoplus_{a\in\Delta} R_a\right)$$

Observe the following:

- Since for any C, $\widetilde{C}(\,\cdot\,;\,\cdot\,)$ is computable by a constant-depth circuit, then $G_{C,\Delta}(\,\cdot\,;\,\cdot\,)$ is computable by a constant-degree polynomial. Using the result from the previous scheme, we have a (q, one)-AD-SIM-secure FE scheme for G.

- Given a functional encryption scheme ford $\mathcal{G}$, it is easy to construct one for $\mathcal{C}$. Decryption works by first recovering the output of $G_{C,\Delta}$ and then applying the decoder for the randomized encoding.
- Informally, $(1, one)$-AD-SIM-security follows from the fact that the ciphertext together with the secret key reveals only the output of $\widetilde{C}(x)$, which in turn reveals no more information than $C(x)$. More formally, given $C(x)$, we can simulate $\widetilde{C}(x)$ and then the ciphertext, using first the simulator for the randomized encoding and then that for the underlying FE scheme.
- The role of the subset Δ is similar to that in the preceding construction — to "rerandomize" the randomness used in G, which is necessary to achieve (q, one)-AD-SIM-security.

Functional Encryption Scheme for $\mathcal{C}$. Let $(\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ be a (q, one)-AD-SIM-secure scheme for $\mathcal{G}$, with a simulator $\mathsf{BdFE.Sim}$. We construct an encryption scheme $(\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for $\mathcal{C}$ works as follows (that takes parameters S, v as before).

- **Setup** $\mathsf{FE.Setup}(1^\kappa)$**:** Run the bounded FE setup algorithm to generate a master public-key/secret-key pair $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{BdFE.Setup}(1^\kappa)$.
- **Key Generation** $\mathsf{FE.Keygen}(\mathsf{MSK}, C)$**:** On input the master secret key MSK and a circuit $C \in \mathcal{C}$, do the following:
  1. Choose a uniformly random set $\Delta \subseteq [S]$ of size v;
  2. Generate the secret key $\mathsf{SK}_{C,\Delta} \leftarrow \mathsf{BdFE.Keygen}(\mathsf{MSK}, G_{C,\Delta})$.

  Output as secret key $\mathsf{SK}_C := (\Delta, \mathsf{SK}_{C,\Delta})$.
- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key MPK and an input message $x \in \mathcal{X}$, do the following:
  1. For $i = 1, 2, \ldots, S$, choose uniformly random $R_i \xleftarrow{\$} \{0,1\}^r$.
  2. Run the bounded degree encryption algorithm $\mathsf{BdFE.Enc}$ to produce a ciphertext
  $$\mathsf{CT} \leftarrow \mathsf{BdFE.Enc}(\mathsf{MPK}, (x, R_1, \ldots, R_S))$$

  Output CT as the ciphertext.
- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** On input a secret key SK_C and a ciphertext CT,
  - Run the bounded FE decryption algorithm to get $\widetilde{y} \leftarrow \mathsf{BdFE.Dec}(\mathsf{SK}_{C,\Delta}, \mathsf{CT})$.
  - Run the randomized encoding decoder on $\widetilde{y}$ to get the output $y \leftarrow \mathsf{RE.Decode}(\widetilde{y})$.

Correctness

We first show correctness of the scheme $\mathcal{FE}$. Given a secret key SK_C and a ciphertext $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x)$, the decryption algorithm computes

$$\widetilde{y} = \mathsf{BdFE.Dec}(\mathsf{SK}_{C,\Delta}, \mathsf{CT}) = G_{C,\Delta}(x, R_1, \ldots, R_S) = \widetilde{C}(x; \bigoplus\nolimits_{a \in \Delta} R_a))$$

Of course, running $\mathsf{RE.Decode}$ on this should return $y = C(x)$, by the correctness of the randomized encoding scheme.

Bootstrapping for Unbounded Queries. Although the transformation above assumes the knowledge of q (the bound on the number of secret key queries of the adversary), we can generalize it to work for unbounded queries as follows. Essentially, the idea is to generate fresh (computational) randomness for each randomized encoding using a pseudo-random function.

In particular, let $\{\mathsf{prf}_S\}_{S\in\{0,1\}^\kappa}$ be a circuit family of weak pseudo-random functions. Consider a new circuit family $\mathcal{C}$ that works in the following way:

$$G_{C,R}(x,S)) := \widetilde{C}\Big(x; \mathsf{prf}_S(R)\Big)$$

Then, essentially the same construction as above works as a way to bootstrap an FE scheme for arbitrary circuits from FE schemes for circuits that can compute the weak PRF followed by the randomized encoding. Assuming the existence of weak PRFs and PRGs that can be computed by circuits of degree $\mathsf{poly}(\kappa)$, we then obtain functional encryption schemes for arbitrary circuits. Note, that by Chapter 4 it is impossible to achieve functional encryption for PRFs under NA-SIM-security for unbounded queries. However, constructions secure under a weaker security definition (for example, indistinguishability) are still open.

5.4.1 Proof of Security

Theorem 5.4.1. *Let $\mathcal{BDFE}$ be a (q, one)-AD-SIM-secure (resp. (q, one)-NA-SIM-secure) functional encryption scheme for any family of circuits computable in NC1. Then, for any family $\mathcal{C}$ of polynomial-size circuits the $\mathcal{FE}$ scheme described above is (q, one)-AD-SIM-secure (resp. (q, one)-NA-SIM-secure).*

We prove that the construction $\mathcal{FE}$ given in Section 5.4 is (q, one)-AD-SIM-*secure* if we start out with a (q, one)-AD-SIM-secure scheme. This subsumes the non-adaptive variant of the proof.

Proof overview. Suppose the adversary sees q^* queries before seeing the ciphertext. The simulator has to simulate the ciphertext and answer the remaining secret key queries. We may again assume that the simulator knows all of $\Gamma_1, \ldots, \Gamma_q, \Delta_1, \ldots, \Delta_q$.

Simulating the ciphertext. The simulator gets $\{C_j(x), C_j, \mathsf{SK}_{C_j}\}_{j\in[q^*]}$ and outputs:

$$\mathsf{CT} \leftarrow \mathsf{BdFE.Sim}\Big(\mathsf{MPK}, \big\{\mathsf{RE.Sim}(C_j(x)), G_{C_j,\Delta_j}, \mathsf{SK}_{C_j,\Delta_j}\big\}_{j\in[q^*]}\Big)$$

with fresh independent randomness for each of the q^* invocations of $\mathsf{RE.Sim}$.

Simulating secret key SK_{C_j}, for $j > q^*$. Here, the simulator gets MSK and $C_j(x), C_j$ and needs to simulate $\mathsf{SK}_{C_j} := (\Delta_j, \mathsf{SK}_{C_j,\Delta_j})$. It proceeds as follows:

1. Picks $\tilde{y}_j \leftarrow \mathsf{RE.Sim}(C_j(x))$.
2. Runs $\mathsf{BdFE.Sim}(\mathsf{MSK}, (\tilde{y}_j, G_{C_j,\Delta_j}))$ to obtain $\mathsf{SK}_{C_j,\Delta_j}$ so that CT decrypts to $\tilde{y}_j$.

Output $\mathsf{SK}_{C_j} = (\Delta_j, \mathsf{SK}_{C_j,\Delta_j})$.

Details. We establish security via a series of Games, where the last Game corresponds to the simulator described above.

Game 0. Normal encryption.

Game 1. We modify the distribution of the ciphertext to use $\mathsf{BdFE.Sim}$ as in the static case for both the ciphertext and the secret-key queries after the adversary sees the ciphertext. That is,

$$\mathsf{CT} \leftarrow \mathsf{BdFE.Sim}\Big(\mathsf{MPK}, \big\{G_{C_j,\Delta_j}(x; R_1, \ldots, R_S), G_{C_j,\Delta_j}, \mathsf{SK}_{C_j,\Delta_j}\big\}_{j\in[q^*]}\Big)$$

Moreover, for $j > q^*$, it

1. Picks $\tilde{y}_j \leftarrow G_{C_j,\Delta_j}(x; R_1, \ldots, R_S)$.
2. Runs $\mathsf{BdFE.Sim}(\mathsf{MSK}, (\tilde{y}_j, G_{C_j,\Delta_j}))$ to obtain $\mathsf{SK}_{C_j,\Delta_j}$ so that CT decrypts to $\tilde{y}_j$.

Output $\mathsf{SK}_{C_j} = (\Delta_j, \mathsf{SK}_{C_j,\Delta_j})$.

Game 2. We replace $\big\{\bigoplus_{a\in\Delta_j} R_a\big\}_{j\in[q]}$ with $\big\{\ R'_j\ \big\}_{j\in[q]}$, where for each j:

$$G_{C_j,\Delta_j}(x; R_1, \ldots, R_S) := \widetilde{C}(x; \bigoplus\nolimits_{a\in\Delta_j} R_a)$$

Game 3. The output of the simulator (that is, switch to using $\mathsf{RE.Sim}$).

Claim 5.4.1.1. *The outputs of Game* 0 *and Game* 1 *are computationally indistinguishable.*

Proof. This follows readily from (q, one)-$\mathsf{AD\text{-}SIM}$-security of the underlying FE scheme. □

Claim 5.4.1.2. *The outputs of Game* 1 *and Game* 2 *are identically distributed.*

Proof. By cover-freeness of $\Delta_1, \ldots, \Delta_q$, we have that

$$\Big\{\bigoplus\nolimits_{a\in\Delta_j} R_a\Big\}_{j\in[q]} \quad \text{and} \quad \Big\{\ R'_j\ \Big\}_{j\in[q]}$$

are identically distributed. □

Claim 5.4.1.3. *The outputs of Game* 2 *and Game* 3 *are computationally indistinguishable.*

Proof. This follows readily from a hybrid argument and the security of the randomized encoding scheme, which says that for each $j = 1, \ldots, q$:

$$\widetilde{C}_j(x; R'_j) \quad \text{and} \quad \mathsf{RE.Sim}(C_j(x))$$

are computationally indistinguishable. □

5.5 Yet Another Bootstrapping Theorem Using FHE

We show a bootstrapping theorem that transforms a q-query FE scheme from Section 5.3 into a q-query FE scheme for arbitrary polynomial-size circuits using, in addition, a fully homomorphic encryption scheme [Gen09a, BV11]. Intuitively, the construction can be viewed as follows: we reduce functional encryption for a circuit C to one for the decryption algorithm for a fully homomorphic encryption scheme computable in NC1. Putting this together with the q-query, NC1 ciruit scheme from Section 5.3 gives us Theorem 5.5.1.

First, we need a generalization of the construction for NC1 circuits from Section 5.3. Assume that the message is split into a *public part* and a *secret part*. Then, the key observation is that the construction from Section 5.3 works for any circuit C which is computable in NC1 in the variables of the secret part. The rationale for this is the same as that used to obtain a predicate encryption with public index from the scheme in Section 5.3. In particular, we do not need to secret share the *public* part of the input.

We show the following theorem:

Theorem 5.5.1. *Let $\mathcal{BDFE}$ be a q-query, FE scheme which works for any circuit computable in NC1 in the secret part of the input, and let $\mathcal{FHE}$ be a semantically secure fully homomorphic encryption scheme whose decryption algorithm* $\mathsf{FHE.Dec}(\mathsf{SK}, ct)$ *can be implemented by an NC1 circuit in the secret key. Then, for any family of poly-size circuits $\mathcal{C}$ there exists a q-query FE scheme $\mathcal{FE} = (\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$.*

*Furthermore, if $\mathcal{BDFE}$ is (q, one)-*NA-SIM*-secure (resp. (q, one)-*AD-SIM*-secure), then so is $\mathcal{FE}$.*

Any of the recent fully homomorphic encryption schemes have decryption algorithms computable in NC1. Putting these together, we get q-bounded FE schemes under the "learning with errors" and the "ring learning with errors" assumptions (together with certain circular security assumptions) [BV11].

Let $\mathcal{C}$ be an arbitrary polynomial-size circuit family. Our construction uses the following components:

- **An Inner Encryption Scheme:** Let $\mathcal{FHE} = (\mathsf{FHE.Keygen}, \mathsf{FHE.Enc}, \mathsf{FHE.Eval}, \mathsf{FHE.Dec})$ be a fully homomorphic encryption scheme where the decryption algorithm $\mathsf{FHE.Dec}$ can be implemented by an NC1 circuit in the secret key.
- **An Outer Encryption Scheme:** Let $\mathcal{BDFE} = (\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ be a q-query functional encryption scheme for the family $\mathcal{G}$ that is computable by NC1 circuits in their secret input defined as follows:

  $$G_C(ct, \mathsf{SK}) := \left[ct, \mathsf{FHE.Dec}(\mathsf{SK}, \mathsf{FHE.Eval}(C, ct))\right]$$

  Note that although $\mathcal{G}$ has circuits that are at least as large as those for $\mathcal{C}$, all we are interested in is its degree in the secret input, namely SK.

Our q-query secure encryption scheme $(\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ for $\mathcal{C}$ works as follows.

- **Setup** $\mathsf{FE.Setup}(1^\kappa)$**:** Run the bounded FE setup algorithm to generate a master public-key/secret-key pair:

  $$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{BdFE.Setup}(1^\kappa)$$

- **Key Generation** $\mathsf{FE.Keygen}(\mathsf{MSK}, C)$**:** On input the master secret key MSK and a circuit $C \in \mathcal{C}$, run the bounded FE key generation algorithm to generate a secret key

$$\mathsf{SK}_C \leftarrow \mathsf{BdFE.Keygen}(\mathsf{MSK}, G_C)$$

  for the circuit G_C.

- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, x)$**:** On input the master public key MPK and an input message $x \in \mathcal{X}$:

  1. Choose a uniformly random public-key/secret-key pair for the fully homomorphic encryption scheme $\mathcal{FHE}$ by running

$$(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{FHE.Keygen}(1^\kappa)$$

  2. Encrypt the input message x using the FHE encryption algorithm

$$ct \leftarrow \mathsf{FHE.Enc}(\mathsf{PK}, x)$$

  3. Run the bounded FE encryption algorithm to encrypt the ciphertext ct together with the fully homomorphic secret key SK:

$$\mathsf{CT} \leftarrow \mathsf{BdFE.Enc}(\mathsf{MPK}, (ct, \mathsf{SK}))$$

  Output CT as the ciphertext.

- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** On input a secret key SK_C and a ciphertext CT, run the bounded FE decryption algorithm to get $[ct, y] \leftarrow \mathsf{BdFE.Dec}(\mathsf{SK}_C, \mathsf{CT})$, and output $[ct, y]$.

Correctness and Security

We first show correctness of the scheme $\mathcal{FE}$. Given a secret key SK_C and a ciphertext $\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x)$, the decryption algorithm computes

$$\begin{aligned}
[ct, y] &= \mathsf{BdFE.Dec}(\mathsf{SK}_C, \mathsf{CT}) \\
&= \mathsf{BdFE.Dec}(\mathsf{SK}_C, \mathsf{BdFE.Enc}(\mathsf{MPK}, (ct, \mathsf{SK}))) \\
&\qquad\qquad (\text{where } ct \leftarrow \mathsf{FHE.Enc}(\mathsf{PK}, x)) \\
&= G_C(ct, \mathsf{SK}) \\
&= [ct, \mathsf{FHE.Dec}(\mathsf{SK}, \mathsf{FHE.Eval}(C, ct))] \\
&= [ct, C(x)]
\end{aligned}$$

We establish security via a series of Games. The simulator is described in Game 2.

Game 0. Normal encryption.

Game 1. Run the q-query simulator on input $([ct \leftarrow \mathsf{FHE.Enc}(\mathsf{PK}, x), C_i(x)], G_{C_i}, \mathsf{SK}_i)_{i=1}^n$, where $n \leq q$ is the number of oracle query calls made to $\mathsf{BdFE.Keygen}$.

Game 2. Run the q-query simulator on input $([ct \leftarrow \mathsf{FHE.Enc}(\mathsf{PK}, 0), C_i(x)], G_{C_i}, \mathsf{SK}_i)_{i=1}^n$, where $n \leq q$ is the number of oracle query calls made to $\mathsf{BdFE.Keygen}$.

5.6 Probabilistic Proofs

5.6.1 Small Pairwise Intersection

Lemma 5.6.1. *Let $\Gamma_1, \ldots, \Gamma_q \subseteq [N]$ be randomly chosen subsets of size $tD + 1$. Let $t = \Theta(q^2\kappa), N = \Theta(D^2q^2t)$. Then,*

$$Pr\left[\left|\bigcup_{i\neq j}(\Gamma_i \cap \Gamma_j)\right| \leq t\right] = 1 - 2^{-\Omega(\kappa)}$$

where the probability is over the random choice of the subsets $\Gamma_1, \ldots, \Gamma_q$.

Proof. For all $i, j \in [q]$ such that $i \neq j$, let X_{ij} be a random variable denoting the size of the intersection of S_i and S_j. Let

$$X = \sum_{i,j\in[q],i\neq j} X_{ij}$$

It is not hard to see that X_{ij}'s are independent random variables. By the linearity of expectation,

$$E[X] = \sum_{i,j\in[q],i\neq j} E[X_{ij}]$$

Now, for a fixed set S_i and a randomly chosen S_j the size of the intersection of S_i and S_j follows a hypergeometric distribution, where $tD + 1$ serves both as the number of success states and number of trials, and N is the population size. Therefore,

$$E[X_{ij}] = \frac{(tD+1)(tD+1)}{N} = \frac{(tD+1)^2}{N}$$

Hence,

$$\mu = E[X] = \frac{q(q-1)(tD+1)^2}{N} \leq \frac{10q^2t^2D^2}{N}$$

By Chernoff bound, for any $\sigma \geq 0$:

$$Pr[X > (1+\sigma)\mu] < exp\left(\frac{-\sigma^2}{2+\sigma}\mu\right)$$

Setting $t = \Theta(q^2\kappa), N = \Theta(D^2q^2t)$ gives us $\mu = \Theta(t) = \Theta(q^2\kappa)$. Applying Chernoff bound,

$$Pr[X > t] = 2^{-\Omega(\kappa)}$$

□

5.6.2 Cover-Freeness

Lemma 5.6.2. *Let $\Delta_1, \ldots, \Delta_q \subseteq [S]$ be randomly chosen subsets of size v. Let $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$. Then, for all $i \in [q]$*

$$Pr[\Delta_i \setminus \left(\bigcup_{j\neq i} \Delta_j\right) \neq \phi] = 1 - 2^{-\Omega(\kappa)}$$

where the probability is over the random choice of subsets $\Delta_1, \ldots, \Delta_q$.

Proof. Let $i \in [q]$ be arbitrary. Let $G := \bigcup_{j \neq i} \Delta_j$. Clearly, $|G| = (q-1)v$. Let X be the random variable denoting $|\Delta_i \setminus G|$. Now,

$$|\Delta_i \setminus G| = |\Delta_i| - |\Delta_i \cap G| = v - |\Delta_i \cap G|$$

Hence,

$$E[X] = v - E[|\Delta_i \cap G|]$$

Now, $E[|\Delta_i \cap G|]$ follows a hypergeometric distribution with v success states, $v(q-1)$ trials and S population size. Hence,

$$E[|\Delta_i \cap G|] = \frac{v^2(q-1)}{S}$$

Therefore, $E[X] = v - (v^2(q-1))/S$. Setting, $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$ we obtain that $\mu = E[X] = \Theta(\kappa)$. By Chernoff bound, for any $0 \leq \sigma \leq 1$:

$$Pr[X \leq (1-\sigma)\mu] < exp\left(\frac{-\sigma^2}{2}\mu\right)$$

Applying it we obtain that $Pr[X \leq (1-\sigma)\mu] = 2^{-\Omega(\kappa)}$. Hence,

$$Pr[\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j\right) \neq \phi] = Pr[X > 0] \geq Pr[X > (1-\sigma)\mu] = 1 - 2^{-\Omega(\kappa)}$$

□

Chapter 6

Conclusion and Open Problems

This work initiated the study of feasibility results for general functional encryption. We showed that it is impossible to construct non-adaptive (and adaptive) simulation-secure general functional encryption for unbounded collusions even for one input message. We also presented a construction secure under the adaptive simulation-based definition for bounded collusions. Despite our impossibility results, functional encryption field is full of interesting open problems. In particular, we can ask for:

- a general construction for unbounded collusions under weaker notions of security, such as indistinguishability-based. However, note that [BSW11] presented a circuit family and a construction which satisfies indistinguishability security, yet intuitively leaks too much information about the input (in fact, the decryptor learns the whole input.) One can also ask for different and meaningful notions of security for functional encryption which are not ruled out by the existing impossibility results.

- a many message unbounded collusions simulation-secure functional encryption for public-index predicates. Our result presented in Section 4 only rules out secret-index circuit families.

Bibliography

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *Asiacrypt*, 2011.

[AGVW12] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. Cryptology ePrint Archive, Report 2012/468, 2012. http://eprint.iacr.org/.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin / Heidelberg, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matt Franklin, editor, *Advances in Cryptology CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 197–206. Springer Berlin / Heidelberg, 2004.

[BB11] Dan Boneh and Xavier Boyen. Efficient selective identity-based encryption without random oracles. *J. Cryptology*, 24(4):659–693, 2011.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 1–10, New York, NY, USA, 1988. ACM.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 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 *TCC*, pages 535–554, 2007.

[CCKM00] Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Müller. One-round secure computation and secure autonomous mobile agents. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, ICALP '00, pages 512–523, London, UK, UK, 2000. Springer-Verlag.

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996. Longer version at `http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-682.pdf`.

[CHH+07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In *ASIACRYPT*, pages 502–518, 2007.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In *In EUROCRYPT*, pages 65–82. Springer-Verlag, 2002.

[DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.

[FKN94] Uri Feige, Joe Killian, and Moni Naor. A minimal model for secure computation (extended abstract). In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, pages 554–563, New York, NY, USA, 1994. ACM.

[Gen06] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.

[Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, pages 579–591, 2008.

[GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.

[GLW12] Shafi Goldwasser, Allison B. Lewko, and David A. Wilson. Bounded-collusion IBE from key homomorphism. In *TCC*, pages 564–581, 2012.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 21–30, New York, NY, USA, 2007. ACM.

[KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, EUROCRYPT'08, pages 146–162, Berlin, Heidelberg, 2008. Springer-Verlag.

[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 *EUROCRYPT*, pages 62–91, 2010.

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

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 404–, Washington, DC, USA, 2003. IEEE Computer Society.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[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.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.

[SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

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

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, pages 53–70, 2011.

[Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, pages 218–235, 2012.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 80 –91, nov. 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.