Functional Encryption for Randomized Functionalities

Vipul Goyal[*] Abhishek Jain[†] Venkata Koppula[‡] Amit Sahai[§]

Abstract

In this work, we present the first definitions and constructions for functional encryption supporting *randomized functionalities*. The setting of randomized functionalities require us to revisit functional encryption definitions by, for the first time, explicitly adding security requirements for *dishonest encryptors*, to ensure that they cannot improperly tamper with the randomness that will be used for computing outputs. Our constructions are built using indistinguishability obfuscation.

1 Introduction

Originally, encryption was thought of as a way to encrypt "point to point" communication. However, in the contemporary world with cloud computing and complex networks, it has become clear that we need encryption to offer more functionality. To address this issue, the notion of functional encryption (FE) has been developed [SW05, GPSW06, BW07, KSW08, BSW11, O'N10]. In a functional encryption for a family $\mathcal{F}$, it is possible to derive secret keys K_f for any function $f \in \mathcal{F}$ from a master secret key. Given an encryption of some input x, that user can use its secret key K_f to obtain $f(x)$, and should learn nothing else about x beyond $f(x)$.

A driving force behind functional encryption research has been to understand what class of functions can be supported by functional encryption. This remarkable line of research has progressed to now encompass all functions describable by deterministic polynomial-size circuits [SS10, GVW12, GKP$^+$13, CIJ$^+$13, GGH$^+$13]. We continue this line of research to move even beyond deterministic polynomial-size circuits: specifically, we consider the case of *randomized* functionalities. Indeed, not only are randomized functionalities strongly motivated by real-world scenarios, but randomized functionalities present new challenges for functional encryption. Techniques developed in the context of functional encryption for deterministic circuit do not directly translate into techniques for randomized circuits. To understand the basic technical problem, below we give an illustrative example.

Let us illustrate the desiderata for functional encryption for randomized functions by considering an example of performing an audit on an encrypted database through random sampling. Suppose there is a bank that maintains large secure databases of the transactions in each of its branches. There is an auditor Alice who would like to gain access to a random sample of database entries

[*] Microsoft Research, India. vipul@microsoft.com

[†] Boston University & MIT. abhishek@csail.mit.edu. Work done in part while visiting Microsoft Research, India.

[‡] UT Austin. kvenkata@cs.utexas.edu. Part of this research was conducted during internship at Microsoft Research, India.

[§] UCLA. sahai@cs.ucla.edu

from each branch in order to manually audit these records and check for improper transactions. We note that random sampling of transactions for manual analysis is quite common during audits. There are two primary concerns:

- The auditor wants to ensure that cheating in a branch is caught with reasonable probability.
- The organization wants to ensure that a malicious auditor cannot learn undesirable information (e.g., too much about a particular customer) from the encrypted databases. In particular, it wants to ensure that a malicious auditor cannot gain access to arbitrarily chosen parts of the database, but rather is limited to seeing only a randomly selected sample for each branch.

If we try to solve this problem naively using functional encryption, by giving the auditor a secret key SK_f that lets it obtain a random subset of an encrypted database CT, we are faced with the question: where does the randomness come from? Clearly, the randomness cannot be specified in the ciphertext alone since then a cheating encrypter (bank branch) could influence it. It cannot be specified in the decryption key alone as well: then auditor would get the same (or correlated) sample from the databases of different branches. (We also stress that since functional encryption does not guarantee function privacy, randomness present in the function f, even if chosen by a trusted party, would be known to Alice.)

Even if the randomness was chosen by an XOR of coins built into the decryption key and the ciphertext, this would allow malicious encryptors, over time, to ensure correlations among the random coins used by the auditor when inspecting different databases (or the same database after updates to it). Such correlations could potentially be used to eventually learn completely the coins embedded in the decryption key (based on the auditor's actions in response to planted improprieties in databases). Another option is to use a pseudorandom function (PRF) whose key is inbuilt in the decryption key. However again, since functional encryption does not guarantee function privacy, the PRF key could be completely leaked to a malicious auditor (rendering all guarantees about the sample looking "random" invalid).

This scenario also illustrates the importance of dealing with *dishonest encryptors* in the context of functional encryption for randomized functionalities, because of the influence they can have on the choice of coins used in computing the output. Indeed, this issue of dishonest encryptors was never considered explicitly in previous work on functional encryption, to the best of our knowledge. This is perhaps because in the context of deterministic functionalities, the issue of dishonest encryptors seems very related to simple correctness, which is not the case in the current work.

Defining functional encryption for randomized functionalities. To avoid the problems sketched in the examples above, we define functional encryption for randomized functionalities using the simulation paradigm: We want that an adversary, given SK_f and an honestly generated encryption of x, be simulatable given only $f(x;r)$ where r is true randomness that is completely unknown to the adversary. At the same time, consider an adversary that can generate dishonest ciphertexts $\hat{\mathsf{CT}}$ and learn from outside the output of decrypting $\hat{\mathsf{CT}}$ using a secret key SK_g (that is unknown to the adversary). We want such an adversary to be simulatable given only $g(\hat{x};r)$, where $\hat{x}$ is an input that is information-theoretically fixed by $\hat{\mathsf{CT}}$ and r is again true randomness that is unknown to the adversary. Note that a crucial feature of our definition is that if a party uses a secret key SK_f on a particular ciphertext CT, it will always get back $f(x;r)$ for the same randomness r. In other words, the user cannot repeatedly sample the functionality to obtain multiple outputs for different random coins. This allows users of our definition to more tightly control how much information an adversary or user learns. However, given two distinct ciphertexts CT_1 and CT_2 both encrypting x, a malicious user possessing SK_f should obtain exactly two independent samples of the output of the function: $f(x; r_1)$ and $f(x; r_2)$.

Application to differentially private data release. A natural application of functional encryption would be to provide non-interactive differentially private data release with high levels of accuracy. Consider a scenario where a government would like to allow researchers to carry out research studies on different hospital patient record databases, but only if the algorithm that analyzes the patient data achieves a sufficient level of differential privacy. Without using cryptography, methods for allowing the hospitals to publish differentially private data that would allow for meaningful and diverse research studies must incur very high accuracy loss [DNR+09]. An alternative would be to have a government agency review a specific research algorithm f, and if the algorithm guarantees sufficient privacy, to issue a secret key SK_f that the researcher could use to obtain the output of her algorithm on any hospital's encrypted patient records. Note that in such a setting, the hospital patient record could be encrypted and stored *without* any noise addition. The noise could be added by the algorithm f after computing the correct output. Such a setting would ensure very high accuracy (essentially the same as the interactive setting where the hospitals store data in clear and answer the researcher queries after adding noise in an online fashion).

Note however, to achieve differential privacy, such an algorithm f must be randomized. Furthermore, typical differentially private algorithms require that the randomness used to compute the output must be correctly and freshly sampled each time and be kept secret (or else the differential privacy could be completely compromised). By realizing functional encryption that would allow such randomized function evaluation, we would simultaneously remove the need for the hospital to participate in any study beyond simply releasing an encrypted database, and remove the need for the researcher to share his hypothesis and algorithm with any entity beyond the government regulatory body that issues secret keys.

1.1 Our Results

We show how to formalize the definition sketched above, generalizing the simulation-based definitions given in [BSW11, O'N10, CIJ+13], and then show how to build functional encryption schemes supporting arbitrary randomized polynomial-size circuits assuming indistinguishability obfuscation for circuits and one-way functions.

The starting point for our construction is the functional encryption scheme of [GGH+13] for polynomial-size deterministic circuits. In that scheme, in essence the secret key SK_f is built upon obfuscating the function f using an indistinguishability obfuscator [BGI+01]. We show how to modify this construction to achieve our notion of functional encryption for randomized functionalities by building upon the recently introduced idea of punctured programming [SW13]. In particular, we embed a psuedo-random function (PRF) key into the obfuscated program, which is executed on the ciphertext, to obtain the randomness used to derive the output. We adapt ideas from [DDN91, Sah99] to ensure that valid ciphertexts are unique. The core of our argument of security is to show that indistinguishability obfuscation guarantees the secrecy of the random coins derived by this method.

Our results immediately imply the application to differential privacy: Consider two "neighboring" databases x_0 and x_1. Differential privacy guarantees that the statistical distance between the distributions of outputs of the mechanism f for these two databases is at most e^ϵ, a small (but non-negligible) quantity. Now consider an adversary's view given an encryption of x_0. By our simulation-based notion of security, the adversary's view can be simulated given only $f(x_0; r)$ where r is true (secret) randomness. This view is e^ϵ close to the view that would be generated given only $f(x_1; r)$, by differential privacy of f. Finally we apply our definition to show that this view is negligibly close to the real adversary's view given an encryption of x_1. Thus, our functional encryption scheme when applied to f yields a computationally differentially private mechanism.

1.2 Subsequent Work

In this work, we consider functional encryption for *single-input* functions. In a recent work, Goldwasser, Goyal, Jain, and, Sahai[GGJS13] study the problem of (multi-input) functional encryption for n-ary functions (for arbitrary $n \geq 1$). In such a scheme, a party who owns a secret key SK_f for an n-ary function f can jointly decrypt n independently computed encryptions of $x_1, \ldots, x_n$ to learn $f(x_1, \ldots, x_n)$. We note that the techniques used in the present paper to handle *randomized* functionalities are relevant to the multi-input setting as well. In particular, Goldwasser et al. [GGJS13] build on our techniques to extend their results for multi-input functional encryption to the case of (n-ary) randomized functions. We refer the reader to [GGJS13] for details.

1.3 Organization

The rest of this paper is organized as follows. We start by presenting the formal definitions for functional encryption for randomized functionalities (Section 2). Next, we recall the definitions for various cryptographic primitives used in our construction (Section 3). We then present our construction of functional encryption for randomized functionalities (Section 4) and prove its security in the selective model (Section 5). Finally, we discuss how to extend our results to achieve full security (Section 6).

2 Functional Encryption for Randomized Functions

In this section, we present definitions for functional encryption for randomized functions (or $\mathsf{rand\text{-}FE}$ for short). We start by presenting the syntax for $\mathsf{rand\text{-}FE}$ and then proceed to give the security definitions for the same.

Syntax. Throughout the paper, we denote the security parameter by 1^κ. Let $\mathcal{X} = \{\mathcal{X}_\kappa\}_{\kappa \in \mathbb{N}}$, $\mathcal{R} = \{\mathcal{R}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ be ensembles where each $\mathcal{X}_\kappa$, $\mathcal{R}_\kappa$ and $\mathcal{Y}_\kappa$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_\kappa\}_{\kappa \in \mathbb{N}}$ be an ensemble where each $\mathcal{F}_\kappa$ is a finite collection of randomized functions. Each function $f \in \mathcal{F}_\kappa$ takes as input a string $x \in \mathcal{X}_\kappa$ and randomness $r \in \mathcal{R}_\kappa$ and outputs $f(x; r) \in \mathcal{Y}_\kappa$.

A functional encryption scheme $\mathcal{FE}$ for randomized functions $\mathcal{F}$ consists of four algorithms ($\mathsf{rFE.Setup}$, $\mathsf{rFE.Enc}$, $\mathsf{rFE.Keygen}$, $\mathsf{rFE.Dec}$):

- **Setup** $\mathsf{rFE.Setup}(1^\kappa)$ is a PPT algorithm that takes as input the security parameter κ and outputs the public key MPK and the master secret key MSK.

- **Encryption** $\mathsf{rFE.Enc}(x, \mathsf{MPK})$ is a PPT algorithm that takes as input a message x and the public key MPK and outputs a ciphertext CT.

- **Key Generation** $\mathsf{rFE.Keygen}(f, \mathsf{MSK})$ is a PPT algorithm that takes as input a function $f \in \mathcal{F}$ and the master secret key MSK and outputs a secret key SK_f.
- **Decryption** $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_f)$ is a deterministic algorithm that takes as input a ciphertext CT, the public key MPK and a secret key SK_f and outputs a string $y \in \mathcal{Y}_\kappa$.

Definition 2.1 (Correctness)**.** *A functional encryption scheme $\mathcal{FE}$ for randomized function family $\mathcal{F}$ is* correct *if for every polynomial $n = n(\kappa)$, every $\vec{f} \in \mathcal{F}_\kappa^n$ and every $\vec{x} \in \mathcal{X}_\kappa^n$, the following two distributions are computationally indistinguishable:*

1. **Real:** $\left\{\mathsf{rFE.Dec}\left(\mathsf{CT}_i, \mathsf{SK}_{f_j}\right)\right\}_{i=1,j=1}^{n,n}$*, where:*
   - $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
   - $\mathsf{CT}_i \leftarrow \mathsf{rFE.Enc}(x_i, \mathsf{MPK})$ *for* $i \in [n]$
   - $\mathsf{SK}_{f_j} \leftarrow \mathsf{rFE.Keygen}(f_j, \mathsf{MSK})$ *for* $j \in [n]$
2. **Ideal:** $\{f_j\,(x_i; r_{i,j})\}_{i=1,j=1}^{n,n}$ *where* $r_{i,j} \leftarrow \mathcal{R}_\kappa$

Remark. We note that unlike the case of deterministic functions where it suffices to define correctness for a single ciphertext and a single key, in the case of randomized functions, it is essential to define correctness for *multiple* ciphertexts and functions. To see this, consider the scenario where a secret key SK_f corresponding to a function f is implemented in such a way that it has some "fixed" randomness r hardwired in it. Now, upon decrypting any ciphertext $\mathsf{CT} \leftarrow \mathsf{rFE.Enc}(x, \mathsf{MPK})$ with SK_f, one would obtain the output $f(x; r)$ w.r.t. the *same* randomness r. Note that this clearly incorrect implementation of SK_f would satisfy the correctness definition for a single ciphertext and a single key, but will fail to satisfy our definition given above.

2.1 Security for Functional Encryption

We now present our security definitions for $\mathsf{rand\text{-}FE}$. We first observe that existing security definitions for functional encryption only consider the *malicious receiver* setting, in that they intuitively guarantee that an adversary who owns a secret key SK_f corresponding to a function f cannot learn anymore than $f(x)$ from an encryption of x. In this work, we are also interested in achieving security against *malicious senders*. In particular, we would like to guarantee that an adversarial encryptor cannot force "bad" outputs on an honest receiver.[1] As discussed earlier, this is particularly important when modeling randomized functions.

We consider a a *unified* adversarial model that simultaneously captures malicious receivers and malicious senders. Following prior works, we consider two notions of security, namely, simulation-based security (or SIM-security, in short) and indistinguishability-based security (or IND-security, in short). We present our security definitions for the *selective model*, where the adversary must decide the challenge messages up front, before the system parameters are chosen. (However, they can be modified in a natural manner to full security.) Further, for simplicity of notation, we omit explicit reference to auxiliary input to the adversary from our definitions.

[1] Note that this models an *active* adversary in contrast to the malicious receiver case, where the adversary is *semi-honest.*

Simulation Based Security. We first present a simulation-based security definition for $\mathsf{rand\text{-}FE}$. We extend the existing SIM-security notion for FE to also provide security against adversarial senders. To understand the main idea behind our definition, let us consider an honest receiver who owns a secret key SK_f corresponding to a function f. Then, in order to formalize the intuition that an adversarial sender cannot force "incorrect" outputs on this honest receiver, we allow the adversary to make *decryption queries* for arbitrary ciphertexts[2] w.r.t. the secret key SK_f. In the ideal world, the simulator must be able to"extract" the plaintext x from each decryption query and compute as output $f(x; r)$ for some true randomness r. We then require that the decryption query in the real world yields an indistinguishable output.

We now proceed to give our formal definition. For simplicity, below we define security w.r.t. black-box simulators, although we note that our definition can be easily extended to allow for non-black-box simulation. Our definition is parameterized by q that denotes the number of challenge messages.

Definition 2.2 (SIM-security for $\mathsf{rand\text{-}FE}$)**.** *A functional encryption scheme $\mathcal{FE}$ for the randomized function family $\mathcal{F}$ is said to be q-SIM-secure if there exists a simulator $S = (S_1, S_2, S_3)$ such that for every PPT adversary $A = (A_1, A_2, A_3)$, the outputs of the following two experiments are computationally indistinguishable:*

```
Experiment $\mathsf{REAL}_A^{\mathcal{FE}}(1^\kappa)$:
    $(\vec{x}, st_1) \leftarrow A_1(1^\kappa)$ where $\vec{x} \in \mathcal{X}_\kappa^q$
    $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
    $\mathsf{CT}_i^* \leftarrow \mathsf{rFE.Enc}(x_i, \mathsf{MPK})$ for $i \in [q]$
    $(\{f\}, st_2) \leftarrow A_2(\mathsf{MPK}, \vec{\mathsf{CT}}^*, st_1)$
    $\mathsf{SK}_f \leftarrow \mathsf{rFE.Keygen}(f_j, \mathsf{MSK}) \ \forall \ f \in \{f\}$
    $\alpha \leftarrow A_3^{\mathcal{O}_1(\mathsf{MSK},\cdot),\ \mathcal{O}_2(\{\mathsf{SK}_f\},\cdot)}(st_2)$
    Output $(\vec{x}, \{f\}, \{g\}, \{y\}, \alpha)$

Experiment $\mathsf{IDEAL}_A^{\mathcal{FE}}(1^\kappa)$:
    $(\vec{x}, st_1) \leftarrow A_1(1^\kappa)$ where $\vec{x} \in \mathcal{X}_\kappa^q$
    $(\mathsf{MPK}, \vec{\mathsf{CT}}^*, st') \leftarrow S_1(1^\kappa)$
    $(\{f\}, st_2) \leftarrow A_2(\mathsf{MPK}, \vec{\mathsf{CT}}^*, st_1)$
    $\alpha \leftarrow A_3^{\mathcal{O}'_1(\cdot),\ \mathcal{O}'_2(\{f\},\cdot)}(st_2)$
    Output $(\vec{x}, \{f\}, \{g'\}, \{y'\}, \alpha)$
```

where,

- *In the real experiment, $\mathcal{O}_1(\mathsf{MSK}, \cdot)$ denotes the key generation oracle $\mathsf{rFE.Keygen}(\cdot, \mathsf{MSK})$ and $\mathcal{O}_2(\{\mathsf{SK}_f\}, \cdot)$ denotes a decryption oracle that takes as input ciphertexts CT such that $\mathsf{CT} \neq \mathsf{CT}_i^*$ and returns $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_f)$ for all $f \in \{f\}$. Further, $\{g\}$ denotes the set of key queries made by A_3 and $\{y\}$ denotes the set of responses of $\mathcal{O}_2$ to the decryption queries of A_3.*

- *In the ideal experiment, $\mathcal{O}'_1(\cdot)$ denotes the simulator algorithm $S_2(st', \cdot)$ that has oracle access to the ideal functionality $\mathsf{KeyIdeal}(\vec{x}, \cdot)$. The functionality $\mathsf{KeyIdeal}$ accepts key queries g' and returns $g'(x_i, r_i)$ for every $x_i \in \vec{x}$ and randomly chosen $r_i \in \mathcal{R}_\kappa$. Similarly, $\mathcal{O}'_2(\{f\}, \cdot)$ denotes the simulator algorithm $S_3(st', \cdot)$ that has oracle access to ideal functionality $\mathsf{DecryptIdeal}(\{f\}, \cdot)$. The functionality $\mathsf{DecryptIdeal}$ accepts input queries x and returns $f_i(x; r_i)$ for every $f_i \in \{f\}$, and randomly chosen $r_i \in \mathcal{R}_\kappa$. Further, $\{g'\}$ denotes the set of queries made by S_2 to $\mathsf{KeyIdeal}$ and $\{y'\}$ denotes the set of response of $\mathsf{DecryptIdeal}$ to the queries of S_3.*

[2] This is similar in spirit to the standard chosen-ciphertext security notion for public-key encryption.

Indistinguishability Based Security. We now present an indistinguishability-based security definition for rand-FE. Similar to the SIM-security case, our definition models both corrupted senders and receivers, and extends the existing IND-security notions for functional encryption. Unlike the SIM-security case, here we do not consider any parameter to bound the number of message queries.

Definition 2.3 (IND-secure rand-FE)**.** *A functional encryption scheme $\mathcal{FE}$ is* IND *Secure if for every PPT adversary $A = (A_1, A_2, A_3)$, the distributions $\mathsf{IND}_0^{\mathcal{FE}}(1^\kappa, A)$ and $\mathsf{IND}_1^{\mathcal{FE}}(1^\kappa, A)$ are computationally indistinguishable, where $\mathsf{IND}_b^{\mathcal{FE}}(1^\kappa, A)$ is defined as follows :*

```
Experiment $\mathsf{IND}_A^{\mathcal{FE}}(1^\kappa)$:
    $(\vec{x}_0, \vec{x}_1, st_1) \leftarrow A_1(1^{1^\kappa})$ where $|\vec{x}_0| = |\vec{x}_1|$
    $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
    $\mathsf{CT}^*[i] \leftarrow \mathsf{rFE.Enc}(x_b[i], \mathsf{MPK}) \ \forall \ x_b[i] \in \vec{x}_b$
    $(\{f\}, st_2) \leftarrow A_2(\mathsf{MPK}, \vec{\mathsf{CT}}^*, st_1)$
    $\mathsf{SK}_f \leftarrow \mathsf{rFE.Keygen}(f, \mathsf{MSK}) \ \forall \ f \in \{f\}$
    Output $A_3^{\mathcal{O}_1(\mathsf{MSK},\cdot),\mathcal{O}_2(\{\mathsf{SK}_f\},\cdot)}(st_2)$
```

In the above experiment, $\mathcal{O}_1(\mathsf{MSK}, \cdot)$ denotes the key generation oracle $\mathsf{rFE.Keygen}(\cdot, \mathsf{MSK})$ *and $\mathcal{O}_2(\{\mathsf{SK}_f\}, \cdot)$ denotes a decryption oracle that takes as input ciphertexts* CT *and returns* $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_f)$ *for all $f \in \{f\}$. We require that:*

- *Every key query g made by A_3 to $\mathcal{O}_1$ is such that for every i, the output distributions $g(x_0[i])$ and $g(x_1[i])$ are computationally indistinguishable.*
- *Every decryption query* CT *made by A_3 to $\mathcal{O}_2$ is such that for every i, $\mathsf{CT} \neq \mathsf{CT}^*[i]$.*

3 Preliminaries

In this section, we present definitions for various cryptographic primitives that we shall use in our construction of functional encryption for randomized functions. We assume familiarity with standard semantically secure public-key encryption and strongly unforgeable signature schemes and omit their formal definition from this text. Below, we recall the notions of indistinguishability obfuscation, puncturable pseudorandom functions, non-interactive witness indistinguishable proof systems and perfectly binding commitment schemes.

3.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation that was defined by Barak et al. [BGI+01]. Intuitively speaking, we require that for any two circuits C_1 and C_2 that are "functionally equivalent" (i.e., for all inputs x in the domain, $C_1(x) = C_2(x)$), the obfuscation of C_1 must be computationally indistinguishable from the obfuscation of C_2. Below we present the formal definition following the syntax of [GGH+13].

Definition 3.1. *(Indistinguishability Obfuscation) A uniform PPT machine $i\mathcal{O}$ is called an indistinguishability obfuscator for a circuit class $\{\mathcal{C}_\kappa\}$ if the following holds:*

- **Correctness:** *For every $\kappa \in \mathbb{N}$, every $C \in \mathcal{C}_\kappa$, every input x in the domain of C, we have that*
$$Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(C)] = 1$$

- **Indistinguishability:** *For every $\kappa \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_\kappa$, if $C_0(x) = C_1(x)$ for all inputs x, then for all PPT adversaries $\mathcal{A}$, we have:*
$$|Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(\kappa)$$

Very recently, Garg et al. [GGH+13] gave the first candidate construction for an indistinguishability obfuscator $i\mathcal{O}$ for the circuit class $P/poly$.

3.2 Puncturable Pseudorandom Functions

Puncturable family of PRFs are a special case of constrained PRFs [BW13, BGI13, KPTZ13], where the PRF is defined on all input strings except for a set of size polynomial in the security parameter. Below we recall their definition, as given by [SW13].

Syntax A *puncturable* family of PRFs is defined by a tuple of algorithms (Key, Eval, Puncture) and a pair of polynomials $n(\cdot)$ and $m(\cdot)$:

- **Key Generation** $\mathsf{Key}(1^\kappa)$ is a PPT algorithm that takes as input the security parameter κ and outputs a PRF key K
- **Punctured Key Generation** $\mathsf{Puncture}(K, S)$ is a PPT algorithm that takes as input a PRF key K, a set $S \subset \{0,1\}^{n(\kappa)}$ and outputs a punctured key K_S
- **Evaluation** $\mathsf{Eval}(K, x)$ is a deterministic algorithm that takes as input a key K (punctured key or PRF key), a string $x \in \{0,1\}^{n(\kappa)}$ and outputs $y \in \{0,1\}^{m(\kappa)}$

Definition 3.2. *A family of PRFs* Key*,* Eval*,* Puncture *is* puncturable *if it satisfies the following properties :*

- **Functionality preserved under puncturing.** *Let $K \leftarrow \mathsf{Key}(1^\kappa)$, $K_S \leftarrow \mathsf{Puncture}(K, S)$. Then, for all $x \notin S$, $\mathsf{Eval}(K, x) = \mathsf{Eval}(K_S, x)$.*
- **Pseudorandom at punctured points.** *For every PPT adversary (A_1, A_2) such that $A_1(1^\kappa)$ outputs a set $S \subset \{0,1\}^{n(\kappa)}$ and $x \in S$, consider an experiment where $K \leftarrow \mathsf{Key}(1^\kappa)$ and $K_S \leftarrow \mathsf{Puncture}(K, S)$. Then*
$$\Big|Pr[A_2(K_S, x, \mathsf{Eval}(K, x)) = 1] - Pr[A_2(K_S, x, U_{m(\kappa)}) = 1]\Big| \leq \mathsf{negl}(\kappa)$$

where U_ℓ denotes the uniform distribution over ℓ bits.

As observed by [KPTZ13, BW13, BGI13], the [GGM86] construction of PRFs from one-way functions easily yield puncturable PRFs.

Theorem 3.3 ([GGM86, KPTZ13, BW13, BGI13])**.** *If one-way functions exist, then for all polynomials $n(\kappa)$ and $m(\kappa)$, there exists a puncturable PRF family that maps $n(\kappa)$ bits to $m(\kappa)$ bits.*

Remark. We note that in the above construction, the size of the punctured key K_S grows linearly with the size of the puncture set S.

3.3 Non-Interactive Witness Indistinguishable Proofs

In this section, we present the definition for non-interactive witness-indistinguishable (NIWI) proofs. We emphasize that we are interested in *proof* systems, i.e., where the soundness guarantee holds against computationally unbounded cheating provers.

Syntax. Let R be an efficiently computable relation that consists of pairs (x, w), where x is called the statement and w is the witness. Let L denote the language consisting of statements in R. A non-interactive proof system for a language L consists of a setup algorithm $\mathsf{NIWI.Setup}$, a prover algorithm $\mathsf{NIWI.Prove}$ and a verifier algorithm $\mathsf{NIWI.Verify}$, defined as follows:

- **Setup** $\mathsf{NIWI.Setup}(1^\kappa)$ is a PPT algorithm that takes as input the security parameter 1^κ and outputs a common reference string crs.
- **Prover** $\mathsf{NIWI.Prove}(\mathsf{crs}, x, w)$ is a PPT algorithm that takes as input the common reference string crs, a statement x along with a witness w. $(x, w) \in R$; if so, it produces a proof string π, else it outputs `fail`.
- **Verifier** $\mathsf{NIWI.Verify}(\mathsf{crs}, x, \pi)$ is a PPT algorithm that takes as input the common reference string crs and a statement x with a corresponding proof π. It outputs 1 if the proof is valid, and 0 otherwise.

Definition 3.4 (NIWI)**.** *A* non-interactive witness-indistinguishable proof system *for a language L with a PPT relation R is a tuple of algorithms* $(\mathsf{NIWI.Setup}, \mathsf{NIWI.Prove}, \mathsf{NIWI.Verify})$ *such that the following properties hold:*

- **Perfect Completeness:** *For every $(x, w) \in R$, it holds that*

$$\Pr[\mathsf{NIWI.Verify}(\mathsf{crs}, x, \mathsf{NIWI.Prove}(\mathsf{crs}, x, w)) = 1] = 1$$

  where $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(1^\kappa)$*, and the probability is taken over the coins of* $\mathsf{NIWI.Setup}$*,* $\mathsf{NIWI.Prove}$ *and* $\mathsf{NIWI.Verify}$*.*

- **Statistical Soundness:** *For every adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathsf{NIWI.Verify}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L \mid \mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(1^\kappa); (x, \pi) \leftarrow \mathcal{A}(\mathsf{crs})] = \mathsf{negl}(1^\kappa)$$

- **Witness Indistinguishability:** *For any triplet (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$, the distributions* $\{\mathsf{crs}, \mathsf{NIWI.Prove}(\mathsf{crs}, x, w_0)\}$ *and* $\{\mathsf{crs}, \mathsf{NIWI.Prove}(\mathsf{crs}, x, w_1)\}$ *are computationally indistinguishable, where* $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(1^\kappa)$*.*

Recently, it was shown by Sahai and Waters [SW13] that NIWI proofs can be constructed from indistinguishability obfuscation and one-way functions.

3.4 Commitment Schemes

A commitment scheme Com is a PPT algorithm that takes as input a string x and outputs $C \leftarrow \mathsf{Com}(x)$. A perfectly binding commitment scheme must satisfy the *perfect binding* and *computational hiding* properties :

- **Perfectly Binding** : This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2, s_1, s_2$ $\mathsf{Com}(x_1; s_1) \neq \mathsf{Com}(x_2; s_2)$
- **Computational Hiding** For all strings x_0 and x_1 (of the same length), for all PPT adversaries $\mathcal{A}$, we have that :

$$|Pr[\mathcal{A}(\mathsf{Com}(x_0)) = 1] - Pr[\mathcal{A}(\mathsf{Com}(x_1) = 1)]| \leq \mathsf{negl}(\kappa)$$

Remark. We note that it is in fact sufficient to use a standard 2-round statistically binding scheme in our construction in Section 4. Note that such a commitment scheme can be based on one way functions. For simplicity of exposition, however, we will present our construction using a non-interactive perfectly binding scheme.

4 Our Construction

Let $\mathcal{F}$ denote the family of all PPT functions. We now present a functional encryption scheme $\mathcal{FE}$ for $\mathcal{F}$. Our scheme provides the following security guarantees:

1. For any a priori bounded $q = \mathsf{poly}(\kappa)$, $\mathcal{FE}$ is q-SIM-secure in the selective model. In this case, the size of the secret keys in $\mathcal{FE}$ grows linearly with q. It follows from [BSW11, BO13] that such a dependence on q is, in fact, *necessary.*
2. $\mathcal{FE}$ is $\mathsf{poly}(\kappa)$-IND-secure in the selective model for any *unbounded* number of message queries q. In this case, the size of the secret keys is independent of q.

Note that $\mathsf{poly}(\kappa)$-IND-security follows from 1-SIM-security by a simple hybrid argument. Therefore, it suffices to prove SIM security. For simplicity of exposition, we will in fact only consider 1-SIM-security. We remark that our construction and proof easily extends to q-SIM-security where q is a fixed $\mathsf{poly}(\kappa)$. Later, in Section 6, we discuss how to achieve full security (as opposed to selective security) for our scheme.

Notation. Let ($\mathsf{NIWI.Setup}$, $\mathsf{NIWI.Prove}$, $\mathsf{NIWI.Verify}$) be a NIWI proof system. Let Com be a perfectly binding commitment scheme. Let $i\mathcal{O}$ be an indistinguishability obfuscator for all efficiently computable circuits. Let (Key, $\mathsf{Puncture}$, Eval) be a puncturable family of PRF. Let (Gen, Sign, Verify) be a strongly unforgeable one-time signature scheme. Finally, let ($\mathsf{PKE.Setup}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$) be a semantically secure public-key encryption scheme.

Let $\mathsf{c\text{-}len} = \mathsf{c\text{-}len}(1^\kappa)$ denote the length of ciphertexts in ($\mathsf{PKE.Setup}$, $\mathsf{PKE.Enc}$, $\mathsf{PKE.Dec}$) . Let $\mathsf{v\text{-}len} = \mathsf{v\text{-}len}(1^\kappa)$ denote the length of verification keys in (Gen, Sign, Verify). We shall use a parameter $\mathsf{len} = 2 \cdot \mathsf{c\text{-}len} + \mathsf{v\text{-}len}$ in the description of our scheme.

We now proceed to describe our scheme $\mathcal{FE} = (\mathsf{rFE.Setup}, \mathsf{rFE.Enc}, \mathsf{rFE.Keygen}, \mathsf{rFE.Dec})$.

Setup $\mathsf{rFE.Setup}(1^\kappa)$**:** The setup algorithm first computes a CRS $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}$ for the NIWI proof system. Next, it computes two key pairs – $(PK_1, SK_1) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$, $(PK_2, SK_2) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$ – of the public-key encryption scheme. Finally, it computes a commitment $C \leftarrow \mathsf{Com}(0^{\mathsf{len}})$.

The public key $\mathsf{MPK} = (\mathsf{crs}, PK_1, PK_2, C)$ and the master secret key $\mathsf{MSK} = SK_1$. The algorithm outputs $(\mathsf{MPK}, \mathsf{MSK})$.

Encryption $\mathsf{rFE.Enc}(x, \mathsf{MPK})$**:** To encrypt a message x, the encryption algorithm first generates a key pair $(sk, vk) \leftarrow \mathsf{Gen}(1^\kappa)$ of the one-time signature scheme. It then computes ciphertexts $c_1 \leftarrow \mathsf{PKE.Enc}(x, PK_1; r_1)$ and $c_2 \leftarrow \mathsf{PKE.Enc}(x, PK_2; r_2)$. Next, it computes a NIWI proof $\pi \leftarrow \mathsf{NIWI.Prove}(\mathsf{crs}, z, w)$ for the NP statement $z = (z_1 \vee z_2)$ where z_1 and z_2 are defined as follows:

$$z_1 := (\exists x, s_1, s_2 \text{ such that } c_1 = \mathsf{PKE.Enc}(x, PK_1; s_1) \wedge c_2 = \mathsf{PKE.Enc}(x, PK_2; s_2)) \tag{1}$$

$$z_2 := (\exists s \text{ such that } C = \mathsf{Com}(c_1 \| c_2 \| vk, s) \tag{2}$$

A witness $w_{\mathsf{real}} = (x, s_1, s_2)$ for z_1 is referred to as the *real* witness, while a witness $w_{\mathsf{trap}} = s$ for z_2 is referred to as the *trapdoor* witness.

The honest encryption algorithm uses the real witness w_{real} to compute π. Finally, it computes a signature $\sigma \leftarrow \mathsf{Sign}(c_1 \| c_2 \| \pi, sk)$ on the string $c_1 \| c_2 \| \pi$ using sk. The output of the algorithm is the ciphertext $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$.

Key Generation $\mathsf{rFE.Keygen}(f, \mathsf{MSK})$**:** On input f ,the key generation algorithm first chooses a fresh PRF key $K \leftarrow \mathsf{Key}(1^\kappa)$. It then computes the secret key $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathcal{G}_f)$ where the function $\mathcal{G}_f$ is described in Figure 1. Note that $\mathcal{G}_f$ has the public key MPK, the master secret key MSK and the PRF key K hardwired in it.

```
$\mathcal{G}_f(\mathsf{CT})$

1. Parse $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$.
2. If $\mathsf{Verify}(\sigma, c_1 \| c_2 \| \pi, vk)$, then output $\perp$ and stop, otherwise continue to the next step.
3. If $\mathsf{NIWI.Verify}(\mathsf{crs}, z, \pi) = 0$, then output $\perp$ and stop, otherwise continue to the next step. Here
   $z = (c_1, c_2, vk, PK_1, PK_2, C)$ is the statement corresponding to $\pi$.
4. Compute $x \leftarrow \mathsf{PKE.Dec}(c_1, SK_1)$.
5. Compute $r \leftarrow \mathsf{Eval}(K, c_1 \| c_2 \| vk)$.
6. Output $y = f(x; r)$.
```

Figure 1: Functionality $\mathcal{G}_f$

The algorithm outputs SK_f as the secret key corresponding to f.

Size of Function $\mathcal{G}_f$. In order to prove that $\mathcal{FE}$ is q-SIM-secure, we require the function $\mathcal{G}_f$ to be padded with zeros such that $|\mathcal{G}_f| = |\mathsf{Sim}.\mathcal{G}_f|$, where the "simulated" functionality $\mathsf{Sim}.\mathcal{G}_f$ is described later in Figure 2. In this case, the size of SK_f grows linearly with q.

Note, however, that such a padding is *not* necessary to prove $\mathsf{poly}(\kappa)$-IND-security for $\mathcal{FE}$. Indeed, in this case, the size of the secret keys SK_f is independent of the number of message queries made by the adversary.

Decryption $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_f)$: On input CT, the decryption algorithm computes and outputs $\mathsf{SK}_f(\mathsf{CT})$.

This completes the description of $\mathcal{FE}$. We prove the correctness of $\mathcal{FE}$ in Appendix B.

Theorem 4.1. *Assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme $\mathcal{FE}$ is* 1-SIM*-secure in the selective model.*

5 Proof of Security

We now prove that the proposed scheme $\mathcal{FE}$ is 1-SIM-secure. As already mentioned, our security proof easily extends to q-SIM-security, for any a priori fixed $q = \mathsf{poly}(\kappa)$.

We first construct an ideal world adversary aka simulator S in Section 5.1. Next, in Section 5.2, we prove indistinguishability of the outputs of the real and ideal world experiments via a hybrid argument.

5.1 Description of Simulator

We describe a simulator $S = (S_1, S_2, S_3)$ that makes black-box use of a real world adversary $A = (A_1, A_2, A_3)$.

Algorithm S_1. S_1 first performs a simulated setup procedure. Namely, it first computes a CRS $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}(1^\kappa)$ for the NIWI proof system and two key pairs – $(PK_1, SK_1) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$ and $(PK_2, SK_2) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$ – for the public-key encryption scheme. Next, it chooses a key pair for the signature scheme - $(sk^*, vk^*) \leftarrow \mathsf{Gen}(1^\kappa)$. Then, it computes the commitment C in the following manner:

- First compute $c_1^* \leftarrow \mathsf{PKE.Enc}(\vec{0}, PK_1)$ and $c_2^* \leftarrow \mathsf{PKE.Enc}(\vec{0}, PK_2)$.
- Next, compute $C \leftarrow \mathsf{Com}(c_1^* \| c_2^* \| vk^*)$. Let s denote the randomness used to compute C.

S_1 constructs a proof π^* by using the *trapdoor* witness s, i.e., $\pi^* \leftarrow \mathsf{NIWI.Prove}(\mathsf{crs}, y, s)$, where the statement $y = (c_1^*, c_2^*, vk^*, PK_1, PK_2, C)$. Finally, it computes a signature $\sigma^* \leftarrow \mathsf{Sign}(c_1^* \| c_2^* \| \pi^*, sk^*)$. It sets $\mathsf{MPK} = (\mathsf{crs}, PK_1, PK_2, C)$ and the challenge ciphertext $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$.

Algorithm S_2. S_2 simulates the key generation oracle for adversary A_3. Whenever A_3 makes a key query for a function f, S_2 performs the following sequence of steps:

1. Query the ideal functionality $\mathsf{KeyIdeal}$ on input f. Let y be the output of $\mathsf{KeyIdeal}$.
2. Compute a PRF key $K \leftarrow \mathsf{Key}(1^\kappa)$ and then compute a punctured key $K' \leftarrow \mathsf{Puncture}(K, c_1^* \| c_2^* \| vk^*)$.
3. Compute the secret key $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$ where the functionality $\mathsf{Sim}.\mathcal{G}_f$ is described in Figure 2. $\mathsf{Sim}.\mathcal{G}_f$ has the public key MPK, master secret key MSK, the punctured key K', the challenge ciphertext CT^* and the output value y hardwired in it.

4. Return SK_f to A_3.

$\mathsf{Sim}.\mathcal{G}_f(\mathsf{CT})$

1. Parse $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$.
2. If $\mathsf{Verify}(\sigma, c_1 \| c_2 \| \pi, vk)$, then output $\perp$ and stop, otherwise continue to the next step.
3. If $\mathsf{NIWI.Verify}(\mathsf{crs}, z, \pi) = 0$, then output $\perp$ and stop, otherwise continue to the next step. Here $z = (c_1, c_2, vk, PK_1, PK_2, C)$ is the statement corresponding to π.
4. If $(c_1 \| c_2 \| vk = c_1^* \| c_2^* \| vk^*)$ output y and stop.
5. Compute $x \leftarrow \mathsf{PKE.Dec}(c_1, SK_1)$.
6. Compute $r \leftarrow \mathsf{Eval}(K', c_1 \| c_2 \| vk)$.
7. Output $y = f(x; r)$.

Figure 2: Functionality $\mathsf{Sim}.\mathcal{G}_f$

Algorithm S_3. S_3 simulates the decryption oracle for the adversary A_3. Let $\{f\}$ denote the challenge functions chosen earlier by A_2. Now, whenever A_3 makes a decryption query $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, S_3 performs the following sequence of steps:

1. If $\mathsf{Verify}(\sigma, c_1 \| c_2 \| \pi, vk)$, then output $\perp$ and stop, otherwise continue to the next step.
2. If $\mathsf{NIWI.Verify}(\mathsf{crs}, z, \pi) = 0$, then output $\perp$ and stop, otherwise continue to the next step. Here $z = (c_1, c_2, vk, PK_1, PK_2, C)$ is the statement corresponding to π.
3. Compute $x \leftarrow \mathsf{PKE.Dec}(c_1, SK_1)$.
4. Query $\mathsf{DecryptIdeal}$ with input x. Let $\vec{y}$ denote the set of values output by $\mathsf{DecryptIdeal}$.
5. Return $\vec{y}$ to A_3.

Remark. Here we assume that the adversary does not make the same decryption query twice. Note that when the adversary makes the same decryption query more than once, then the simulator can simply "reuse" a previously computed output. For convenience of notation, we omit this from our description.

5.2 Indistinguishability of the Outputs

We now describe a series of hybrid experiments $\mathsf{H}_0, \ldots, \mathsf{H}_{11}$, where H_0 corresponds to the real world and H_{11} corresponds to the ideal world experiment. In Appendix A, we prove that for every i, the output of H_i is computationally indistinguishable from the output of H_{i+1}.

Hybrid H_0: This is the real experiment.

Hybrid H_1: This experiment is the same as H_0 except in the manner in which the key queries of the adversary are answered. Let $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$ denote the challenge ciphertext. Whenever the adversary makes a key query f, we perform the following steps:

1. Compute a PRF key $K \leftarrow \mathsf{Key}(1^\kappa)$ and then compute a punctured key $K' \leftarrow \mathsf{Puncture}(K, c_1^* \| c_2^* \| vk^*)$.
2. Compute $r \leftarrow \mathsf{Eval}(K, c_1^* \| c_2^* \| vk^*)$ and $y = f(x; r)$.
3. Compute the secret key $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$ where the functionality $\mathsf{Sim}.\mathcal{G}_f$ is described in Figure 2. Note that $\mathsf{Sim}.\mathcal{G}_f$ has the public key MPK, master secret key MSK, the punctured key K', the challenge ciphertext components c_1^*, c_2^*, vk^* and the output value y (as computed above) hardwired in it.
4. Return SK_f to A_3.

Hybrid H_2: This experiment is the same as H_1, except that we now answer the key queries of A_3 in the same manner as the simulator S_2.

Hybrid H_3: This experiment is the same as H_2, except that the setup algorithm computes the commitment C in the following manner: let $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$ denote the challenge ciphertext. Then, $C \leftarrow \mathsf{Com}(c_1^* \| c_2^* \| vk^*)$.

Hybrid H_4: This experiment is the same as H_3, except that we modify the challenge ciphertext $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$. Specifically, the proof string π^* is now computed using the *trapdoor* witness s where s is the randomness used to compute the commitment C.

Hybrid H_5: This experiment is the same as H_4, except that in the challenge ciphertext $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$, the *second* ciphertext c_2^* is an encryption of zeros, i.e., $c_2^* \leftarrow \mathsf{PKE.Enc}(\vec{0}, PK_2)$.

Hybrid H_6: This experiment is the same as H_5, except that for every key query f, the secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim}.\mathcal{G}'_f)$ where $\mathsf{Sim}.\mathcal{G}'_f$ is the same as function $\mathsf{Sim}.\mathcal{G}_f$ except that:

1. It has secret key SK_2 hardwired instead of SK_1.
2. It decrypts the *second* component of each input ciphertext using SK_2. More concretely, in Step 5 of $\mathsf{Sim}.\mathcal{G}'_f$, plaintext x is computed as $x \leftarrow \mathsf{PKE.Dec}(c_2, SK_2)$.

Hybrid H_7: This experiment is the same as H_6, except that we modify the manner in which the decryption queries of A_3 are answered. For every decryption query $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, we decrypt c_2 using SK_2; that is, in Step 3, $x \leftarrow \mathsf{PKE.Dec}(c_2, SK_2)$.

Hybrid H_8: This experiment is the same as H_7, except that in the challenge ciphertext $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$, the *first* ciphertext c_1^* is an encryption of zeros, i.e., $c_1^* \leftarrow \mathsf{PKE.Enc}(\vec{0}, PK_1)$.

Hybrid H_9: This experiment is the same as H_8, except that we modify the manner in which the decryption queries of A_3 are answered. For every decryption query $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, we decrypt c_1 using SK_1; that is, in Step 3, $x = \mathsf{PKE.Dec}(c_1, SK_1)$.

Hybrid H_{10}: This experiment is the same as H_9, except that we change the manner in which the key queries are answered. For every key query f, the secret key SK_f is computed as $\mathsf{SK}_f \leftarrow i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$.

Hybrid H_{11}: This experiment is the same as H_{10}, except that we now answer the decryption queries of A_3 in the same manner as the simulator algorithm S_3. That is, on receiving a decryption query $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, we first decrypt c_1 using SK_1 (as in the previous hybrid) to obtain $x \leftarrow \mathsf{PKE.Dec}(c_1, SK_1)$ and then query the ideal functionality $\mathsf{DecryptIdeal}$ with input x to receive $\vec{y}$. We return $\vec{y}$ to A_3. Note that this is the ideal experiment.

This completes the description of the hybrid experiments. We prove their indistinguishability in Appendix A.

6 From Selective to Full Security

Our results can be extended to achieve full security by using either of the following two approaches:

1. We can use complexity leveraging to prove full security of the proposed scheme $\mathcal{FE}$. Specifically, by assuming indistinguishability obfuscation and one-way functions that are subexponentially secure, we can simply guess the challenge messages of the adversary and then prove security in the same manner as in Section 5. (To be more concrete, we would require the indistinguishability obfuscation and one-way function to be secure against adversaries running in time $\mathcal{O}(2^M)$, where M is the total length of the challenge message vectors.)

2. Alternatively, following [BCP13, ABG$^+$13], we can use a modified construction based on *differing-inputs obfuscation* [BGI$^+$01] and simulation-sound non-interactive zero knowledge [Sah99] to directly achieve full security in the standard model. We remark that the usage of differing-inputs obfuscation in the context of functional encryption was recently explored by Boyle et al. [BCP13] and Ananth et al [ABG$^+$13]. Here, we use the specific approach of [ABG$^+$13] to achieve full security in our setting of randomized functionalities.

   Very briefly, consider a scheme $\mathcal{FE}'$ which is the same as $\mathcal{FE}$, except for the following modifications:

   - In the setup algorithm, we now choose a CRS for a simulation-sound NIZK (as opposed to NIWI). Further, we do *not* need to compute the commitment C anymore.
   - To encrypt a message x, we compute ciphertexts c_1 and c_2 as before, but now we simply prove (via a simulation-sound NIZK) that c_1 and c_2 are encryptions of the same message. The rest of the encryption algorithm is the same as before.
   - Finally, a secret key SK_f for a function f is computed as a differing-inputs obfuscation (as opposed to an indistinguishability obfuscation) of the functionality $\mathcal{G}_f$ (defined in the same manner as earlier).

   The security of $\mathcal{FE}'$ is proven similarly to $\mathcal{FE}$ with the following key modifications. Intuitively, now to compute the challenge ciphertexts, the simulator will run the simulator of the simulation-sound NIZK to compute simulated proofs. Further, in the key hybrid experiment H_6 where we modify the secret keys SK_f to use SK_2 instead of SK_1, we no longer need to argue that $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ are functionally equivalent. Instead, we only need to argue that they "appear" to be functionally equivalent to a computationally bounded adversary. In particular, from the security of differing-inputs obfuscation, we can extract from an adversary that distinguishes between H_5 and H_6, an input ciphertext CT such that $\mathsf{Sim}.\mathcal{G}_f(\mathsf{CT}) \neq \mathsf{Sim}.\mathcal{G}'_f(\mathsf{CT}')$. Now, from the definition of $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$, it follows that CT must be different from the challenge ciphertext and must contain c_1 and c_2 that are encryptions of *different* messages. Thus, π in CT is an accepting proof for a false statement, which can be used to contradict simulation-soundness of the NIZK.

7 Acknowledgements

We thank Ran Canetti, Shafi Goldwasser, Brent Waters and Xiang Xe for useful discussions.

References

[ABG$^+$13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013:689, 2013.

[BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. *IACR Cryptology ePrint Archive*, 2013:650, 2013.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *Lecture Notes in Computer Science*, pages 1–18. Springer-Verlag, 2001.

[BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.

[BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. 2013.

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

[BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

[CIJ$^+$13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO (2)*, 2013.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.

[DNR^{+}09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

[GGH^{+}13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.

[GGJS13] Shafi Goldwasser, Vipul Goyal, Abhishek Jain, and Amit Sahai. Multi-input functional encryption. *IACR Cryptology ePrint Archive*, 2013, 2013.

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

[GKP^{+}13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.

[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*, 2006.

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

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology ePrint Archive*, 2013:379, 2013.

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010, 2010.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.

[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*, 2005.

[SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013, 2013.

A Completing the Security Proof

Here we prove that for every i, the outputs of experiments H_i and H_{i+1} (as described in Section 5.2) are computationally indistinguishable.

Lemma A.1. *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, hybrid experiments H_0 and H_1 are computationally indistinguishable.*

Proof. Note that the only difference in H_0 and H_1 is that in the former experiment, we output $i\mathcal{O}(\mathcal{G}_f)$ as the key corresponding to any key query f, while in the latter experiment, we output $i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$. In order to prove that these two hybrids are computationally indistinguishable, we show that for every key query f, $\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}_f$ have identical input-output behavior. Then, by security of indistinguishability obfuscation, we would have that $i\mathcal{O}(\mathcal{G}_f)$ and $i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$ are computationally indistinguishable, which in turn would imply H_0 and H_1 are computationally indistinguishable.

Observation A.2. *For any input $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, $\mathcal{G}_f$ outputs $\perp$ if and only if $\mathsf{Sim}.\mathcal{G}_f$ outputs $\perp$.*

Note that both $\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}_f$ output $\perp$ if and only if either the signature σ does not verify or the proof π does not verify; that is, either $\mathsf{Verify}(\sigma, c_1\|c_2\|\pi, vk) = 0$ or $\mathsf{NIWI.Verify}(\mathsf{crs}, y, \pi) = 0$ where $y = (c_1, c_2, vk, PK_1, PK_2, C)$. Let us call an input $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$ *valid* if both the signature σ and proof π verify. Next, we prove that both $\mathcal{G}_f$ in H_0 and $\mathsf{Sim}.\mathcal{G}_f$ in H_1 have the same functionality for all valid inputs.

Claim A.3. *For any valid input $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, $\mathcal{G}_f(\mathsf{CT}) = \mathsf{Sim}.\mathcal{G}_f(\mathsf{CT})$.*

Proof. We consider two cases : $c_1\|c_2\|vk \neq c_1^*\|c_2^*\|vk^*$ and $c_1\|c_2\|vk = c_1^*\|c_2^*\|vk^*$. For the first case, note that by the first property of constrained PRF, it follows that $\mathsf{Eval}(K, c_1\|c_2\|vk) = \mathsf{Eval}(K', c_1\|c_2\|vk) = r$. Both $\mathcal{G}_f$ in H_0 and $\mathsf{Sim}.\mathcal{G}_f$ in H_1 decrypt c_1 using SK_1 to compute x, and then output $f(x, r)$.

In the second case, $\mathcal{G}_f$ computes $r \leftarrow \mathsf{Eval}(K, c_1^*\|c_2^*\|vk^*)$, and then computes x by decrypting c_1 and outputs $y' = f(x; r)$. On the other hand, $\mathsf{Sim}.\mathcal{G}_f$ simply outputs the hard-wired value y when $c_1\|c_2\|vk = c_1^*\|c_2^*\|vk^*$. However, note that the value y is computed identically to y', thereby ensuring that $\mathcal{G}_f(\mathsf{CT}^*) = \mathsf{Sim}.\mathcal{G}_f(\mathsf{CT}^*)$. □

Using the above claims, we can now describe our reduction. Assume A_3 makes ℓ queries. We define hybrids $\mathsf{H}_{0,i}$, $0 \leq i \leq \ell$, as follows: in $\mathsf{H}_{0,i}$, we respond to the first $\ell - i$ queries using $\mathsf{FE.Keygen}$ as in H_0, and respond to the last i queries as in H_1.

Claim A.4. *If $\exists$ a PPT distinguisher $\mathcal{A}$ that can distinguish the outputs of $\mathsf{H}_{0,i}$ and $\mathsf{H}_{0,i+1}$ with non negligible advantage, then there exists a PPT adversary $\mathcal{B}$ that can break the security of $i\mathcal{O}$ with non-negligible advantage.*

Let $\mathcal{C}$ be the challenger for obfuscation. Adversary $\mathcal{B}$ works as follows:

1. It first honestly computes $(\mathsf{MPK}, st', \mathsf{CT}^*)$.
2. For the first $(\ell - i - 1)$ key queries f by A_3, $\mathcal{B}$ computes the key for f using $\mathsf{rFE.Keygen}(\cdot, \mathsf{MSK})$. For the last i key queries f by A_3, $\mathcal{B}$ computes the key for f as in H_1.

3. For the $(\ell - i)$'th key query for function f, $\mathcal{B}$ chooses a PRF key K, computes $K' \leftarrow \mathsf{Puncture}(K, c_1^* \| c_2^* \| vk^*)$ and $y = f(x; \mathsf{Eval}(K, c_1^* \| c_2^* \| vk^*))$. It then defines programs $\mathcal{G}_f, \mathsf{Sim}.\mathcal{G}_f$ and sends them to $\mathcal{C}$, and receives an obfuscation SK_f, which it passes on to A.
4. $\mathcal{B}$ runs the rest of the experiment in the same manner as in H_0 and H_1.
5. Finally, $\mathcal{B}$ sends the output of the experiment to $\mathcal{A}$ and returns its output to $\mathcal{C}$.

Now, if $\mathcal{C}$ returns obfuscation of $\mathcal{G}_f$, then $\mathcal{B}$ perfectly simulates experiment $\mathsf{H}_{0,i}$, else it simulates experiment $\mathsf{H}_{0,i+1}$. Thus, if $\mathcal{A}$ distinguishes the outputs with non negligible advantage, then clearly $\mathcal{B}$ breaks the security of indistinguishability obfuscation with non negligible advantage. □

Lemma A.5. *Assuming (*$\mathsf{Key}, \mathsf{Eval}, \mathsf{Puncture}$*) is a puncturable family of PRFs, hybrid experiments* H_1 *and* H_2 *are computationally indistinguishable.*

Proof. Assume A_3 makes ℓ key queries. We consider ℓ intermediate hybrids $\mathsf{H}_{1,i}$ for $0 \leq i \leq \ell$ where in $\mathsf{H}_{1,i}$, we respond to the first $\ell - i$ key queries of A_3 as in H_1, and the remaining i key queries as in H_2. We show that if there exists a PPT distinguisher $\mathcal{A}$ that can distinguish the outputs of $\mathsf{H}_{1,i}$ and $\mathsf{H}_{1,i+1}$ with non-negligible advantage, then there exists a PPT adversary $\mathcal{B}$ that can break the security of puncturable PRFs with non-negligible advantage. The construction of $\mathcal{B}$ is as follows :

1. $\mathcal{B}$ first computes $\mathsf{MPK}, \mathsf{MSK}, \mathsf{CT}^*$ honestly.
2. For the first $(\ell - i - 1)$ key queries from A_3, $\mathcal{B}$ responds in the same manner as in H_1. For the last i key queries, $\mathcal{B}$ responds as in H_2.
3. For the $(\ell - i)$'th key query f, $\mathcal{B}$ first sends $(c_1^* \| c_2^* \| vk^*)$ to the challenger $\mathcal{C}$ and receives (K', r), where $K' = \mathsf{Puncture}(K, c_1^* \| c_2^* \| vk^*)$ for some PRF key K and r is either $\mathsf{Eval}(K, c_1^* \| c_2^* \| vk^*)$ or a uniformly random string in $\mathcal{R}_\kappa$. It then defines the function $\mathsf{Sim}.\mathcal{G}_f$ as before. $\mathcal{B}$ sends $i\mathcal{O}(\mathsf{Sim}.\mathcal{G}_f)$ as the key for function f.
4. $\mathcal{B}$ runs the rest of the experiment in the same manner as in H_1 and H_2.
5. Finally, $\mathcal{B}$ sends the output of the experiment to $\mathcal{A}$ and returns its output to $\mathcal{C}$.

Note that if r was computed as $\mathsf{Eval}(K, c_1^* \| c_2^* \| vk^*)$, then $\mathcal{B}$ perfectly simulates experiment $\mathsf{H}_{1,i}$, else it simulates $\mathsf{H}_{1,i+1}$. Thus, if $\mathcal{A}$ can distinguish the outputs of $\mathsf{H}_{1,i}$ and $\mathsf{H}_{1,i+1}$ with non-negligible advantage, then $\mathcal{B}$ can break security of puncturable PRFs with non-negligible advantage. □

Lemma A.6. *Assuming* Com *is a computationally hiding commitment scheme, hybrid experiments* H_2 *and* H_3 *are computationally indistinguishable.*

Proof. Note that the only difference between experiments H_2 and H_3 is that C is computed as a commitment to 0^{len} in the former case and $(c_1^* \| c_2^* \| vk^*)$ in the latter. Then, assume that $\exists$ PPT distinguisher $\mathcal{A}$ that can distinguish the outputs of H_2 and H_3 with non-negligible advantage. Using $\mathcal{A}$, we can construct a PPT algorithm $\mathcal{B}$ that breaks the computational hiding property of Com as follows:

1. $\mathcal{B}$ first runs A_1 to obtain x. It then computes $(PK_1, SK_1) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$, $(PK_2, SK_2) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$, $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}$ and $(sk^*, vk^*) \leftarrow \mathsf{Gen}(1^\kappa)$.
2. Next, it computes $c_1^* \leftarrow \mathsf{PKE.Enc}(x, PK_1)$, $c_2^* \leftarrow \mathsf{PKE.Enc}(x, PK_2)$ and constructs a valid proof π^* using the real witness. Then it signs $c_1^* \| c_2^* \| \pi^*$ using sk^* to compute σ^*. It sets $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$

3. $\mathcal{B}$ sends 0^{len} and $(c_1^*\|c_2^*\|vk^*)$ to $\mathcal{C}$, and receives C, which is either a commitment to 0^{len} or $(c_1^*\|c_2^*\|vk^*)$.
4. $\mathcal{B}$ simulates the rest of the experiment as in H_2 and H_3.
5. Finally, $\mathcal{B}$ sends the output of the experiment to $\mathcal{A}$ and returns its output to $\mathcal{C}$.

Now, if C is a commitment to 0^{len}, then $\mathcal{B}$ perfectly simulates H_2, else it simulates H_3. Thus, if $\mathcal{A}$ can distinguish the outputs of H_4 and H_5 with non-negligible advantage, then $\mathcal{B}$ breaks the hiding of Com. □

Lemma A.7. *Assuming witness indistinguishability of* NIWI*, hybrid experiments* H_3 *and* H_4 *are computationally indistinguishable.*

Proof. In H_3, we use the *real witness* for proving that c_1^* and c_2^* are encryptions of the same message, while in H_4, we use the *trapdoor witness* for proving that C is a commitment to $(c_1^*\|c_2^*\|vk^*)$. Since NIWI is witness indistinguishable, the two hybrids are computationally indistinguishable. □

Lemma A.8. *Assuming* $(\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is* $\mathsf{IND\text{-}CPA}$ *secure, hybrid experiments* H_4 *and* H_5 *are computationally indistinguishable.*

Proof. We show that if there exists an efficient distinguisher $\mathcal{A}$ that can distinguish between H_4 and H_5, then there exists an efficient adversary $\mathcal{B}$ that breaks $\mathsf{IND\text{-}CPA}$ security. $\mathcal{B}$ is defined as follows:

1. $\mathcal{B}$ first receives a public key pk from $\mathsf{IND\text{-}CPA}$ challenger $\mathcal{C}$.
2. $\mathcal{B}$ computes $(PK_1, SK_1) \leftarrow \mathsf{PKE.Setup}(1^\kappa)$, $\mathsf{crs} \leftarrow \mathsf{NIWI.Setup}$, $(sk^*, vk^*) \leftarrow \mathsf{Gen}(1^\kappa)$ and sets $PK_2 = pk$. Next, it encrypts the challenge message x using PK_1 to compute ciphertext c_1^*
3. $\mathcal{B}$ sends $(\vec{0}, x)$ as its challenge messages to $\mathcal{C}$, and receives a ciphertext c. It sets $c_2^* = c$. Next, it computes the commitment $C = \mathsf{Com}(c_1^*\|c_2^*\|vk^*)$.
4. $\mathcal{B}$ runs the rest of the experiment in the same manner as in H_4 and H_5.
5. Finally, $\mathcal{B}$ sends the output of the experiment to $\mathcal{A}$.
6. If $\mathcal{A}$ outputs H_4, then $\mathcal{B}$ outputs that c is an encryption of x. Else it outputs c is an encryption of $\vec{0}$.

Now, if c is an encryption of x, then $\mathcal{B}$ perfectly simulates experiment H_4, else it simulates H_5. Then, clearly, if $\mathcal{A}$'s output is correct, then so is $\mathcal{B}$'s output. Hence, if $\mathcal{A}$ can distinguish the outputs of the two experiments with non negligible advantage, then $\mathcal{B}$ can win the $\mathsf{IND\text{-}CPA}$ game with the same advantage. □

Lemma A.9. *Assuming* NIWI *is statistically sound,* $i\mathcal{O}$ *is indistinguishability obfuscator and* Com *is perfectly binding, hybrid experiments* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. As in the proof of Lemma A.1, we first argue that both $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ have identical input-output behavior.

Observation A.10. *For all inputs* $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$*,* $\mathsf{Sim}.\mathcal{G}_f(\mathsf{CT}) = \perp$ *if and only if* $\mathsf{Sim}.\mathcal{G}'_f(\mathsf{CT}) = \perp$*.*

Both $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ output $\perp$ if and only if either $\mathsf{Verify}(\sigma, c_1 \| c_2 \| \pi, vk) = 0$ or $\mathsf{NIWI.Verify}(\mathsf{crs}, y, \pi) = 0$ where $y = (c_1, c_2, vk, PK_1, PK_2, C)$. Therefore, we only need to consider valid inputs. Next, we show that any valid input must satisfy one of the two properties listed below.

Claim A.11. *Any valid ciphertext* $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$ *should satisfy one of the following properties :*

- c_1 *and* c_2 *are encryptions of the same message*
- $c_1 \| c_2 \| vk = c_1^* \| c_2^* \| vk^*$.[3]

Proof. Suppose, on the contrary, there exists a valid input such that it satisfies neither of the properties. Since NIWI is statistically sound, if the input is valid, then the statement $y = (c_1, c_2, vk, PK_1, PK_2, C)$ must have either a real witness or a trapdoor witness. Since c_1 and c_2 are encryptions of different messages, a real witness does not exist. Therefore, for the input to be valid, there must exist a trapdoor witness; that is, there exists an s such that $C = \mathsf{Com}(c_1 \| c_2 \| vk; s)$. However, since $C = \mathsf{Com}(c_1^* \| c_2^* \| vk^*)$ and Com is perfectly binding, it follows that $(c_1 \| c_2 \| vk) = (c_1^* \| c_2^* \| vk^*)$. Thus, we have a contradiction. □

Using the previous claim, we can now argue that both $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ have identical input-output behavior.

Claim A.12. *For all valid inputs* $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$*, both* $\mathsf{Sim}.\mathcal{G}_f$ *and* $\mathsf{Sim}.\mathcal{G}'_f$ *have the same functionality.*

Proof. If both c_1 and c_2 are encryptions of the same message, then we have that $\mathsf{PKE.Dec}(c_1, SK_1) = \mathsf{PKE.Dec}(c_2, SK_2) = x$. Therefore both programs $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ output $f(x; r)$, where $r \leftarrow \mathsf{Eval}(K, c_1 \| c_2 \| vk) = \mathsf{Eval}(K', c_1 \| c_2 \| vk)$. If $c_1 \| c_2 \| vk = c_1^* \| c_2^* \| vk^*$, then both $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ output y, where y is $\mathsf{KeyIdeal}$'s response to query x. Therefore, for all valid inputs, $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ have identical input-output behavior. □

We now describe our reduction. Assume A_3 makes ℓ key queries. Consider intermediate hybrids $\mathsf{H}_{5,i}$ $0 \leq i \leq \ell$. In $\mathsf{H}_{5,i}$, we use SK_1 for the first $\ell - i$ key queries, and SK_2 for the remaining i queries. Now, suppose that there exists a PPT distinguisher $\mathcal{A}$ that can distinguish the outputs of $\mathsf{H}_{5,i}$ and $\mathsf{H}_{5,i+1}$. Then, there $\exists$ an adversary $\mathcal{B}$ that can break the security of $i\mathcal{O}$. $\mathcal{B}$ is constructed as follows:

1. $\mathcal{B}$ generates $\mathsf{MPK}, \mathsf{CT}^*$ as in H_5. It sets $st' = SK_1, SK_2, \mathsf{CT}^*$.
2. For the first $(\ell - i - 1)$ key queries by A, $\mathcal{B}$ responds as in H_5. For the last i queries, $\mathcal{B}$ responds as in H_6.
3. For the $(\ell - i)$'th key query f, $\mathcal{B}$ queries $\mathsf{KeyIdeal}$ with f and receives y. Next, it chooses a PRF Key K, computes punctured key $K' \leftarrow \mathsf{Puncture}(K, c_1^* \| c_2^* \| vk^*)$ and defines functions $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$. $\mathcal{B}$ sends $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ to the obfuscation challenger $\mathcal{C}$, receives challenge obfuscation SK_f, which it passes on to A_2.
4. $\mathcal{B}$ runs the rest of the experiment in the same manner as in H_5 and H_6.
5. Finally, $\mathcal{B}$ sends the output of the experiment to $\mathcal{A}$ and forwards $\mathcal{A}$'s response to $\mathcal{C}$.

[3] We thank Xiang Xe for bringing to our attention the incorrect use of signatures in the proof of this lemma in the previous version of this paper. The proof has been fixed in the revised version.

Now, if $\mathcal{C}$ returns obfuscation of $\mathcal{G}_f$, then $\mathcal{B}$ perfectly simulates experiment $\mathsf{H}_{5,i}$, else it simulates experiment $\mathsf{H}_{5,i+1}$. Thus, if $\mathcal{A}$ distinguishes the outputs with non negligible advantage, then clearly $\mathcal{B}$ breaks the security of indistinguishability obfuscation with non negligible advantage. □

Lemma A.13. *Assuming (*Gen*,* Sign*,* Verify*) is a strongly unforgeable one time signature scheme,* NIWI *is statistically sound and* Com *is perfectly binding, hybrid experiments* H_6 *and* H_7 *are statistically indistinguishable.*

Proof. As shown in claim A.11, any valid ciphertext $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$ is such that either c_1 and c_2 are encryptions of the same message or $c_1\|c_2\|vk = c_1^*\|c_2^*\|vk^*$. However, recall that for decryption queries, we only require that $\mathsf{CT} \neq \mathsf{CT}^*$.

If both c_1 and c_2 encrypt the same value, then clearly the use of either secret key is indistinguishable. Then, lets consider the case where $c_1\|c_2\|vk = c_1^*\|c_2^*\|vk^*$, yet $\mathsf{CT} \neq \mathsf{CT}^*$. In this case, it must be that $\pi^*\|\sigma^* \neq \pi\|\sigma$. Now, if $\pi \neq \pi^*$, then since $vk = vk^*$ and $(c_1\|c_2\|\pi) \neq (c_1^*\|c_2^*\|\pi^*)$, we have that σ is a forgery for $(c_1\|c_2\|\pi)$. On the other hand, if $\pi = \pi^*$, then it must be that $\sigma \neq \sigma^*$. In this case, we have that σ is a strong forgery for $(c_1\|c_2\|\pi) = (c_1^*\|c_2^*\|\pi^*)$. We can therefore break the security of the strongly unforgeable one time signature scheme. □

Lemma A.14. *Assuming* (PKE.Setup, PKE.Enc, PKE.Dec) *is* IND-CPA *secure, hybrid experiments* H_7 *and* H_8 *are computationally indistinguishable.*

Proof. Same as proof for Lemma A.8. □

Lemma A.15. *Assuming (*Gen*,* Sign*,* Verify*) is a strongly unforgeable one time signature scheme,* NIWI *is statistically sound and* Com *is perfectly binding, hybrid experiments* H_8 *and* H_9 *are statistically indistinguishable.*

Proof. Same as in proof of Lemma A.13. □

Lemma A.16. *Assuming (*Gen*,* Sign*,* Verify*) is a strongly unforgeable one time signature scheme,* NIWI *is statistically sound, $i\mathcal{O}$ is indistinguishability obfuscator and* comm *is perfectly binding, hybrid experiments* H_9 *and* H_{10} *are computationally indistinguishable.*

Proof. Same as in proof for Lemma A.9 □

Lemma A.17. *Assuming (*Key, Eval, Puncture*) is a puncturable family of PRFs, hybrid experiments* H_{10} *and* H_{11} *are computationally indistinguishable.*

Proof. In H_{10}, on receiving a decryption query $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, we first compute x by decrypting c_1. Next, we compute a fresh PRF key K_i for each $f_i \in \{f\}$ and output $\{f_i(x, \mathsf{Eval}(K_i, c_1\|c_2\|vk))\}$. On the other hand, in H_{11}, we compute x as in H_{10}, but then query DecryptIdeal on x to receive $\{f_i(x, r_i)\}$ where $r_i \xleftarrow{\$} \mathcal{R}_\kappa$. Now, if there exists an efficient adversary that can distinguish between the outputs of H_{10} and H_{11} with non negligible probability, then there exists an efficient adversary that can distinguish between the output of Eval from a truly random string with non negligible probability, thereby breaking the security of a pseudorandom function. □

B Correctness of $\mathcal{FE}$

Theorem B.1. *If* $(\mathsf{Key}, \mathsf{Puncture}, \mathsf{Eval})$ *is a PRF, then the proposed scheme* $\mathcal{FE}$ *satisfies correctness.*

Proof. We first prove this theorem for a single key. Fix any $f \in \mathcal{F}_\kappa, \vec{x} \in \mathcal{X}_\kappa^n$. Consider the distribution $Real_1$: $\{\mathsf{rFE.Dec}(\mathsf{CT}_i, \mathsf{SK}_f)\}_{i=1}^n$, where $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$, $\mathsf{CT}_i = (c_{i,1}, c_{i,2}, \pi_i, vk_i, \sigma_i) \leftarrow \mathsf{rFE.Enc}(x_i, \mathsf{MPK})$ for $i \in [n]$ and $K_f \leftarrow \mathsf{rFE.Keygen}(f, \mathsf{MSK})$. Similarly, consider the $Ideal_1$ distribution $\{f(x_i, r_i)\}_{i=1}^n$, where $r_i \leftarrow \mathcal{R}_\kappa$.

Claim B.2. *Assuming* $\mathsf{Eval}(\cdot, \cdot)$ *is a PRF,* $Real_1$ *and* $Ideal_1$ *distributions are computationally indistinguishable.*

Proof. Note that $\mathsf{rFE.Dec}(\mathsf{CT}_i, \mathsf{SK}_f) = f(x_i, \mathsf{Eval}(K, c_{i,1} \| c_{i,2} \| vk_i))$. Therefore, the $Real_1$ distribution is $\{f(x_i, \mathsf{Eval}(K, c_{i,1} \| c_{i,2} \| vk_i))\}_{i=1}^n$. Suppose there exists an adversary $\mathcal{A}$ that can distinguish between the distributions $Real_1$ and $Ideal_1$ with non-negligible advantage. Then there exists an adversary $\mathcal{B}$ that can break the PRF security of $\mathsf{Eval}(\cdot, \cdot)$. The reduction is as follows :

1. PRF challenger $\mathcal{C}$ chooses a bit $b \leftarrow \{0, 1\}$.
2. For $i = 1$ to n
   (a) $\mathcal{B}$ sends $(c_{i,1} \| c_{i,2} \| vk_i)$ to $\mathcal{C}$, and receives r. If $b = 0$, $r = \mathsf{Eval}(K, c_{i,1} \| c_{i,2} \| vk_i)$, else $r \leftarrow \mathcal{R}_\kappa$.
   (b) $\mathcal{B}$ computes $y_i = f(x_i, r)$.
3. $\mathcal{B}$ sends $\vec{y}$ to $\mathcal{A}$, and depending on $\mathcal{A}$'s guess, $\mathcal{B}$ outputs 0 or 1.

Clearly, if $\mathcal{A}$ distinguishes between the distributions $Real_1$ and $Ideal_1$ with non-negligible advantage, then $\mathcal{B}$ breaks the PRF security with non-negligible advantage. □

This lemma can be extended, via a hybrid argument, to prove that the Real and Ideal distributions are computationally indistinguishable. □