Functional Encryption for Randomized Functionalities

Vipul Goyal[1], Abhishek Jain[2,⋆], Venkata Koppula[3,⋆⋆], and Amit Sahai[4,⋆⋆⋆]

[1] Microsoft Research, India
vipul@microsoft.com
[2] Johns Hopkins University, USA
abhishek@cs.jhu.edu
[3] University of Texas, Austin, USA
kvenkata@cs.utexas.edu
[4] UCLA and the Center for Encrypted Functionalities, USA
sahai@cs.ucla.edu

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 [25,18,5,19,4,21]. 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 $f(x)$.

⋆ The author is partly funded by NSF CNS-1414023. Part of the research was conducted while visiting Microsoft Research, India.
⋆⋆ Part of this research was conducted during internship at Microsoft Research, India.
⋆⋆⋆ Research supported in part from a DARPA/ONR PROCEED award, NSF Frontier Award 1413955, NSF grants 1228984, 1136174, 1118096, and 1065276, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11- 1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

Y. Dodis and J.B. Nielsen (Eds.): TCC 2015, Part II, LNCS 9015, pp. 325–351, 2015.

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 [24,16,15,8,11]. 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 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. As a result, the sample would not be "random" anymore in the auditor's view (since he knows the PRF key).

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. The issue of dishonest encryptors is, in fact, also relevant to the case of determinstc functionalities.[1] However, to the best of our knowledge, this issue was never considered explicitly in previous work on functional encryption. 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 [10]. An alternative would be to have a government agency review a

[1] For example, the FE schemes in [16,15] are not secure against a dishonest encryptor who uses the simulator algorithm to create ciphertexts. Indeed, such an adversary can force arbitrary outputs on an honest receiver. However, a straightforward compilation of these schemes with simulation-sound NIZK proofs of knowledge yields security against dishonest encryptors.

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 security definitions given in [4,21]. We then construct a functional encryption scheme supporting arbitrary randomized polynomial-size circuits assuming indistinguishability obfuscation for circuits and one-way functions. We prove security in the selective model that can be amplified to full security using standard complexity leveraging.

While our focus is on simulation-based security, we note that it cannot be realized for an unbounded number of messages [4,3]. Towards that end, in Sect. 2.1, we also provide indistinguishability based security definitions for randomized functions, generalizing the case of deterministic functions [4,21]. We prove security in the selective model for an unbounded number of messages (again, this can be amplified to full security using standard complexity leveraging[2]).

The starting point for our construction is the functional encryption scheme of [11] 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 [2]. 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 [26]. 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 [9,23] to ensure that valid ciphertexts are unique.

[2] Subsequent to our work, Waters [27] gave a construction of fully secure functional encryption (for deterministic functions) from indistinguishability obfuscation, without complexity leveraging. We leave the problem of adapting our techniques to the scheme of [27] for future work.

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 Other Applications

Subsequent to our work, Garg et al. [12] use functional encryption for randomized functions in NC^1 as a crucial tool to construct fully secure functional encryption for all circuits from multilinear maps. We refer the reader to their paper for more details.

1.3 Related Work

In an independent and concurrent work, Alwen et al. [1] also study functional encryption for randomized functions.[3] The main difference between their work and ours is that they do not consider security against malicious encryptors. In particular, they provide a construction of FE for randomized functions from FE for deterministic functions by encrypting a PRF key along with every message. This PRF key is evaluated over the identifier associated with a function key to sample randomness on the fly, which is then used to compute the function output. Interestingly, they show that a 2-ary version of randomized FE can be used to construct fully homomorphic encryption (see [1] for details). However, they do not provide a construction of such an FE scheme.

We note that while the security definition of [1] suffices for their target application, in this work, we model randomized functionalities following the standard approach in secure computation where in the ideal world, no single party has full control over the randomness used in the function evaluation and instead the randomness is chosen by the trusted party. In particular, we require that the randomness used for the computation is chosen uniformly even if either of the parties is malicious. Indeed, as discussed earlier, this is the main source of non-triviality in our results.

[3] See [17] for the eprint version of our work.

1.4 Organization

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

2 Functional Encryption for Randomized Functions

In this section, we present definitions for functional encryption for randomized functions (or rand-FE for short). We start by presenting the syntax for rand-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 1 (Correctness). *A functional encryption scheme $\mathcal{FE}$ for randomized function family $\mathcal{F}$ is* correct *if for every polynomial $n = n(\kappa)$, every $\boldsymbol{f} \in \mathcal{F}_\kappa^n$ and every $\boldsymbol{x} \in \mathcal{X}_\kappa^n$, 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 1. 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 rand-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. As discussed earlier, this is particularly important when modeling randomized functions.

We consider a a unified adversarial model that captures both malicious receivers and malicious senders. We present both simulation-based and indistinguishability based security definitions. For simplicity, 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.

Simulation Based Security. We now present a simulation-based security definition (or, SIM-security) for rand-FE. If we only consider malicious receivers, then our definition looks essentially identical to the standard (selective) SIM-security definition for FE (for deterinistic functions) [4,21]. In order to provide security against adversarial senders, we extend the existing definition. 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[4] 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 following [3,8]. Our definition is parameterized by q that denotes the number of challenge messages.

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

Definition 2 (SIM-security for rand-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 $\mathsf{REAL}_A^{\mathcal{FE}}(1^\kappa)$:
$(\boldsymbol{x}, st_1) \leftarrow A_1(1^\kappa)$ where $\boldsymbol{x} \in \mathcal{X}_\kappa^q$
$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
$st_2 \leftarrow A_2^{\mathcal{O}_1(\mathsf{MSK},\cdot),\ \mathcal{O}_2(\mathsf{MSK},\cdot,\cdot)}(\mathsf{MPK}, st_1)$
$\mathsf{CT}_i^* \leftarrow \mathsf{rFE.Enc}(x_i, \mathsf{MPK})$ for $i \in [q]$
$\alpha \leftarrow A_3^{\mathcal{O}_1(\mathsf{MSK},\cdot),\ \mathcal{O}_2(\mathsf{MSK},\cdot,\cdot)}(\mathbf{CT}^*, st_2)$
Output $(\boldsymbol{x}, \{f\}, \{g\}, \{y\}, \alpha)$

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

where,

1. **Real experiment:** *In this experiment, $\mathcal{O}_1(\mathsf{MSK}, \cdot)$ denotes the key generation oracle $\mathsf{rFE.Keygen}(\cdot, \mathsf{MSK})$. The set $\{f\}$ denotes the key queries made by A_2 and A_3.*
   $\mathcal{O}_2(\mathsf{MSK}, \cdot, \cdot)$ denotes a decryption oracle that takes inputs of the form (CT, g) where $g \in \mathcal{F}$. If the query is from A_3, then we require that $\mathsf{CT} \neq \mathsf{CT}_i^*$. $\mathcal{O}_2$ computes $\mathsf{SK}_g \leftarrow \mathsf{rFE.Keygen}(g, \mathsf{MSK})$ and returns $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_g)$. The set $\{g\}$ denotes the functions that appear in the decryption queries of A_2 and A_3 and $\{y\}$ denotes the responses of $\mathcal{O}_2$.
2. **Ideal experiment:** *$\mathcal{O}_1'(\cdot)$ denotes the simulator algorithm $S_2(st', \cdot)$ that has oracle access to the ideal functionality $\mathsf{KeyIdeal}(\boldsymbol{x}, \cdot)$. The functionality KeyIdeal accepts key queries f' and returns $f'(x_i, r_i)$ for every $x_i \in \boldsymbol{x}$ and randomly chosen $r_i \in \mathcal{R}_\kappa$. The set $\{f'\}$ denotes the key queries made by S_2 to KeyIdeal.*
   $\mathcal{O}_2'(\cdot, \cdot)$ denotes the simulator algorithm $S_3(st', \cdot, \cdot)$ that has oracle access to ideal functionality $\mathsf{DecryptIdeal}(\cdot, \cdot)$. The functionality DecryptIdeal accepts input queries (x, g') and returns $y' = g'(x; r)$ for randomly chosen $r \in \mathcal{R}_\kappa$. The set $\{g'\}$ denotes the functions that appear in the queries of S_3 and $\{y'\}$ denotes the responses of DecryptIdeal.

We note that in the above selective security definition, pre-ciphertext key queries are essentially redundant since an adversary can defer all such queries to the post-ciphertext key query phase. Nevertheless, we present our definition in the above form to remain syntactically consistent with the full security definition that consists of two distinct key query phases.

Indistinguishability Based Security. Here we present indistinguishability-based security definitions for rand-FE. We give two (incomparable) definitions: the first definition, referred to as $\mathsf{IND}_{\mathsf{pre}}$-security allows for adversaries that make key queries *before* obtaining the public key. The second definition, referred to as

$\mathsf{IND}_{\mathsf{post}}$-security, allows for key queries *after* the adversary receives the public key, but puts additional constraints on the distribution of these queries. In both cases, we strengthen the adversary by allowing decryption queries in a similar manner as the SIM-security definition.

Security against key queries before public key. We first give a security definition for the case where the adversary is restricted to making key queries before obtaining the public key. Similar to the FE definition for deterministic functions [4,21], we consider two worlds: a left world where the adversary requests ciphertexts for challenge message x_0, and a right world where the challenge message is x_1. Our definition differs from standard definition for (deterministic) FE in two ways. First, instead of requiring the outputs corresponding to x_0 and x_1 to be equal (for every key query f), we now require them to be *computationally indistinguishable*[5] (given the auxiliary input of the adversary). Second, we strengthen the adversary by allowing her to make decryption queries in the same manner as the SIM-security definition.

Definition 3 ($\mathsf{IND}_{\mathsf{pre}}$-secure rand-FE). *A functional encryption scheme $\mathcal{FE}$ is $\mathsf{IND}_{\mathsf{pre}}$-secure if for every non-uniform PPT adversary $A = (A_1, A_2, A_3)$, every $z \in \{0,1\}^*$, the distributions $\mathsf{Exp}^0_{\mathcal{FE},A}(1^\kappa, z)$ and $\mathsf{Exp}^1_{\mathcal{FE},A}(1^\kappa, z)$ are computationally indistinguishable, where $\mathsf{Exp}^b_{\mathcal{FE},A}(1^\kappa, z)$ is defined as follows :*

Experiment $\mathsf{Exp}^b_{\mathcal{FE},A}(1^\kappa, z)$:
$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
$(x_0, x_1, st_1) \leftarrow A_1^{\mathsf{rFE.Keygen}(\cdot,\mathsf{MSK})}(1^\kappa, z)$ where $x_0, x_1 \in \mathcal{X}_\kappa$
$st_2 \leftarrow A_2^{\mathcal{O}(\mathsf{MSK},\cdot,\cdot)}(\mathsf{MPK}, st_1)$
$\mathsf{CT}^* \leftarrow \mathsf{rFE.Enc}(x_b, \mathsf{MPK})$
Output $A_3(\mathsf{CT}^*, st_2)$

In the above experiment:

1. *Let $\{f\}$ denote the list of key queries made by A_1 to the key generation oracle. Then, the distributions $(z, \{f(x_0)\})$ and $(z, \{f(x_1)\})$ are computationally indistinguishable.*
2. *$\mathcal{O}(\mathsf{MSK}, \cdot, \cdot)$ denotes a decryption oracle that takes inputs of the form (CT, g) where $g \in \mathcal{F}$. It then computes $\mathsf{SK}_g \leftarrow \mathsf{rFE.Keygen}(g, \mathsf{MSK})$ and returns $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_g)$.*

Remark 2 (Unbounded $\mathsf{IND}_{\mathsf{pre}}$ security). Definition 3 can be naturally extended to allow for multiple challenge messages. The constraint on the key queries $\{f\}$ made by A_2 will now be that given the challenge message vectors $(\boldsymbol{x}_0, \boldsymbol{x}_1)$, for every i, the distributions $(z, \{f(x_0[i])\})$ and $(z, \{f(x_1[i])\})$ are computationally indistinguishable. We call this *unbounded* $\mathsf{IND}_{\mathsf{pre}}$ security.

Note that by a standard hybrid argument, $\mathsf{IND}_{\mathsf{pre}}$ security (for one message) implies unbounded $\mathsf{IND}_{\mathsf{pre}}$ security.

[5] We note that this condition cannot be verified efficiently.

Security against key queries after public-key. Next we give a security definition for the case where the adversary is allowed to make key queries after obtaining the public key. The crucial difference from the previous definition is that we now require that the output distributions in the left and right world should be *statistically indistinguishable.*

Definition 4 ($\mathsf{IND}_{\mathsf{post}}$-secure rand-FE). *A functional encryption scheme $\mathcal{FE}$ is $\mathsf{IND}_{\mathsf{post}}$-secure for the randomized function family $\mathcal{F}$ if for every non-uniform PPT adversary $A = (A_1, A_2)$, every $z \in \{0,1\}^*$, the distributions $\mathsf{Exp}^0_{\mathcal{FE},A}(1^\kappa, z)$ and $\mathsf{Exp}^1_{\mathcal{FE},A}(1^\kappa, z)$ are computationally indistinguishable, where $\mathsf{Exp}^b_{\mathcal{FE},A}(1^\kappa, z)$ is defined as follows :*

Experiment $\mathsf{Exp}^b_{\mathcal{FE},A}(1^\kappa, z)$*:*
$(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{rFE.Setup}(1^\kappa)$
$(x_0, x_1, st_1) \leftarrow A_1(1^\kappa, z)$ where $x_0, x_1 \in \mathcal{X}_\kappa$
$\mathsf{CT}^* \leftarrow \mathsf{rFE.Enc}(x_b, \mathsf{MPK})$
Output $A_2^{\mathsf{rFE.Keygen}(\cdot,\mathsf{MSK}),\mathcal{O}(\mathsf{MSK},\cdot,\cdot)}(\mathsf{MPK}, \mathsf{CT}^*, st_1)$

In the above experiment:

1. *Let $\{f\}$ denote the list of key queries made by A_2 to the key generation oracle. Then the distributions $(\mathsf{MPK}, z, \{f(x_0)\})$ and $(\mathsf{MPK}, z, \{f(x_1)\})$ are* statistically *indistinguishable.*
2. *$\mathcal{O}(\mathsf{MSK}, \cdot, \cdot)$ denotes a decryption oracle that takes inputs of the form (CT, g) where $\mathsf{CT} \neq \mathsf{CT}^*$ and $g \in \mathcal{F}$. It computes $\mathsf{SK}_g \leftarrow \mathsf{rFE.Keygen}(g, \mathsf{MSK})$ and returns $\mathsf{rFE.Dec}(\mathsf{CT}, \mathsf{SK}_g)$.*

Remark 3 (Unbounded $\mathsf{IND}_{\mathsf{post}}$ security). Similar to Definition 3, the above definition can also be naturally extended to capture security for multiple challenge messages. We call this *unbounded* $\mathsf{IND}_{\mathsf{post}}$ security. Note that one-message $\mathsf{IND}_{\mathsf{post}}$ security implies unbounded $\mathsf{IND}_{\mathsf{post}}$ security.

Remark 4 (Statistical vs Computational Indistinguishability). Note that if we modify Definition 4 by requiring the output distributions to be computationally indistinguishable (as in Definition 3, then it may result in a circularity. Consider a key query f from A_2 that simply re-encrypts the plaintext underlying the challenge ciphertext CT^*_b.[6] In this case, the requirement on the output distributions is the same as our desired security guarantee for the challenge ciphertexts, which results in a vaccuous definition. By requiring the output distributions to be statistically indistinguishable, we are able to break such circularity.

SIM implies IND. It is easy to see that SIM-security implies both $\mathsf{IND}_{\mathsf{pre}}$ and $\mathsf{IND}_{\mathsf{post}}$ security. Furthermore, since $\mathsf{IND}_{\mathsf{pre}}$ (resp., $\mathsf{IND}_{\mathsf{post}}$) security for one message implies unbounded $\mathsf{IND}_{\mathsf{pre}}$ (resp., $\mathsf{IND}_{\mathsf{post}}$) security, we have that 1-SIM security implies unbounded $\mathsf{IND}_{\mathsf{pre}}$ and $\mathsf{IND}_{\mathsf{post}}$ security. We state it below:

[6] Note that in Definition 3, such a query is not possible since the adversary is required to make all the key queries *before* receiving the public key.

Lemma 1. *Let $\mathcal{FE}$ be a* 1-SIM*-secure FE scheme for randomized function family $\mathcal{F}$. Then $\mathcal{FE}$ is also unbounded* $\mathsf{IND}_{\mathsf{pre}}$*-secure and unbounded* $\mathsf{IND}_{\mathsf{post}}$*-secure for $\mathcal{F}$.*

The proof follows in the same manner as the case of deterministic functions [4]. We provide a sketch in Appendix B for the case of one message. Combining this with remarks 2 and 3 yields the proof of lemma 1 for unbounded messages.

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. [2]. 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 [11].

Definition 5. *(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:*

$$|Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| \leq \mathsf{negl}(\kappa)$$

Recently, Garg et al. [11] gave the first candidate construction for an indistinguishability obfuscator $i\mathcal{O}$ for the circuit class $P/poly$. Subsequent to their work, Pass et al [22] construct an indistinguishability obfuscator based on an "uber" assumption on multilinear encodings. More recently, Gentry et al [13] construct an indistinguishability obfuscator based on the multilinear subgroup elimination assumption.

3.2 Puncturable Pseudorandom Functions

Puncturable family of PRFs are a special case of constrained PRFs [6,7,20], 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 [26].

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 6. *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* $S \subset \{0,1\}^{n(\kappa)}$, $x \in S$ *and state* st, *consider an experiment where* $K \leftarrow \mathsf{Key}(1^\kappa)$ *and* $K_S \leftarrow \mathsf{Puncture}(K, S)$. *Then*

$$\left|Pr[A_2(K_S, x, \mathsf{Eval}(K, x), \mathrm{st}) = 1] - Pr[A_2(K_S, x, U_{m(\kappa)}, \mathrm{st}) = 1]\right| \leq \mathsf{negl}(\kappa)$$

where U_ℓ *denotes the uniform distribution over* ℓ *bits.*

As observed by [20,6,7], the [14] construction of PRFs from one-way functions easily yield puncturable PRFs.

Theorem 1 ([14,20,6,7]). *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.*

We note that in the above construction, the size of the punctured key K_S grows linearly with the size of the punctured 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 7 (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 [26] 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 :

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

For simplicity of exposition, we present our FE scheme in Sect. 4 using a non-interactive perfectly binding scheme. We stress, however, that it is actually sufficient to use a standard 2-round statistically binding scheme in our construction. Such schemes can be based on one way functions.

4 Our Construction

Let $\mathcal{F}$ denote the family of all PPT functions. We now present a functional encryption scheme $\mathcal{FE}$ for $\mathcal{F}$. For any a priori bounded $q = \mathsf{poly}(\kappa)$, we prove that $\mathcal{FE}$ is q-SIM-secure. Note that from Lemma 1, it follows that $\mathcal{FE}$ is also unbounded $\mathsf{IND}_{\mathsf{pre}}$ and $\mathsf{IND}_{\mathsf{post}}$ secure.

Note that in the case of SIM-security, the size of the secret keys in $\mathcal{FE}$ grows linearly with q. It follows from [4,3,8] that such a dependence on q is necessary.

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 Fig. 1. Note that $\mathcal{G}_f$ has the public key MPK, the secret key SK_1 and the PRF key K hardwired in it.

Input: Ciphertext CT
Constants: MPK, SK_1, K, f

1. Parse $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$.
2. If $\mathsf{Verify}(\sigma, c_1\|c_2\|\pi, vk) = 0$, 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. Compute $x \leftarrow \mathsf{PKE.Dec}(c_1, SK_1)$.
5. Compute $r \leftarrow \mathsf{Eval}(K, c_1\|c_2\|vk)$.
6. Output $f(x; r)$.

Fig. 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 Fig. 2. In this case, the size of SK_f grows linearly with q.

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 A.

Theorem 2. *Assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme* $\mathcal{FE}$ *is* 1-SIM*-secure.*

5 Proof of Theorem 2

We now prove that the proposed scheme $\mathcal{FE}$ is 1-SIM-secure. Our proof can be naturally extended to q-SIM-security, for any a priori fixed $q = \mathsf{poly}(\kappa)$.

We first construct an ideal world adversary aka simulator S in Sect. 5.1. Next, in Sect. 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: (a) First compute $c_1^* \leftarrow \mathsf{PKE.Enc}(\mathbf{0}, PK_1)$ and $c_2^* \leftarrow \mathsf{PKE.Enc}(\mathbf{0}, PK_2)$. (b) 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, that is, $\pi^* \leftarrow \mathsf{NIWI.Prove}(\mathsf{crs}, y, s)$, where the statement $y = (c_1^*, c_2^*, vk^*, PK_1, PK_2, C)$. Finally, it computes $\sigma^* \leftarrow \mathsf{Sign}(c_1^* \| c_2^* \| \pi^*, sk^*)$. It sets $\mathsf{MPK} = (\mathsf{crs}, PK_1, PK_2, C)$ and challenge ciphertext $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$.

Algorithm S_2. S_2 simulates the key generation oracle. Whenever A_2 or 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 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 Fig. 2. $\mathsf{Sim}.\mathcal{G}_f$ has the public key MPK, secret key SK_1, the punctured key K', the challenge ciphertext CT^* and the output value y^* hardwired in it.
4. Return SK_f.

Algorithm S_3. S_3 simulates the decryption oracle. Whenever A_2 or A_3 makes a decryption query (CT, g) where $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$, 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. Return $\mathsf{DecryptIdeal}(x, g)$.

Input: Ciphertext CT
Constants: MPK, SK_1, K', f, $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$, y^*

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 $f(x; r)$.

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

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.

Hybrid H_0*:* This is the real experiment. Here, each decryption query (CT, g) is answered using a decryption key $sk_g \leftarrow i\mathcal{O}(\mathcal{G}_g)$ where $\mathcal{G}_g$ is defined in the same manner as $\mathcal{G}_f$, except that it has function g hardwired in it.

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 A_2 or A_3 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 Fig. 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 ct^* and the output value y^* (as computed above) hardwired in it.
4. Return SK_f.

Hybrid H_2*:* This experiment is the same as H_1, except that we now answer the key queries of A_2 and 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 $\mathsf{CT}^* = (c_1^*, c_2^*, \pi^*, vk^*, \sigma^*)$: 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., $c_2^* \leftarrow \mathsf{PKE.Enc}(\mathbf{0}, PK_2)$.

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_2 and A_3 are answered: each query (CT, g) is answered using a decryption key $sk_g \leftarrow i\mathcal{O}(\mathcal{G}'_f)$ where $\mathcal{G}'_g$ is the same as function $\mathcal{G}_g$ 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 4 of $\mathcal{G}_g$, plaintext x is computed as $x \leftarrow \mathsf{PKE.Dec}(c_2, SK_2)$.

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., $c_1^* \leftarrow \mathsf{PKE.Enc}(\mathbf{0}, PK_1)$.

Hybrid H_9*:* This experiment is the same as H_8, except that we modify the manner in which the decryption queries of A_2 and A_3 are answered: each query (CT, g) is answered using a decryption key $sk_g \leftarrow i\mathcal{O}(\mathcal{G}_f)$.

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_2 and A_3 in the same manner as the simulator algorithm S_3. Note that this is the ideal experiment.

This completes the description of the hybrid experiments. Next, we prove that for every i, the outputs of experiments H_i and H_{i+1} are computationally indistinguishable.

Lemma 2. *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 1. *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 1. *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 $y^* = y$, 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_2 and A_3 together make a total of ℓ key 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 2. *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, $\mathcal{B}$ computes the key for f using $\mathsf{rFE.Keygen}(\cdot, \mathsf{MSK})$. For the last i key queries f, $\mathcal{B}$ 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 the adversary.

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 3. *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_2 and A_3 make a total of 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 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 4. *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 $0^{|\mathsf{en}|}$ 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 5. *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 6. *Assuming* (PKE.Setup, PKE.Enc, PKE.Dec) *is* IND-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 IND-CPA security. $\mathcal{B}$ is defined as follows:

1. $\mathcal{B}$ first receives a public key pk from IND-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 $(\mathbf{0}, x)$ 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 $\mathbf{0}$.

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 IND-CPA game with the same advantage.

Lemma 7. *Assuming* NIWI *is statistically sound, iO is an indistinguishability obfuscator and* Com *is perfectly binding, hybrid experiments* H_5 *and* H_6 *are computationally indistinguishable.*

Proof. As in the proof of Lemma 2, we first argue that both $\mathsf{Sim}.\mathcal{G}_f$ and $\mathsf{Sim}.\mathcal{G}'_f$ have identical input-output behavior.

Observation 2. *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 3. *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*
- $c_1\|c_2\|vk = c_1^*\|c_2^*\|vk^*$.

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 4. *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_2 and A_3 make a total of ℓ key queries. Consider intermediate hybrids $\mathsf{H}_{5,i}$ $0 \le i \le \ell$. 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 $\mathcal{C}$.

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 8. *Assuming (*$\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}$*) 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 3, 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 SK_1 or SK_2 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 9. *Assuming* $(\mathsf{PKE.Setup}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *is* $\mathsf{IND\text{-}CPA}$ *secure, hybrid experiments* H_7 *and* H_8 *are computationally indistinguishable.*

Proof. Same as proof for Lemma 6.

Lemma 10. *Assuming (*$\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}$*) 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 8.

Lemma 11. *Assuming (*$\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify}$*) 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 7.

Lemma 12. *Assuming (*$\mathsf{Key}, \mathsf{Eval}, \mathsf{Puncture}$*) 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 (CT, g), we sample PRF key K and decrypt CT using $sk_g \leftarrow i\mathcal{O}(\mathcal{G}_g)$, where $\mathsf{CT} = (c_1, c_2, \pi, vk, \sigma)$ and $\mathcal{G}_g$ uses randomness $r \leftarrow \mathsf{Eval}(K, c_1 \| c_2 \| vk)$ to compute the output $g(x, \mathsf{Eval}(K, c_1 \| c_2 \| vk))$.

On the other hand, in H_{11}, the output is computed as $g(x, r) \leftarrow \mathsf{DecryptIdeal}(x, g)$ where $r \xleftarrow{\$} \mathcal{R}_\kappa$.

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.

Acknowledgements. We thank Gil Segev for helpful comments on our security definitions. We also thank Ran Canetti, Shafi Goldwasser, Brent Waters and Xiang Xe for useful discussions.

References

1. Alwen, J., Barbosa, M., Farshim, P., Gennaro, R., Gordon, S.D., Tessaro, S., Wilson, D.A.: On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In: Stam, M. (ed.) IMACC 2013. LNCS, vol. 8308, pp. 65–84. Springer, Heidelberg (2013)
2. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (Im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001)
3. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition (2013)
4. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
5. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
6. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
7. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
8. De Caro, A., Iovino, V., Jain, A., O'Neill, A., Paneth, O., Persiano, G.: On the achievability of simulation-based security for functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 519–535. Springer, Heidelberg (2013)
9. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: STOC, pp. 542–552 (1991)
10. Dwork, C., Naor, M., Reingold, O., Rothblum, G.N., Vadhan, S.P.: On the complexity of differentially private data release: efficient algorithms and hardness results. In: STOC, pp. 381–390 (2009)
11. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: FOCS (2013)

12. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Fully secure functional encryption without obfuscation. IACR Cryptology ePrint Archive 2014, 666 (2014), http://eprint.iacr.org/2014/666
13. Gentry, C., Lewko, A.B., Sahai, A., Waters, B.: Indistinguishability obfuscation from the multilinear subgroup elimination assumption. IACR Cryptology ePrint Archive 2014, 309 (2014), http://eprint.iacr.org/2014/309
14. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (1986), http://doi.acm.org/10.1145/6490.6503
15. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: STOC (2013)
16. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)
17. Goyal, V., Jain, A., Koppula, V., Sahai, A.: Functional encryption for randomized functionalities. IACR Cryptology ePrint Archive 2013, 729 (2013)
18. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security (2006)
19. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
20. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: ACM CCS (2013)
21. O'Neill, A.: Definitional issues in functional encryption. IACR Cryptology ePrint Archive 2010 (2010)
22. Pass, R., Seth, K., Telang, S.: Indistinguishability obfuscation from semantically-secure multilinear encodings. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 500–517. Springer, Heidelberg (2014)
23. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (1999)
24. Sahai, A., Seyalioglu, H.: Worry-free encryption: functional encryption with public keys. In: ACM Conference on Computer and Communications Security, pp. 463–472 (2010)
25. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
26. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: Deniable encryption, and more. In: STOC (2014)
27. Waters, B.: A punctured programming approach to adaptively secure functional encryption. IACR Cryptology ePrint Archive 2014, 588 (2014), http://eprint.iacr.org/2014/588

A Correctness of $\mathcal{FE}$

Theorem 3. *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 $f \in \mathcal{F}_\kappa, \boldsymbol{x} \in \mathcal{X}_\kappa^n$. 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 5. *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 $\boldsymbol{y}$ 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 standard hybrid argument, to prove that the Real and Ideal distributions are computationally indistinguishable.

B SIM Security Implies $\mathsf{IND}_{\mathsf{pre}}$ and $\mathsf{IND}_{\mathsf{post}}$ Security

We first prove that 1-SIM security implies one-message $\mathsf{IND}_{\mathsf{pre}}$ security. We actually prove the stronger statement that 1-$\widetilde{\mathsf{SIM}}$ security implies one-message $\mathsf{IND}_{\mathsf{pre}}$ security where in $\widetilde{\mathsf{SIM}}$ security, the adversary is restricted to making all of the key queries before receiving the public key. Let $x_0, x_1 \in \mathcal{X}_\kappa$ be any two messages. Let $\mathsf{REAL}^0(1^\kappa)$ correspond to real world experiment in Definition 3 where the challenge ciphertext corresponds to the encryption of x_0. From 1-$\widetilde{\mathsf{SIM}}$ security, we have that $\mathsf{REAL}^0(1^\kappa)$ is computationally indistinguishable to $\widetilde{\mathsf{IDEAL}}^0(1^\kappa)$ where $\widetilde{\mathsf{IDEAL}}^0(1^\kappa)$ is the corresponding ideal world in Definition 2. (In particular, in $\widetilde{\mathsf{IDEAL}}^1(1^\kappa)$, the simulator receives the output of every key query f on message x_0.) Now, since Definition 3 requires the promise that $(z, \{f(x_0)\})$ and $(z, \{f(x_1)\})$ are computationally indistinguishable, we have that $\widetilde{\mathsf{IDEAL}}^0(1^\kappa)$ is computationally indistinguishable from $\widetilde{\mathsf{IDEAL}}^1(1^\kappa)$, where $\widetilde{\mathsf{IDEAL}}^1(1^\kappa)$ is defined analogously to $\mathsf{IDEAL}^0(1^\kappa)$.[7] Now, finally, we can invoke 1-$\widetilde{SIM}$-security

[7] One may note that since the simulator in our definition performs the key generation in the ideal world, we actually require $(\mathsf{MPK}, \mathsf{MSK}, z, \{f(x_0)\})$ and $(\mathsf{MPK}, \mathsf{MSK}, z, \{f(x_1)\})$ to be computationally indistinguishable. This, however, follows immediately since the key queries $\{f\}$ are independent of the public key MPK.

once again to argue that $\widetilde{\mathsf{IDEAL}}^{1}(1^{\kappa})$ and $\mathsf{REAL}^{1}(1^{\kappa})$ are computationally indistinguishable. Combining the above, we have that $\mathsf{REAL}^{0}(1^{\kappa})$ and $\mathsf{REAL}^{1}(1^{\kappa})$ are computationally indistinguishable, as required.

The proof that 1-SIM security implies one-message $\mathsf{IND}_{\mathsf{post}}$ security follows in a similar manner as above. In particular, note that in this case, we have the promise from Definition 4 that $(z, \{f(x_0)\})$ and $(z, \{f(x_1)\})$ are *statistically* indistinguishable. This immediately implies that $(\mathsf{MPK}, \mathsf{MSK}, z, \{f(x_0)\})$ and $(\mathsf{MPK}, \mathsf{MSK}, z, \{f(x_1)\})$ are computationally indistinguishable. The rest of the steps of the proof follow similarly as above.