Functional Encryption with Bounded Collusions via Multi-Party Computation

Sergey Gorbunov[⋆1], Vinod Vaikuntanathan[⋆⋆1], and Hoeteck Wee[⋆⋆⋆2]

[1] University of Toronto
[2] George Washington University

Abstract. We construct functional encryption schemes for polynomial-time computable functions secure against an a-priori bounded polynomial number of collusions. 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.

Along the way, we show a "bootstrapping theorem" that builds a q-query functional encryption scheme for arbitrary functions starting from a q-query functional encryption scheme for *bounded-degree* functions. All our constructions rely heavily on techniques from secure multi-party computation and randomized encodings.

Our constructions are secure under a strong simulation-based definition of functional encryption.

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 requires that we come up with a much more fine-grained access control mechanism for encrypted data.

[⋆] Supported by NSERC Alexander Graham Bell Graduate Scholarship.

[⋆⋆] 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.

[⋆⋆⋆] Supported by NSF CAREER Award CNS-1237429.

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 function $F(\cdot,\cdot)$ on two inputs – a "key" K and a "message" M – associates secret keys SK_K with every K and ciphertexts CT with every M. The owner of a secret key SK_K and a ciphertext CT of a message M should be able to obtain $F(K, M)$, but learn nothing else about the message M itself.

Constructions of functional encryption are known only for limited classes of functions (see [BF01,Coc01,SW05,GPSW06,KSW08,LOS$^+$10] and others), leaving open a challenging open question: *Can we build functional encryption schemes for arbitrary (polynomial-time) functions?*

Much like its predecessors, functional encryption schemes are required to satisfy rather stringent security notions. In particular, a large part of the difficulty in constructing functional encryption schemes lies in the fact that we typically require security against adversaries who obtain secret keys for an unbounded number of inputs $K_1, \ldots, K_q$.

In this work, we consider a relaxed notion of security where the adversary is given secret keys for an *a-priori bounded number of inputs* $K_1, \ldots, K_q$ 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 aituations. In particular, one could envision scenarios where an authority releases secret keys for arbitrarily many inputs K, however the adversary can only form collusions of size at most q.

Our main result in this paper is a construction of q-bounded secure functional encryption schemes for *arbitrary polynomial-time functions* under *minimal 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 more challenging landscape since (1) a secret key SK_K in functional encryption 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 key and the message together. Our constructions leverage interesting ideas from the study of (information-theoretic) multi-party computation (à la [BGW88,BMR90,DI05]) and randomized encodings [Yao86,IK00,AIK06].

1.1 Our Results

The main result of this work is the construction of q-query functional encryption schemes that:

1. handle arbitrary (polynomial-time computable) functions, and
2. are based on the existence of semantically secure public key encryption schemes, and pseudorandom generators (PRG) computable by polynomials of degree poly(κ), where κ 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 function F defined as:

$$F(K,(M,\mu)) = \begin{cases} (M,\mu) \text{ if } g(K,M) = \mathsf{true} \\ (M,\bot) \text{ otherwise} \end{cases}$$

for some predicate g. In other words, the second input for F is divided into two parts – a "public index" M on which the computation takes place, and a secret "payload" μ which is conditionally released. For the special case of predicate encryption schemes with *public index*, our construction handles arbitrary polynomial-time functions 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 functions, the most general being inner product predicates [KSW08,LOS$^+$10,OT10]. In particular, constructing an unbounded-query secure functional encryption scheme for general functions 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 NC^1 circuits.

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

Theorem 1 (Main Theorem, Informal). *Let κ 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.*

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 Techniques

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]).[3] Building on this, our construction proceeds in two steps.

Construction 1: Functional Encryption for Bounded-Degree Functions In the first step, we show how to construct a q-query functional encryption scheme for bounded-degree functions starting from any 1-query scheme (such as the one of [SS10]).

By bounded degree, we mean functions F such that for every K, the degree of the restriction $F(K,\cdot)$ is bounded a-priori by $D = D(\kappa)$ in the variables of M. This captures both arithmetic and boolean circuits of depth at most $\log D$ (with constant multiplicative fan-in). 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 construction assumes only the existence of semantically secure public-key encryption schemes. In addition, it also gives us for free a predicate encryption scheme with public index for *arbitrary polynomial-time* functions (with no a-priori bound on the degree).

The starting point of our construction is the BGW semi-honest MPC protocol without degree reduction (c.f. [DI05, Section 2.2]). Our main idea is to use the fact that this protocol is *completely non-interactive* when used to compute *bounded-degree* functions.

Suppose the encryptor holds input $M = (M_1, \ldots, M_\ell)$, the decryptor holds input K, and the goal is for the decryptor to learn $F(K, M_1, \ldots, M_\ell)$. The public/secret keys of the system consists of N independent public/secret keys for the 1-query scheme. To encrypt M, the encryptor first chooses ℓ random polynomials $\mu_1, \ldots, \mu_\ell$ of degree t with constant terms $M_1, \ldots, M_\ell$ respectively. In addition, she chooses a random polynomial ζ of degree Dt with constant term 0. (Here, t and N are parameters of the construction). Now, since $F(K,\cdot)$ has degree at most D, observe that

$$P(\cdot) := F(K, \mu_1(\cdot), \ldots, \mu_\ell(\cdot)) + \zeta(\cdot)$$

is a univariate polynomial of degree at most Dt and whose constant term is $F(K,(M_1, \ldots, M_\ell))$. The encryptor simply encrypts the shares $P(1), \ldots, P(N)$ using the N public keys.

The key generation algorithm associates the receiver with a random subset $\Gamma \subseteq [N]$ of size $Dt + 1$ and generates secret keys for the public keys MPK_i

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

for $i \in \Gamma$. Using the underlying (1-query) FE scheme, the decryptor learns the evaluation of P on the points in Γ, which allows her to recover $F(K, M_1, \ldots, M_\ell)$.

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

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 message M, which together reveal no information about M.

For technical reasons, we cannot establish security of the basic scheme. Informally, we need to rerandomize the polynomial P for each of the q queries. This can be done by having the encryptor hard-code additional randomness into the ciphertext. For more details, see Section 4.

To obtain a *predicate encryption scheme with public index*, we observe that the construction above satisfies a more general class of functions. 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 function F 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 F in its secret inputs. The bottomline is that since predicate encryption schemes with *public index* deal with functions that have very low degree in the secret input (degree 1, in particular), our construction handles arbitrary predicates.

Construction 2: Bootstrapping Theorem for Functional Encryption 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 degree-D circuits into one that is q-query secure for arbitrary polynomial-time functions, 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) function F, we use it to compute its randomized encoding $\widetilde{F}$ which is typically a much easier function to compute.

In particular, secret keys are generated for K and the encryption algorithm for the bounded-degree scheme is used to encrypt the pair (M, R), where R is a uniformly random string. The rough intuition for security is that the randomized encoding $\widetilde{F}(K, M; R)$ reveals "no more information than" $F(K, M)$ 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 $\widetilde{F}(K_1, M; R), \ldots, \widetilde{F}(K_q, M; R)$. Furthermore, since all these encodings use *the same randomness* R, the regular notion of security of randomized encodings does not apply *as-is*. We solve this issue by hard-coding a large number of random strings (proportional to q) in the ciphertext and using a cover-free set construction, ensuring that the adversary learns q randomized encodings with independently chosen randomness. See Section 5 for more details.

Putting this construction together with a randomized encoding scheme for polynomial-time computable functions (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 small functions into a q-query FE scheme for polynomial-time functions, using a fully homomorphic encryption scheme [Gen09,BV11]. We defer the details to the full version.

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 functionalities (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. Due to lack of space, we deal with the non-adaptive setting in this paper, postponing a discussion of adaptive security to the full version.

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 a two-party functionality $F(K, M)$. Specifically, we can view the ciphertext as encoding computation of $F(\,\cdot\,, M)$ on some fixed value M, such that we can "delegate" computation on q different inputs $K_1, \ldots, K_q$ without leaking any information about M beyond $F(K_1, M), \ldots, F(K_q, M)$.

Organization of the Paper. After describing a simulation-based definition of functional encryption in Section 2, we describe Construction 1 for bounded-degree circuits in Section 4 and Construction 2 for bootstrapping in Section 5. For completeness, we have also included the construction of a 1-query functional encryption in the appendix.

2 Functional Encryption against Bounded Collusions

A functional encryption scheme for a family of functions $F = \{F_\kappa : \mathcal{K}_\kappa \times \mathcal{M}_\kappa \to \{0,1\}\}_{\kappa \in \mathbb{N}}$ is a four-tuple of algorithms $(\mathsf{FE.Setup}, \mathsf{FE.Keygen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ where:

- $\mathsf{FE.Setup}(1^\kappa)$ generates a pair of keys – a master public key MPK and a master secret key MSK;
- $\mathsf{FE.Keygen}(\mathsf{MSK}, K)$ takes as input $K \in \mathcal{K}_\kappa$ and generates a secret key SK_K;
- $\mathsf{FE.Enc}(\mathsf{MPK}, M)$ takes as input $M \in \mathcal{M}_\kappa$ and generates a ciphertext CT; and
- given SK_K and CT, $\mathsf{FE.Dec}$ outputs $y \in \{0,1\}$.

We require that for all but a negligible fraction of $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$, for all $SK_K \in \mathsf{FE.Keygen}(\mathsf{MSK}, K)$ and all $\mathsf{CT} \in \mathsf{FE.Enc}(\mathsf{MPK}, M)$, the decryption algorithm $\mathsf{FE.Dec}(SK_K, \mathsf{CT})$ outputs $y = F_\kappa(K, M)$.

We now 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 deferred to the full version.

Definition 1 (q-NA-SIM- and q-AD-SIM- Security). *Let $\mathcal{FE}$ be a functional encryption scheme for a family of functions $F = \{F_\kappa : \mathcal{K}_\kappa \times \mathcal{M}_\kappa\}_{\kappa \in \mathbb{N}}$. 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.*

We distinguish between two cases of the 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 $S_2^{F(\cdot, M)}(\mathsf{MSK}, st', \cdot)$.*

   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 K, S_2 makes just a single query to its oracle $F(\cdot, M)$ on K itself.*

```
$\underline{\mathsf{Exp}^{\mathsf{real}}_{\mathcal{FE},A}(1^\kappa)}$:
1: $(\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(M, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
3: $\boxed{\mathsf{CT} \leftarrow \mathsf{FE.Enc}(\mathsf{MPK}, M)}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot)}(\mathsf{MPK},\mathsf{CT}, st)}$
5: Output $(\alpha, M)$
```

```
$\underline{\mathsf{Exp}^{\mathsf{ideal}}_{\mathcal{FE},S}(1^\kappa)}$:
1: $(\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
2: $(M, st) \leftarrow A_1^{\mathsf{FE.Keygen}(\mathsf{MSK},\cdot)}(\mathsf{MPK})$
   ▶ Let $(K_1, \ldots, K_q)$ be $A_1$'s oracle queries
   ▶ Let $\mathsf{SK}_i$ be the oracle reply to $K_i$
   ▶ Let $\mathcal{V} := \{F_\kappa(K_i, M), K_i, \mathsf{SK}_i\}$.
3: $\boxed{(\mathsf{CT}, st') \leftarrow S_1(\mathsf{MPK}, \mathcal{V}, 1^{|M|})}$
4: $\boxed{\alpha \leftarrow A_2^{\mathcal{O}'(\mathsf{MSK},st',\cdot)}(\mathsf{MPK},\mathsf{CT}, st)}$
5: Output $(\alpha, M)$
```

*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 function alone. A number of remarks on this definition are in order.

1. In the non-adaptive definition, the only difference between the real and ideal experiments is in how the ciphertext is generated – in the real experiment, the ciphertext is an encryption of M, whereas in the ideal experiment, the simulator generates a simulated ciphertext given (the secret keys $\mathsf{SK}_{K_1}, \ldots, \mathsf{SK}_{K_q}, K_1, \ldots, K_q$ and) the output values $F(K_1, M), \ldots, F(K_q, M)$. In the adaptive definition, we additionally allow the simulator to "program" the answers to the post-ciphertext secret-key queries.
2. Even if the the adversary does not request any secret keys, he learns the length of M 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.

3. We remark that our definitions imply (and are stronger than) those of presented in the work of Boneh, Sahai and Waters [BSW11]. We defer a discussion of this and other definitional implications to the full version.

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 the full version of this paper, 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.

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.

3 Preliminaries and Standard Cryptographic Definitions

3.1 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).

3.2 Decomposable Randomized Encoding

Let $\mathcal{C}$ be a circuit that takes inputs $K \in \{0,1\}^\ell, M \in \{0,1\}^n$ and outputs $\mathcal{C}(K, M) \in \{0,1\}^m$. A decomposable randomized encoding scheme $\mathcal{RE}$ consists of two algorithms ($\mathsf{RE.Encode}, \mathsf{RE.Decode}$) satisfying the following properties:

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

   $(\tilde{\mathcal{C}}_1(\cdot, M; R), \dots, \tilde{\mathcal{C}}_\ell(\cdot, M; R))$ for $i \in [\ell]$, where $\tilde{\mathcal{C}}_i(\cdot, M; R)$ depends only on K_i

2. **Decoding.** $\mathsf{RE.Decode}((\widetilde{Y}_i)_{i=1}^\ell)$**:** On input an encoding of a circuit $\tilde{Y}_i = \tilde{\mathcal{C}}_i(K_i, M; R)$ for some $K = (K_1, \dots, K_\ell)$ output $\mathcal{C}(K, M)$.
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, every circuit $\mathcal{C}$, and inputs $K = (K_1, \dots, K_\ell)$ and M, the outputs of the following two distributions are computationally indistinguishable:

$$\left\{\tilde{\mathcal{C}}_i(K_i, M; R))_{i=1}^\ell \ : \ \tilde{\mathcal{C}}_i(\cdot, M; R) \leftarrow \mathsf{RE.Encode}(1^\kappa, \mathcal{C}, M)\right\} \approx_c \left\{\tilde{\mathcal{C}}_i(K_i, M; R))_{i=1}^\ell \leftarrow \mathsf{RE.Sim}(1^\kappa, \mathcal{C}, \mathcal{C}(K, M))\right\}$$

Note that such a randomized encoding for arbitrary polynomial-size circuits follows from Yao's garbled circuit construction [Yao86,AIK06].

4 A Construction for Bounded-Degree Functions

In this section, we construct a functional encryption scheme for functions that can be computed by circuits of bounded degree (see below for a precise definition), secure against an a-priori bounded number of non-adaptive secret key queries. Our construction will rely on any semantically secure public-key encryption scheme.

The Class of Functions. We consider the class of determinitic functions that computes a bounded-degree polynomial over the message space for some fixed degree bound D. That is, the message space $\mathcal{M} = \mathbb{F}^\ell$ is an ℓ-tuple of field elements, and for every key $K \in \mathcal{K}$, $F(K, \cdot)$ is an ℓ-variate polynomial over $\mathbb{F}$ of total degree at most D (in the second input). This captures both arithmetic and boolean circuits of depth at most $\log D$ (with constant multiplicative fan-in). Specifically, to handle boolean circuits, we let $\mathbb{F}$ be a sufficiently large field extension of $\mathbb{F}_2$. 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.

Building Block. The main result of this section shows how to construct a functional encryption scheme for degree-D functions secure against q (non-adaptive) secret-key queries, starting from one that is secure against *a single non-adaptive secret-key query*. Sahai and Seyalioglu [SS10] have shown that FE schemes (for general polynomial-time functions) secure against a single query can be readily constructed using Yao's garbled circuits [Yao86].

4.1 Our Construction

Let F be a functionality with circuits of degree $D = D(\kappa)$ in its second input (namely, the message M), 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 4.2).

We start by defining a new functionality G as follows:

$$G((K, \Delta), (M, Z_1, \ldots, Z_S)) := F(K, M) + \sum_{i \in \Delta} Z_i \tag{1}$$

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

Let $(\mathsf{OneQFE.Setup}, \mathsf{OneQFE.Keygen}, \mathsf{OneQFE.Enc}, \mathsf{OneQFE.Dec})$ be a functional encryption scheme for G secure against a *single* secret key query. Our q-query secure scheme $\mathcal{BDFE} = (\mathsf{BdFE.Setup}, \mathsf{BdFE.Keygen}, \mathsf{BdFE.Enc}, \mathsf{BdFE.Dec})$ for F 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** $\mathsf{BdFE.Keygen}(\mathsf{MSK}, K)$**:** On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality,
  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

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

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

– **Encryption** $\mathsf{BdFE.Enc}(\mathsf{MPK}, M)$**:** On input the master public key $\mathsf{MPK} = (\mathsf{MPK}_i)_{i=1}^{N}$ and a message $M = (M_1, \ldots, M_\ell) \in \mathcal{M}$:
  1. For $i = 1, 2, \ldots, \ell$, pick a random degree t polynomial $\mu_i(\cdot)$ whose constant term is M_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

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

  for $i = 1 \ldots N$.

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

– **Decryption** $\mathsf{BdFE.Dec}(\mathsf{SK}_K, \mathsf{CT})$**:** On input a secret key SK_K and a ciphertext CT, do the following:
  1. Parse $SK_K == (\Gamma, \Delta, (\mathsf{SK}_{K,\Delta,i})_{i\in\Gamma})$ and $\mathsf{CT} = (\mathsf{CT}_i)_{i=1}^{N}$.
  2. Compute a degree Dt polynomial $\eta(\cdot)$ such that

$$\eta(i) = \mathsf{OneQFE.Dec}(\mathsf{SK}_{K,\Delta,i}, \mathsf{CT}_i)$$

  for all $i \in \Gamma$.
  3. Output $\eta(0)$.

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

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

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

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

Hence, $\eta(0) = F(K, (M_1, \ldots, M_\ell)) = F(K, M)$.

4.2 Setting the Parameters

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

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

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 message M, which together reveal no information about M.

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

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

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

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

4.3 Proof of Security

Overview. We prove that $\mathcal{BDFE}$ is q-NA-SIM-*secure*. Recall that the simulator gets as input all of the following values: 1) the public key: $\mathsf{MPK} = (\mathsf{MPK}_1, \ldots, \mathsf{MPK}_N)$; 2) the queries and outputs of F: $K_1, \ldots, K_q$), and all the outputs $F(K_1, M), \ldots, F(K_q, M)$; 3) the corresponding secret keys: $\mathsf{SK}_1, \ldots, \mathsf{SK}_q$, which determine the sets $\Gamma_1, \ldots, \Gamma_q, \Delta_1, \ldots, \Delta_q$.

We describe our strategy for simulating the ciphertext $\mathsf{CT} = (\mathsf{CT}_1, \ldots, \mathsf{CT}_N)$. Let $\mathcal{I} := \bigcup_{j \neq j'} (\Gamma_j \cap \Gamma_{j'})$. We will consider two cases:

- $i \notin \mathcal{I}$: 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. Therefore, we may appeal to the security of the underlying one-query FE scheme. Specifically, we simulate CT_i computationally using the simulator for the underlying one-query FE scheme.
- $i \in \mathcal{I}$: 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 simulate CT_i statistically as in an honestly generated ciphertext.

We refer the reader to the full version for the formal proof of security.

5 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 polynomial-size circuit $F(K, M)$, and let $\widetilde{F}(K, M; R)$ denote a randomized encoding of F that is computable by a constant-depth circuit with respect to the inputs M 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 F.

Consider the function G that works in the following way:

$$G((K, \Delta), (M, R_1, \ldots, R_S)) := \widetilde{F}\left(K, M; \bigoplus_{a \in \Delta} R_a\right)$$

Observe the following:

- Since $G(K, \ \cdot \ ; \ \cdot \ )$ is computable by a constant-depth circuit, then $G((K, \Delta), \ \cdot \ )$ is computable by a constant-degree polynomial. Using the result from the previous scheme, we have a q-NA-SIM-secure FE scheme for G.
- Given a functional encryption scheme for G, it is easy to construct one for F. Decryption works by first recovering the output of G 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 $G(K, M)$, which in turn reveals no more information than $F(K, M)$. More formally, given $F(K, M)$, we can simulate $G(K, M)$ and then the ciphertext, using first the simulator for the randomized encoding and then that for the underlying FE scheme.
– The role of the subset Δ is similar to that in the preceding construction — to "rerandomize" the randomness used in G, which is necessary to achieve q-AD-SIM-security.

Functional Encryption Scheme for F. Let ($\mathsf{BdFE.Setup}$, $\mathsf{BdFE.Keygen}$, $\mathsf{BdFE.Enc}$, $\mathsf{BdFE.Dec}$) be a q-AD-SIM-secure scheme for 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 F 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** $\mathsf{FE.Keygen}(\mathsf{MSK}, K)$**:** On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality F, do the following:
  1. Choose a uniformly random set $\Delta \subseteq [S]$ of size v;
  2. Generate the secret key $\mathsf{SK}_{K,\Delta} \leftarrow \mathsf{BdFE.Keygen}(\mathsf{MSK}, (K, \Delta))$. for the functionality G.

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

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

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

$$\widetilde{Y} = \mathsf{BdFE.Dec}(\mathsf{SK}_K, \mathsf{CT}) = G((K, \Delta), (M, R_1, \ldots, R_S)) = \widetilde{F}(K, M; \bigoplus\nolimits_{a \in \Delta} R_a))$$

Of course, running $\mathsf{RE.Decode}$ on this should return $Y = F(K, M)$, by the correctness of the randomized encoding scheme.

The security of the scheme follows in a straightforward way from the security of $\mathcal{BDFE}$, and that of the randomized encoding.

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 family of weak pseudo-random functions. Consider a function G that works in the following way:

$$G((K, R), (M, S)) := \widetilde{F}\Big(K, M; \mathsf{prf}_S(R)\Big)$$

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.

References

[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.

[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.

[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 One-Query General Functional Encryption from Randomized Encodings

We describe a construction of a one-query functional encryption scheme that is essentially from Sahai and Seyalioglu [SS10]. They proved the construction secure in the 1-NA-SIM sense; we observe that their "bootstrapping" construction works for 1-AD-SIM. Let F be arbitrary polynomial-size functionality. We construct the scheme $\mathcal{ONEQFE}$ for F as follows.

Our starting point is a "brute-force construction" first presented by Boneh et al. [BSW11, Section 4.1]. We call this $\mathcal{BFFE}$. Essentially, Boneh et al. presented an NA-SIM-secure scheme for any functionality where the key space has polynomial size, starting from any semantically secure public-key encryption scheme. (They only claimed indistinguishability-based security, but it clearly satisfies simulation-based security too.) For simplicity, we just use their construction for the key-space $\mathcal{K} = \{0, 1\}$. In addition, we observe that the scheme can be made AD-SIM-secure (for bounded message spaces) by replacing the underlying encryption scheme by an appropriate "non-committing type" scheme; the details are deferred to the full version.

In a high-level the idea is this: suppose we wish to construct an FE scheme for a polynomial-size circuit $F(K, M)$, and let $\widetilde{F}(K, M; R)$ denote a randomized encoding of F where for every M, R, $\widetilde{F}(\,\cdot\,, M; R)$ has small locality; specifically, every output bit of $\widetilde{F}(K, M; R)$ depends only on one input bit of K. Assume the key has length λ. Then, we can write

$$\widetilde{F}(K, M; R) = (\widetilde{F}_1(K_1, M; R), \ldots, \widetilde{F}_\lambda(K_\lambda, M; R))$$

where $\widetilde{F}_i(\,\cdot\,, M; R)$ depends only on K_i, the ith bit of K.

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

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

  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** $\mathsf{FE.Keygen}(\mathsf{MSK}, K)$**:** On input the master secret key MSK and a key $K \in \mathcal{K}$ for the functionality, pick

$$\mathsf{SK}_{K,i} \leftarrow \mathsf{BFFE.Keygen}(\mathsf{MSK}_i, K_i) \qquad \text{for } i = 1, \ldots, \lambda$$

  Output as secret key $\mathsf{SK}_K := ((\mathsf{SK}_{K,i})_{i\in[\lambda]})$.
- **Encryption** $\mathsf{FE.Enc}(\mathsf{MPK}, M)$**:** On input the master public key MPK and a message $M \in \mathcal{M}$, compute

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

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

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