# Functional Encryption with Bounded Collusions via Multi-Party Computation*

Sergey Gorbunov† Vinod Vaikuntanathan‡ Hoeteck Wee§

September 5, 2012

**Abstract**

We construct a functional encryption scheme secure against an a-priori bounded polynomial number of collusions for the class of all polynomial-size circuits. Our constructions require only semantically secure public-key encryption schemes and pseudorandom generators computable by small-depth circuits (known to be implied by most concrete intractability assumptions). For certain special cases such as predicate encryption schemes with public index, the construction requires only semantically secure encryption schemes, which is clearly the minimal necessary assumption.

Our constructions rely heavily on techniques from secure multi-party computation and randomized encodings. All our constructions are secure under a strong, adaptive simulation-based definition of functional encryption.

**Keywords:** Functional Encryption, Multi-Party Computation, Randomized Encodings.

*A preliminary version of this work appeared in the Proceedings of the $32^{nd}$ Annual International Conference on Cryptology (CRYPTO 2012).

†University of Toronto. Email: sgorbunov@cs.toronto.edu. Supported by NSERC Alexander Graham Bell Graduate Scholarship.

‡University of Toronto. Email: vinodv@cs.toronto.edu. Supported by an NSERC Discovery Grant and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

§George Washington University. Email: hoeteck@alum.mit.edu. Supported by NSF CAREER Award CNS-1237429.

# Contents

# 1 Introduction

Traditional notions of public-key encryption provide *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. While such "black-and-white" notions of encryption have served us well for the past thirty years and are indeed being widely used for secure communications and storage, it is time to move beyond. In particular, the advent of cloud computing and the resulting demand for privacy-preserving technologies demands a much more fine-grained access control mechanism for encrypted data.

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 $\mathsf{SK}_C$ with every circuit $C$, and ciphertexts $\mathsf{CT}$ with every input $x$. The owner of the secret key $\mathsf{SK}_C$ and the ciphertext $\mathsf{CT}$ 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 $C_1(x), \ldots, C_q(x)$.

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 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 a similar vein, fuzzy IBE schemes correspond to a circuit that detects proximity between two strings, and attribute based encryption schemes correspond to circuit that can be computed by Boolean formulas. 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?*

To date, constructions of functional encryption are known only for these limited classes of circuits (see [BF01, Coc01, SW05, GPSW06, KSW08, LOS+10] and others). More concretely, the state-of-the-art constructions are limited to predicate encryption schemes, where the predicate itself is computable by a "low complexity" class, such as Boolean formula and inner product 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 $C_1, \ldots, C_q$. This raises the following natural question: can we build functional encryption schemes for arbitrary circuits for some meaningful relaxation of this security requirement?

[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].

**Functional Encryption for Bounded Collusions.** In this work, 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). 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 main result in this paper 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 $\mathsf{SK}_C$ 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.

### 1.1 Our Results

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 by polynomials of degree $\mathrm{poly}(\kappa)$, where $\kappa$ is the security parameter. 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* (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, $\mathsf{ind}$ is the so-called public index, and $\mu$ is sometimes refered to as the payload message. For the special 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-degree PRG" assumption for this construction.

In contrast, functional encryption schemes that handle an unbounded number of secret-key queries are known only for very limited classes of circuit families, the most general being inner

product predicates [KSW08, LOS+10, OT10]. In particular, constructing an unbounded-query secure functional encryption scheme for general circuit families is considered a major open problem in this area [BSW11]. As for functional encryption schemes with public index (also referred to as "attribute-based encryption" by some authors) that handle an unbounded number of secret-key queries, there are a handful of constructions for polynomial-size formulas [GPSW06, OSW07], which themselves are a sub-class of NC1 circuits.

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** (Main Theorem, Informal)**.** *Let $\kappa$ be a security parameter. Assuming the existence of semantically secure encryption schemes as well as PRGs computable by arithmetic circuits of degree-$\mathsf{poly}(\kappa)$, for every $q = q(\kappa)$, there exists a general functional encryption scheme secure against $q$ secret key queries.*

**Corollary 1.2** (Informal)**.** *Let $\kappa$ 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.*

We have so far avoided discussing the issue of which security definition to use for functional encryption. Indeed, there are a number of different definitions in the literature, including both *indistinguishability* style and *simulation* style definitions. In a nutshell, we prove our constructions secure under a strong, adaptive simulation-based definition; see Section 1.3 for details.

## 1.2 Overview of Our Constructions

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]).[2] 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.

### 1.2.1 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.

[2] We note that [SS10] is completely insecure for collusions of size two: in particular, given two secret keys $\mathsf{SK}_{0^\ell}$ and $\mathsf{SK}_{1^\ell}$, an adversary can derive the $\mathsf{SK}_C$ for any other $C$, and moreover, completely recover $x$.

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. [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 $\mathsf{MPK}_i$ 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 $\Gamma$ is completely determined by $C$.)

- To encrypt $x$, the encryptor first chooses $\ell$ random polynomials $\mu_1, \ldots, \mu_\ell$ of degree $t$ with constant terms $x_1, \ldots, x_\ell$ respectively. The encryptor computes $\mathsf{CT}_i$ 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 $\mathsf{CT}_i$ 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 $\mathsf{MPK}_j$. Since security of $\mathsf{MPK}_j$ 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 $\mathsf{MPK}_j$, 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.

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

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.

### 1.2.2 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 6 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 [Gen09, BV11]. See appendix 7 for more details.

### 1.3 Definitions of Functional Encryption

Our constructions are shown secure under a strong simulation-based definition, in both the adaptive and non-adaptive sense. The non-adaptive variant requires the adversary to make all its secret key queries before receiving the challenge ciphertext whereas in the adaptive variant, there is no such restriction. Although the adaptive variant is clearly stronger, Boneh, Sahai and Waters [BSW11] recently showed that it is also impossible to achieve, even for very simple circuit families (related to IBE). We observe that the BSW impossibility result holds only if the adversary obtains an unbounded number of ciphertexts (essentially because of a related lower bound for non-committing encryption schemes with unbounded messages). Faced with this state of affairs, we show our constructions are shown secure in the non-adaptive sense, as well as in the adaptive sense with a bounded number of messages.

In addition, we show a number of implications between different variants of these definitions; see Section 3 and Appendix A for more details.

### 1.4 A Perspective: Bounded-Use Garbled Circuits

The reason why the construction of Sahai and Seyalioglu only achieves security against collusions of size 1 is intimately related to the fact that Yao's garbled circuits become completely insecure when used more than once. Our constructions may be viewed as a stateless variant of Yao's garbled circuit that can be reused for some a-priori bounded number of executions. Fix two-parties inputs to be $C$ and $x$. We can view the ciphertext as encoding of a "universal" circuit of $U_x(\cdot)$ on some fixed input value $x$, such that we can "delegate" computation on $q$ different inputs $C_1,\ldots,C_q$ without leaking any information about $x$ beyond $C_1(x),\ldots,C_q(x)$.

**Organization of the Paper.** We describe the preliminaries and a simulation-based definition of functional encryption in Sections 2 and 3, respectively. For completeness, we describe a construction for 1-query functional encryption and prove its security in the adaptive setting in Section 4. Readers familiar with this construction can go ahead to the next section. We describe our Construction 1 for NC1 circuits in Section 5 and our Construction 2 for bootstrapping in Section 6. An additional FHE-based Construction 3 for bootstrapping is presented in Section 7. An interested reader is referred to the appendices for the definitional implications.

## 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. For $n \in \mathbb{N}$, let $[n]$ denote the set of numbers $1, \ldots, n$. Let $\kappa$ denote the security parameter.

### 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 $x \in \mathcal{X}_\kappa$ and outputs $C(x) \in \mathcal{Y}_\kappa$.

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 $\mathsf{MSK}$ and a circuit $C \in \mathcal{C}_\kappa$ and outputs a corresponding secret key $\mathsf{SK}_C$.
- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, x)$ is a p.p.t. algorithm that takes as input the master public key $\mathsf{MPK}$ and an input message $x \in \mathcal{X}_\kappa$ and outputs a ciphertext $\mathsf{CT}$.
- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$ is a deterministic algorithm that takes as input the secret key $\mathsf{SK}_C$ and a ciphertext $\mathsf{CT}$ and outputs $C(x)$.

**Definition 2.1** (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* $\mathsf{FE.Setup}$*,* $\mathsf{FE.Keygen}$*, and* $\mathsf{FE.Enc}$*.*

Refer to Section 3 for the security definition.

### 2.2 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 $\mu_1(i) + \mu_2(i) = (\mu_1 + \mu_2)(i)$ meaning that that sharing scheme is additively homomorphic. Multiplying shares gives us $\mu_1(i)\mu_2(i) = (\mu_1\mu_2)(i)$ meaning that the scheme is also multiplicatively homomorphic (where $\mu_1\mu_2$ denotes the product of the polynomials). 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).

### 2.3 Public Key Encryption.

A public key encryption scheme $\mathcal{PKE} = (\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, over message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa\in\mathbb{N}}$, is a triple of PPT 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 $\mathsf{PK}$ and a message $M \in \mathcal{M}_\kappa$ and outputs a ciphertext $\mathsf{CT}$.
- **Decryption.** $\mathsf{PKE.Dec}_{\mathsf{SK}}(\mathsf{CT})$**:** takes the secret key $\mathsf{SK}$ and a ciphertext $\mathsf{CT}$ and outputs a message $M^* \in \mathcal{M}_\kappa$.

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

**Definition 2.2.** *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* $\mathsf{PKE.Setup}$, $\mathsf{PKE.Enc}$.

**Definition 2.3.** *A public key encryption scheme $\mathcal{PKE}$ is $(t, \epsilon)$-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* $\mathsf{PKE.Enc}$ *and the coins of the adversary $\mathcal{A}$.*

### 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 $\mathcal{C}(k, x) \in \{0,1\}^m$. A decomposable randomized encoding scheme $\mathcal{RE}$ consists of two algorithms $(\mathsf{RE.Encode}, \mathsf{RE.Decode})$ satisfying the following properties:

1. **Decomposable Encoding.** $\mathsf{RE.Encode}(1^\kappa, \mathcal{C}, x)$**:** A p.p.t. algorithm takes as inputs a security parameter, a description of a circuit $\mathcal{C}$, an input $x$ and outputs a randomized encoding:

$$(\tilde{\mathcal{C}}_1(\cdot, x; R), \ldots, \tilde{\mathcal{C}}_\ell(\cdot, x; R)) \text{ for } i \in [\ell], \text{ where } \tilde{\mathcal{C}}_i(\cdot, x; R) \text{ depends only on } k_i$$

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

---

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

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 $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^{\ell}$

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

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].

# 3 Security of Functional Encryption against Bounded Collusions

In this section, we first describe simulation-based definitions for functional encryption with *bounded* collusions, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10]. We then go on to discuss relations between various flavors of these definitions, with details in Appendix A.

**Definition 3.1** ($q$-NA-SIM- and $q$-AD-SIM- Security)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a circuit family $\mathcal{C} = \{\mathcal{C}_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$. 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:*

---

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

1: $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(x, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
3: $\boxed{\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, x)}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK}, \mathsf{CT}, st)}$
5: *Output* $(\alpha, x)$

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

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

---

*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* $C \in \mathcal{C}_\kappa$.

*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* $q$-query simulation-secure for one message against adaptive adversaries ($q$-AD-SIM-secure, for short) *if there is an* admissible *p.p.t. simulator* $S = (S_1, S_2)$ *such that for every p.p.t. adversary* $A = (A_1, A_2)$ *that makes at most* $q$ *queries, the following two distributions are computationally indistinguishable:*

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

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* $q$-query simulation-secure for one message against non-adaptive adversaries ($q$-NA-SIM-secure, for short) *if there is a p.p.t. simulator* $S = (S_1, \perp)$ *such that for every p.p.t. adversary* $A = (A_1, A_2)$ *that makes at most* $q$ *queries, the two distributions above are computationally indistinguishable.*

Intuitively, our 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;
   (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 $\mathsf{MSK}$ 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][1], except we only consider a *single* ciphertext and impose an upper bound on the number of secret key queries.

[1] A sketch of the proof is presented in the Appendix A.

**Why focus on this definition?** First, as mentioned above, our definition is at least as strong as the definition presented in [BSW11]. In addition, in Appendix A 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 1 and proved in Appendix A. As a result of these definitional implications, we focus on proving that our constructions are secure under the *single* message simulation definitions for both adaptive and non-adaptive worlds.

# 4 Background Constructions

## 4.1 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-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 $\mathsf{MSK}$ 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 $\mathsf{MPK}$ 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-AD-SIM-secure. We define a simulator 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 $\lambda$-wise repetition (that is, $\lambda$ independent master public keys, etc).

### 4.2 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-AD-SIM-secure FE encryption for singleton circuit space $\mathcal{C}' = \{C^*\}$, then we obtain a 1-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 $\mathsf{MSK}$ 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 $\mathsf{MPK}$ 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 $\mathsf{SK}_b$ 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-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 $\mathsf{CT}_0$ and $\mathsf{CT}_1$.
  - if $b = 1$, follow $\mathsf{BFFE.Sim}$ (that is, generate $\mathsf{CT}_0$ 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 $\mathsf{SK}_0$ using $\mathsf{BasicFE.Sim}$);
– if $b = 1$, follow normal encryption (that is, generate $\mathsf{SK}_1$ 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-AD-SIM of the underlying scheme. The same applies to the outputs of Games 1 and 2.

### 4.3 One-Query General Functional Encryption from Randomized Encoding

Sahai and Seyalioglu [SS10] proved 1-NA-SIM; we observe the same "bootstrapping" construction works for 1-AD-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 $\lambda$, 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 $i$th 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 $\lambda$ 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 $\mathsf{MSK}$ 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 $\mathsf{MPK}$ 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-NA-SIM-*secure* (See below on how to modify the proof to show 1-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 $\mathsf{MPK}_i$ 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 $\lambda$, 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 4.0.1.** *The outputs of Game* 0 *and Game* $\lambda$ *are computationally indistinguishable.*

*Proof.* The only different between Games 0 and $\lambda$ 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 4.0.2.** *The outputs of Game $\lambda$ 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-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 BFFE.Sim many times for all $\mathsf{MSK}_i$. For every oracle queries $\widetilde{U}_i(C[i], \cdot\,; \cdot)$ made by 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 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 $\ell$-tuple of field elements, and for every circuit $C \in \mathcal{C}$, $C(\cdot)$ is an $\ell$-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.

## 5.1 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 5.2).

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

$$G_{C,\Delta}(x, Z_1, \ldots, Z_S) := C(x) + \sum_{i\in\Delta} Z_i \tag{1}$$

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

Let (OneQFE.Setup, OneQFE.Keygen, OneQFE.Enc, OneQFE.Dec) be a functional encryption scheme for $\mathcal{G}$ secure against a *single* secret key query. Our $q$-query secure encryption scheme $\mathcal{BDFFE} = (\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ 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 $\mathsf{MSK}$ 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 $\eta(0)$.

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

### 5.2 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 $\Gamma$ and $\Delta$ 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 $\mathsf{MPK}_j$. Since security of $\mathsf{MPK}_j$ 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 $\mathsf{MPK}_j$. 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 $N(\kappa) = \Theta(D^2q^2t)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$. For details, we refer an interested reader to Appendix B.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 $1 - 2^{-\Omega(q^2v^2/S)}$) as long as $q^2v^2/S \leq v/100$. 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 Appendix B.2.

We remark that in our construction, multiple secret key queries for the same $C \in \mathcal{C}$ result in different secret keys $\mathsf{SK}_C$, essentially because of the different random choices of the sets $\Delta$ and $\Gamma$. 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 Proof of Security

**Theorem 5.1.** *Let $\mathcal{ONEQFE}$ be a* 1-AD-SIM*-secure (resp.* 1-NA-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* $q$-AD-SIM*-secure (resp.* $q$-NA-SIM*-secure).*

We prove that the construction $\mathcal{BDFE}$ given in Section 5 is $q$-AD-SIM-*secure* if we start out with a 1-AD-SIM-secure scheme. This subsumes the non-adaptive variant of the proof. By Theorem A.1, this implies that $\mathcal{BDFE}$ is $q$-NA-SIM-*secure* for many messages. However, it is only single-message

$q$-AD-SIM-*secure* (see Figure 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 $\Gamma_j, \Delta_j$ from $\mathsf{SK}_j$;
- for $j > q^*$, the simulator gets to program $\Gamma_j, \Delta_j$ 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 $\mathsf{CT}_i$ 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 $\mathsf{CT}_i$ 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 $\ell$ 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 $\zeta_a$ 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 $\zeta_{a^*}$ 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 $\zeta_a$ 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 $\mathsf{CT}_i$ as follows: at most one of $\Gamma_1, \ldots, \Gamma_q$ contains $i$.

   - If such a set exists, let $j$ denote the unique set $\Gamma_j$ 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 $\mathsf{SK}_j$.
   - 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 $\mathsf{SK}_j$, 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, $\Gamma_j$ 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 $\mathsf{CT}_i$ 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 5.1.1.** *The outputs of Game* 0 *and Game* 1 *are identically distributed.*

*Proof.* In the normal encryption, $\zeta_{a^*}$ 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 $\zeta_{a^*}$. It is easy to see that reversing the order of how the polynomials are chosen produces the same distribution. □

**Claim 5.1.2.** *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 $\mathsf{CT}_i$ in Game 1 and 2 are computationally indistinguishable (given $\mathsf{MPK}_i$ 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 $\mathsf{SK}_j$ 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

$$\begin{aligned} \eta_j(i) &= C_j(\mu_1(i), \ldots, \mu_\ell(i)) + \sum_{a \in \Delta_j} \zeta_a(i) \\ &= G_{C_j,\Delta_j}(\mu_1(i), \ldots, \mu_\ell(i), \zeta_1(i), \ldots, \zeta_S(i)) \end{aligned} \tag{2}$$

which means that in both Games 1 and 2, $\mathsf{CT}_i$ decrypts to the same value. Now, note that in Game 1, $\mathsf{CT}_i$ 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 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 $\mathsf{CT}_i$ 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 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 5.1.3.** *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. □

# 6 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 pseudo-random 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 $q$-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-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 $\Delta$ is similar to that in the preceding construction — to "rerandomize" the randomness used in $G$, which is necessary to achieve $q$-AD-SIM-security.

**Functional Encryption Scheme for $\mathcal{C}$.** Let $(\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ be a $q$-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 $\mathsf{MSK}$ 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 $\mathsf{MPK}$ 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 $\mathsf{CT}$ as the ciphertext.
- **Decryption** $\mathsf{FE.Dec}(\mathsf{SK}_C, \mathsf{CT})$**:** On input a secret key $\mathsf{SK}_C$ and a ciphertext $\mathsf{CT}$,
  - 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 $y \leftarrow \mathsf{RE.Decode}(\widetilde{y})$.

### 6.0.1 Correctness

We first show correctness of the scheme $\mathcal{FE}$. Given a secret key $\mathsf{SK}_C$ 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 [AGVW12] 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.

## 6.1 Proof of Security

**Theorem 6.1.** *Let $\mathcal{BDFE}$ be a $q$-AD-SIM-secure (resp. $q$-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 $q$-AD-SIM-secure (resp. $q$-NA-SIM-secure).*

We prove that the construction $\mathcal{FE}$ given in Section 6 is $q$-AD-SIM-*secure* if we start out with a $q$-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 RE.Sim.

**Simulating secret key $\mathsf{SK}_{C_j}$, 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 $\mathsf{CT}$ 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 6.1.1.** *The outputs of Game* 0 *and Game* 1 *are computationally indistinguishable.*

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

**Claim 6.1.2.** *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 6.1.3.** *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. □

# 7 Yet Another Bootstrapping Theorem Using FHE

We show a bootstrapping theorem that transforms a $q$-query FE scheme supporting NC1 circuits into a $q$-query FE scheme for arbitrary polynomial-size circuits using, in addition, a fully homomorphic encryption scheme [Gen09, 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 gives us Theorem 7.1.

First, we need a generalization of the construction for NC1 circuits from Section 5. 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 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.

We show the following theorem:

**Theorem 7.1.** *Let $\mathcal{BDFE}$ be a $q$-query, FE scheme which works for any NC1 circuit, 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 $q$-$\mathsf{NA\text{-}SIM}$-secure (resp. $q$-$\mathsf{AD\text{-}SIM}$-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 $\mathsf{SK}$.

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 $\mathsf{MSK}$ 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 $\mathsf{MPK}$ 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 $\mathsf{SK}$:

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

  Output $CT$ as the ciphertext.

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

### 7.0.1 Correctness and Security

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

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

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 $\mathsf{BdFE.Keygen}$.

## References


[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.

[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.

[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.

[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.

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

[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.

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

[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.

[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 *EUROCRYPT*, pages 146–162, 2008.

[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 Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010.

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

[OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010.

[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.

[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.

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

# A Relations between Definitions of Functional Encryption

In this section, we first describe simulation-based and indistinguishability-based definitions for many input messages functional encryption, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10]. We then go on to show relations between various flavors of these definitions.

## A.1 A Simulation-based Definition

**Definition A.1** (NA-SIM- and AD-SIM- Security)**.** *Let $\mathcal{FE}$ be a functional encryption scheme for a circuit family $\mathcal{C} = \{\mathcal{C}_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa\}_{\kappa\in\mathbb{N}}$. 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:*

---

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

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* $(\alpha, x_1, \ldots, x_\ell)$

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

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* $\mathsf{SK}_i$ *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*.*

Note that this definition is the generalization of the one presented in Section 3 to the case where the adversary receives multiple ciphertexts. Intuitively, the above security definition states that whatever information adversary is able to learn from the ciphertexts and secret keys, can be obtained by a simulator from the secret keys and the outputs of the functionality for the messages only.

We remark that our definitions imply (and are stronger than) those of presented in the work of Boneh, Sahai and Waters [BSW11]. More formally, for the adaptive variant we can instantiate [BSW11] simulator $(Sim_1, Sim_O, Sim_2)$ as follows.

1. $Sim_1$ runs FE.Setup and sets $pp := \mathsf{MPK}, \sigma := \mathsf{MSK}$.
2. $Sim_O$ runs FE.Keygen algorithm on MSK and updates $\sigma$ to include all oracle queries and replies $(C_i, \mathsf{SK}_i)$.
3. $Sim_2$ 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 $A^\circ$ on the ciphertext, and on any FE.Keygen call it uses our $S_2$ to obtain a secret key. Finally, output the same $\alpha$ as $A^\circ$. The non-adaptive variant follows similarly.

## A.2 An Indistinguishability-Based Definition

**Definition A.2** (NA-IND- and AD-IND-Security)**.** *Let* $\mathcal{FE}$ *be a functional encryption scheme for a circuit family* $\mathcal{C} = \{\mathcal{C}_\kappa : \mathcal{X}_\kappa \to \mathcal{Y}_\kappa\}_{\kappa\in\mathbb{N}}$*. For every function* $\ell = \ell(\kappa)$*, every p.p.t. adversary* $A = (A_1, A_2)$*, consider the following two experiments:*

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

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* $b$

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

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* $\kappa$*:*

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

   *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* $\kappa$*.*

   *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.

### A.3 Relations Between Definitions

In this section, we prove the following relations between the definitions.

- *Non-adaptive Definitions:* When considering non-adaptive definitions (namely, where the adversary is constrained to making secret key queries only *before* he receives the challenge ciphertext), we show that *one-message* definitions are equivalent to *many-message* definitions, both in the indistinguishability and the simulation worlds.

  Put together, this shows that it is sufficient to prove security for one message in the simulation sense, which is precisely what we will do for our schemes.

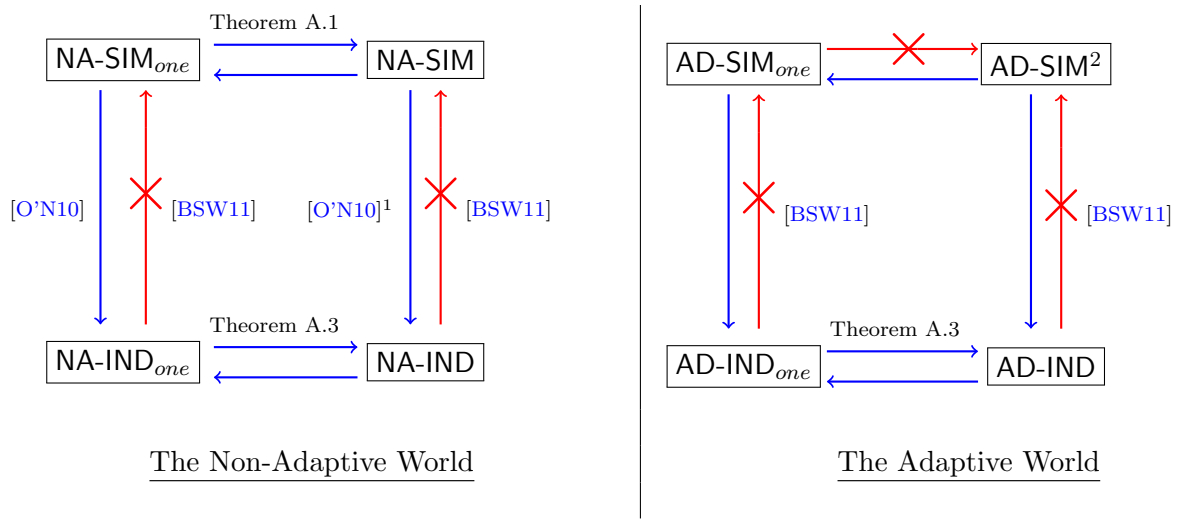


Figure 1: Relations between definitions of functional encryption in the non-adaptive and adaptive flavors. Regular blue arrows indicate an implication between the definitions, and a red arrow with a cross on it indicates a separation. The citations for all non-trivial implications and separations are 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.)

- *Adaptive Definitions:* When considering adaptive definitions (namely, where the adversary is allowed to make secret key queries *after* receiving the challenge ciphertext) we show that for any $q$, $(q, one)$-AD-SIM implies $(q, one)$-AD-IND which is equivalent to $(q, many)$-AD-IND. We also construct a functional encryption scheme and prove it secure under $(q, one)$-AD-SIM definition. Therefore, from the work of Boneh et al. [BSW11] we can conclude that $(q, one)$-AD-SIM does not imply $(q, many)$-AD-SIM.

These relationships are summarized in Figure 1.

**Theorem A.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

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

[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 FE.Keygen calls (2-bounded collusion). Since we show a secure construction satisfying AD-SIM$_{one}$, this implication follows.

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 $\mathsf{CT}^r$'s and $\mathsf{CT}^s$'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

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

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 A.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 $\alpha := (b', st')$.

[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 $\mathsf{SK}$.

Now, in the real experiment $b = b'$ with probability $1/2 + \epsilon$ for some noticeable $\epsilon$. 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 A.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. For simplicity we denote it as $q$-NA-SIM- *and* $q$-AD-SIM- *security.*

# B Probabilistic Proofs

## B.1 Small Pairwise Intersection

**Lemma B.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,

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

□

## B.2 Cover-Freeness

**Lemma B.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,

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

□