Hierarchical Functional Encryption

Zvika Brakerski[*] Gil Segev[†]

Abstract

Functional encryption provides fine-grained access control for encrypted data, allowing each user to learn only specific functions of the encrypted data. We study the notion of *hierarchical* functional encryption, which augments functional encryption with *delegation* capabilities, offering significantly more expressive access control.

We present a *generic transformation* that converts any general-purpose public-key functional encryption scheme into a hierarchical one without relying on any additional assumptions. This significantly refines our understanding of the power of functional encryption, showing that the existence of functional encryption is equivalent to that of its hierarchical generalization.

Instantiating our transformation with the existing functional encryption schemes yields a variety of hierarchical schemes offering various trade-offs between their delegation capabilities (i.e., the depth and width of their hierarchical structures) and underlying assumptions. When starting with a scheme secure against an unbounded number of collusions, we can support *arbitrary* hierarchical structures. In addition, even when starting with schemes that are secure against a bounded number of collusions (which are known to exist under rather minimal assumptions such as the existence of public-key encryption and shallow pseudorandom generators), we can support hierarchical structures of bounded depth and width.

[*]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot 76100, Israel. Email: `zvika.brakerski@weizmann.ac.il`. Supported by the Israel Science Foundation (Grant No. 468/14) and by the Alon Young Faculty Fellowship.

[†]School of Computer Science and Engineering, Hebrew University of Jerusalem, Jerusalem 91904, Israel. Email: `segev@cs.huji.ac.il`. Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11), by the US-Israel Binational Science Foundation (Grant No. 2014632), and by a Google Faculty Research Award.

1 Introduction

The rapidly evolving vision of functional encryption [SW08, BSW11, O'N10] offers tremendous flexibility when accessing encrypted data. Specifically, functional encryption schemes support restricted decryption keys that allow users to learn specific functions of the encrypted data and nothing else. Motivated by the early examples of functional encryption schemes for specific functionalities (such as identity-based encryption [Sha84, BF03, Coc01]), extensive research has recently been devoted to the study of functional encryption (see, for example, [SW08, BSW11, O'N10, GVW12, AGV$^+$13, BO13, BCP14, GGH$^+$13, GKP$^+$13, ABS$^+$15, Wat15, GGH$^+$16, AS15, BS15, KSY15, BKS16] and the references therein).

In a functional encryption scheme, a trusted authority holds a master secret key known only to the authority. When the authority is given the description of some function f as input, it uses its master secret key to generate a functional key sk_f associated with the function f. Now, anyone holding the functional key sk_f and an encryption of any message x, can compute $f(x)$ but cannot learn any additional information about the message x. Such fine-grained access to encrypted data is extremely useful for a wide variety of applications, including expressive access control, spam filtering, mining encrypted databases, and more (we refer the reader to the survey by Boneh, Sahai and Waters [BSW12] for an in-depth discussion of these applications).

Hierarchical functional encryption. Motivated by the applicability of functional encryption to expressive access-control systems, in this paper we study the notion of *hierarchical* functional encryption, which was introduced by Ananth, Boneh, Garg, Sahai and Zhandry [ABG$^+$13]. The hierarchical notion augments standard functional encryption with delegation capabilities, enabling significantly more expressive access control.

Specifically, recall that in a functional encryption scheme, the holder of the master secret key msk can generate a functional key sk_f corresponding to any given function f. In a hierarchical functional encryption scheme, the holder of any such functional key sk_f can in turn generate a functional key $\mathsf{sk}_{g\circ f}$ corresponding to the function $g \circ f$ for any given function g. Now, anyone holding the delegated functional key $\mathsf{sk}_{g\circ f}$ and an encryption of any message x, can compute $(g\circ f)(x) = g(f(x))$ but cannot learn any additional information about the message x. Such expressive delegation capabilities give rise to *hierarchical* access control, which is a sought-after ingredient in modern access control systems. In particular, the notion of hierarchical functional encryption generalizes those of hierarchical attribute-based encryption, hierarchical identity-based encryption and more (see the discussion at the end of Section 3 on the delegation capabilities of functional encryption).

Ananth et al. formalized a notion of security for hierarchical functional encryption schemes, and sketched how the functional encryption scheme of Garg et al. [GGH$^+$13] can be transformed into a hierarchical one by using a general-purpose indistinguishability obfuscator [BGI$^+$12, GGH$^+$13].[1] Their approach, however, is both tailored to the specific functional encryption scheme of Garg et al. [GGH$^+$13], and can only support hierarchical structures of *constant* depth (i.e., can only support a constant number of successive delegations).[2]

The significance of augmenting functional encryption with delegation capabilities was also recognized by an independent work of Chandran, Goyal, Jain and Sahai [CGJ$^+$15]. Their work uses sub-exponentially-secure indistinguishability obfuscation (and sub-exponentially-secure hierarchical

[1] It was recently shown by Ananth and Jain [AJ15] and by Bitansky and Vaikuntanathan [BV15] that indistinguishability obfuscation can be constructed from some flavors of functional encryption. Specifically, from *succinct* functional encryption with *sub-exponential security*. Our approach is both more direct and requires no such properties.

[2] In the hierarchical scheme of Ananth et al. [ABG$^+$13], a delegated functional key $\mathsf{sk}_{g\circ f}$ for the function $g \circ f$ is computed from sk_f by applying the obfuscator to a program that contains sk_f as part of its description. Thus, since sk_f itself consists of such an obfuscated program, this allows for only a constant number of successive delegations.

IBE) for constructing an adaptively-secure hierarchical functional encryption scheme that supports hierarchical structures of any *pre-determined* polynomial depth (see Table 1 for for a comparison of the assumptions underlying the known hierarchical functional encryption schemes and of their supported hierarchical structures). In addition, their work introduces and realizes the notion of a multi-authority functional encryption scheme, which is an interesting distributed extension of functional encryption which is not a subject of study in our work.

1.1 Our Contributions

We present a *generic transformation* that converts any general-purpose public-key functional encryption scheme into a hierarchical one without relying on any additional assumptions (and, in particular, without relying on indistinguishability obfuscation). Our transformation allows instantiations based both on unbounded-collusion functional encryption schemes and on bounded-collusion ones. This level of generality yields a variety of hierarchical schemes based on various assumptions in the standard model. These include strong assumptions such as indistinguishability obfuscation [GGH$^+$13, Wat15] or multilinear maps [GGH$^+$16], and much milder assumptions such as learning with errors [GKP$^+$13], and even the existence of any public-key encryption scheme and low-depth pseudorandom generator [GVW12].

We stress that our result stands even if it turns out that indistinguishability obfuscation is impossible to achieve or requires strong computational assumptions. One could view it as evidence that the hierarchical properties do not stem from the "obfuscation-like" features of functional encryption, but rather from the more rudimentary properties that are achievable under minimal assumptions.

Security and efficiency. In terms of security, the schemes resulting from our transformation guarantee semi-adaptive security. Moreover, by assuming that the underlying functional encryption scheme is sub-exponentially secure, we obtain adaptive security via a standard complexity leveraging argument. In terms of efficiency, our approach results in keys and ciphertexts with a rather low overhead compared to the efficiency of the underlying functional encryption scheme. A ciphertext in our scheme is essentially a ciphertext of the underlying scheme. A delegated functional key for $g \circ f$ contains two functional keys for functions that essentially compute f and g, respectively, in addition to some basic cryptographic computation (an evaluation of a PRF, and an encryption of a ciphertext of the underlying scheme). We refer the reader to the overview below for more details on the security and efficiency of the schemes resulting from our approach.

Instantiations. The variety of schemes resulting from our transformation offer different trade-offs between their underlying assumptions and their delegation capabilities (i.e., the depth and width of the hierarchical structures that they support). For example, instantiating our transformation with the schemes of Garg et al. [GGH$^+$13] and Waters [Wat15] results in hierarchical schemes that support hierarchical structures of any polynomial depth and any polynomial width (where these polynomials do not have to be specified in advance). In addition, instantiating our transformation with the schemes of Goldwasser et al. [GKP$^+$13] or Gorbunov et al. [GVW12] results in hierarchical schemes that support hierarchical structures of any constant depth and any pre-determined polynomial width. This should be compared to the hierarchical scheme of Ananth et al. [ABG$^+$13] that is constructed from the specific functional encryption scheme of Garg et al. [GGH$^+$13], and can only support hierarchical structures of *constant* depth, and to the hierarchical scheme of Chandran et al. [CGJ$^+$15] that is constructed based on even stronger assumptions and still requires an a-priori bound on the depth of the support hierarchical structures. We refer the reader to Table 1 for a comparison of the assumptions underlying the known hierarchical functional encryption schemes and of their supported hierarchical structures.

Assumption	Hierarchical Structure	
	Depth	Width
$i\mathcal{O}$ [ABG+13]	Constant	Unbounded
Sub-exponentially-secure $i\mathcal{O}$ and sub-exponentially-secure HIBE [CGJ+15]	Any fixed polynomial	Unbounded
Unbounded-collusion FE (our work)	Unbounded	Unbounded
Bounded-collusion FE (our work)	Constant	Any fixed polynomial

Table 1: A comparison of the assumptions underlying the known hierarchical functional encryption schemes and of their supported hierarchical structures. We note that indistinguishability obfuscation ($i\mathcal{O}$) is known to imply unbounded-collusion functional encryption [Wat15], which in turn clearly implies bounded-collusion functional encryption. In addition, bounded-collusion functional encryption is implied by much milder assumptions such as learning with errors [GKP+13], and even the existence of any public-key encryption scheme and low-depth pseudorandom generator [GVW12].

1.2 Overview of Our Approach

Formally, a hierarchical FE scheme contains the standard $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ algorithms, in addition to a new *delegation* algorithm $\mathsf{Delegate}$. The delegation algorithm $\mathsf{Delegate}(\mathsf{hsk}_f, g)$ is identical in syntax to the KG algorithm, except that it takes a functional key hsk_f (which can itself be the output of a previous delegation) instead of the master secret key msk. Its output is a key $\mathsf{hsk}_{g\circ f}$ corresponding to the composed function $g \circ f$.

Indeed, the way we implement this functionality is by associating a unique master secret key with any delegable functional key. Namely, a delegable key hsk_f (with respect to the master key pair $(\mathsf{msk}, \mathsf{mpk})$) will contain a fresh master secret key msk' in addition to a "standard" functional key for a re-encryption function $\mathsf{sk}_{\mathsf{ReEnc}_{f,\mathsf{mpk}'}}$ (the key pair $(\mathsf{msk}', \mathsf{mpk}')$ is generated using the standard setup procedure). The function $\mathsf{ReEnc}_{f,\mathsf{mpk}'}(x)$, intuitively, takes an input x and outputs a *functional encryption* of $f(x)$ under the new key mpk'. It is obvious that since msk' is a part of hsk_f, then the owner of $\mathsf{hsk}_f = (\mathsf{sk}_{\mathsf{ReEnc}_{f,\mathsf{mpk}'}}, \mathsf{msk}')$ can derive $f(x)$ itself if it so desires. The beauty of this procedure is that it can then be repeated. If hsk_f needs to be delegated via $\mathsf{Delegate}(\mathsf{hsk}_f, g)$, then one only needs to generate a new pair $(\mathsf{msk}'', \mathsf{mpk}'')$, use msk' to obtain $\mathsf{sk}'_{\mathsf{ReEnc}_{g,\mathsf{mpk}''}}$ and output $\mathsf{hsk}_{g\circ f} = (\mathsf{sk}_{\mathsf{ReEnc}_{f,\mathsf{mpk}'}}, \mathsf{sk}'_{\mathsf{ReEnc}_{g,\mathsf{mpk}''}}, \mathsf{msk}'')$. In the decryption process, we start with some $\mathsf{ct} = \mathsf{FE.Enc}_{\mathsf{mpk}}(x)$, use the first component of the key to obtain $\mathsf{ct}' = \mathsf{FE.Enc}_{\mathsf{mpk}'}(f(x))$, and then using the second component to obtain $\mathsf{ct}'' = \mathsf{FE.Enc}_{\mathsf{mpk}''}(g(f(x)))$. Finally, msk'' is used to decrypt ct'' and thus learn the value $g(f(x))$.

Care needs to be taken in order to securely realize the above intuition. In particular, one has to come up with a source of randomness for the re-encryption process. This is done by slightly modifying the encryption algorithm of the hierarchical scheme such that $\mathsf{Enc}(\mathsf{mpk}, x) = \mathsf{FE.Enc}(\mathsf{mpk}, (x, K, \perp))$, where K is a key to a puncturable pseudorandom function PRF, and $\perp$ is a placeholder that will only be used in the proof. Similarly, we will generate functional keys of the form $\mathsf{sk}_{\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}}$, where t is a random tag and c is a random string that will be used in the proof. The function $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}(x, K, \perp)$ will compute $f(x)$ and encrypt the tuple $(f(x), K', \perp)$ under msk' using randomness r'. The randomness for the generation of K' and r' is produced by evaluating $\mathsf{PRF}_K(t)$.

For the sake of our security proof, one last addition is made to the description of $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}$.

If its input is of the form $(\cdot, \cdot, \mathsf{k})$, where k is a key for a symmetric encryption scheme, then the first two arguments are ignored and $\mathsf{SKE.Dec}_{\mathsf{k}}(c)$ is output. Thus we implement a "trapdoor circuit" (or a "Trojan") as per [DIJ+13, ABS+15].

Security notion. The notions of security that we consider in this work are those formalized by Ananth et al. [ABG+13]. Specifically, we consider adversaries that obtain functional keys for various functions of their choice by issuing key-generation queries and delegation queries. We require that such adversaries have only a negligible advantage in distinguishing the encryptions of two challenge messages, x_0^* and x_1^*, of their choice as long as for any function f for which they obtain a functional key it holds that $f(x_0^*) = f(x_1^*)$, where such a key may be produced either as a result of a key-generation query or a delegation query (we refer the reader to Section 3 for more details).

We prove that if the underlying scheme $\mathcal{FE}$ is selectively secure then the resulting hierarchical scheme is selectively secure, and if $\mathcal{FE}$ is semi-adaptively secure then the resulting hierarchical scheme is semi-adaptively secure.[3] We leave it as an intriguing open problem to design a hierarchical functional encryption scheme that guarantees adaptive security. We note that already in the less-expressive setting of identity-based encryption, designing adaptively-secure hierarchical schemes is extremely challenging. In particular, Lewko and Waters [LW14] recently showed why known proof methods fall short of proving adaptive security even for adaptively-secure hierarchical identity-based encryption (which is a special case of hierarchical FE) without degrading exponentially with the number of delegation levels.

Proof overview. Let us focus on the case of selective security, semi-adaptive security follows by a practically identical argument. In the selective security game, the adversary first specifies challenge messages x_0^* and x_1^*, receives mpk, and then makes a sequence of key-generation and delegation queries. One could visualize the structure that is generated by these queries as a tree, whose root is $(\mathsf{msk}, \mathsf{mpk})$ and whose nodes are the key pairs that are generated upon each call to KG or $\mathsf{Delegate}$. Each such call generates a new child for one of the nodes in the tree, as per the adversary's choice. Each node is associated with a function f which was input to $\mathsf{KG}/\mathsf{Delegate}$ in its creation, and also with a function $\tilde{f}$, which is the composition of all functions from the root to that node. If $\tilde{f}(x_0^*) = \tilde{f}(x_1^*)$ then we say that the node is *observable*, since the adversary is allowed to see the functional key $\mathsf{hsk}_{\tilde{f}}$ associated with that node. We can assume w.l.o.g that all the leaves of the tree are observable.

The high-level intuition of the proof is the following. Let us pretend that ReEnc is actually capable of outputting a re-encrypted ciphertext which is encrypted with true randomness, rather than with pseudorandomness. Now, consider an unobservable node i (i.e., a node corresponding to f_i and $\tilde{f}_i$ for which $\tilde{f}_i(x_0^*) \neq \tilde{f}_i(x_1^*)$) that only has observable children. This means that all functions $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}$ that are generated relative to this node's msk_i will output the same value whether the challenge ciphertext is an encryption of x_0^* or of x_1^*. The security of the underlying scheme will guarantee that the re-encrypted ciphertext cannot be used to distinguish x_0^* from x_1^*. Let us take another leap of faith and pretend that not only the re-encrypted ciphertext cannot distinguish x_0^* from x_1^* but it is in fact identical in both cases. Then the above process can propagate towards the root of the tree, where at every step we increase the number of nodes whose output is the same regardless of whether the challenge ciphertext encrypts x_0^* or x_1^*. Once this process gets all the way

[3] We briefly remind the reader the differences between selective, semi-adaptive, and adaptive security. *Selective* security considers adversaries that specify their challenge messages before seeing the public parameters or making any key queries. *Semi-adaptive* security, as recently defined by Chen and Wee [CW14], considers adversaries that specify their challenge messages after seeing the public parameters but before making any key queries. Finally, *adaptive* security considers adversaries that specify their challenge messages even after seeing the public parameters and making key queries.

to the root and applied to the challenge ciphertext itself, the proof is complete.

This intuition is implemented using the mechanisms of punctured programming [SW14] and "trapdoor circuits" [DIJ$^+$13] (or "Trojans" [ABS$^+$15]). We will replace the c values in $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}$ with symmetric encryptions of our "fantasy ciphertexts" (ones that are encrypted with true randomness), and append the challenge ciphertext with the appropriate symmetric decryption key (in fact, multiple symmetric keys will be needed, one for every level of the hierarchy, and one has to carefully control their propagation along the tree). Puncturable PRFs will be used to argue that the use of fantasy ciphertexts is indistinguishable from the actual output of $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}$, which will allow the proof idea from above to go through. This requires a careful and delicate argument since we can only puncture a PRF key that had been generated with fresh randomness, hence one has to also consider fantasy PRF keys and propagate them along the tree as well together with the fantasy ciphertexts. The formal proof thus contains many fine points and a large number of steps, and is provided in Section 4.

1.3 Related Work

Hierarchical encryption schemes. Encryption schemes supporting a hierarchical structure have been extensively studied in the setting of identity-based encryption, and have been recently studied in the more general setting of attribute-based encryption and functional encryption.

The line of research on hierarchical identity-based encryption has been extremely successful, starting with schemes in the random oracle model, evolving through selectively-secure schemes in the standard model and graduating to adaptively secure schemes for polynomially many levels. It is far beyond the scope of this paper to provide an extensive overview of this line of research, and we refer the reader to [GS02, HL02, BB04, BBG05, BW06, GH09, Wat09, ABB10a, ABB10b, LOS$^+$10, LW10, LW11, CHK$^+$12, LW14] and the references therein.

Recently, Boneh et al. [BGG$^+$14] constructed an attribute-based encryption scheme that supports delegation of keys. This scheme enables anyone holding a key sk_P corresponding to a predicate P to generate a key $\mathsf{sk}_{P \wedge Q}$ corresponding to the predicate $P \wedge Q$ for any given predicate Q. Now, given the key $\mathsf{sk}_{P \wedge Q}$ and an encryption of any pair (x, m), one can recover m if and only if $(P \wedge Q)(x) = 1$. Although the setting of attribute-based encryption is significantly more expressive than the identity-base one, it does not seem to come close to the general setting of functional encryption that we consider in this paper.

Finally, as discussed above, Ananth et al. [ABG$^+$13] sketched how the functional encryption scheme of Garg et al. [GGH$^+$13] can be transformed into a hierarchical one by using a general-purpose indistinguishability obfuscator. When compared to their scheme our approach offers two main advantages. First, whereas Ananth et al. rely on a specific scheme and on indistinguishability obfuscation, we can rely on any underlying general-purpose scheme. This enables us to rely on a variety of underlying assumptions, including learning with errors and even the existence of any public-key encryption scheme and low-depth pseudorandom generators, as discussed in Section 1.1. Furthermore, as new functional encryption schemes are presented, they can immediately be plugged in to our construction. Second, the schemes resulting from our transformation guarantee semi-adaptive security, whereas the scheme of Ananth et al. guarantees only the somewhat less realistic notion of selective security.

Encapsulation techniques in functional encryption. Key encapsulation is a very useful approach for improving both the efficiency and the security of encryption schemes. Specifically, key encapsulation typically means that instead of encrypting a message x under a fixed key sk, one can instead sample a fresh key k, encrypt x under k, and then encrypt k under sk. Recently, Ananth et al. [ABS$^+$15], followed by Brakerski et al. [BKS16], showed that key encapsulation is useful also for functional encryption, and can be used for generically enhancing the functionality and the security of such schemes. Their approaches suggest that encapsulation techniques may in fact be a general tool that is useful in the design of functional encryption schemes. As discussed in Section 1.2, our construction relies on encapsulation techniques as a key ingredient, significantly extending the initial ideas of Ananth et al. and Brakerski et al. from encapsulating keys to realizing a re-encryption mechanism that generates a hierarchical structure.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we provide an overview of the notation, definitions, and tools underlying our constructions. In Section 3 we present the notion of a hierarchical functional encryption scheme and define its security. In Section 4 we present our generic construction of a hierarchical functional encryption scheme.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. Throughout the paper, we denote by λ the security parameter. A function $\mathsf{neg} : \mathbb{N} \to \mathbb{R}$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{neg}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *computationally indistinguishable* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that $\left|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]\right| \leq \mathsf{neg}(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

2.1 Pseudorandom Functions

Let $\{\mathcal{K}_\lambda, \mathcal{X}_\lambda, \mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of sets and let $\mathcal{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ be a function family with the following syntax:

- $\mathsf{PRF.Gen}$ is a probabilistic polynomial-time algorithm that takes as input the unary representation of the security parameter λ, and outputs a key $K \in \mathcal{K}_\lambda$.
- $\mathsf{PRF.Eval}$ is a deterministic polynomial-time algorithm that takes as input a key $K \in \mathcal{K}_\lambda$ and a value $x \in \mathcal{X}_\lambda$, and outputs a value $y \in \mathcal{Y}_\lambda$.

The sets $\mathcal{K}_\lambda$, $\mathcal{X}_\lambda$, and $\mathcal{Y}_\lambda$ are referred to as the *key space*, *domain*, and *range* of the function family, respectively. For $K \in \mathcal{K}_\lambda$, we use the notation $\mathsf{PRF.Eval}(K, \cdot)$, $\mathsf{PRF.Eval}_K(\cdot)$ and $\mathcal{PRF}_K(\cdot)$ interchangeably. The following is the standard definition of a pseudorandom function family.

Definition 2.1 (Pseudorandomness)**.** A function family $\mathcal{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ is *pseudorandom* if for every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\begin{aligned}
&\mathsf{Adv}_{\mathcal{PRF},\mathcal{A}}(\lambda) \\
&\stackrel{\mathsf{def}}{=} \left| \Pr_{K \leftarrow \mathsf{PRF.Gen}(1^\lambda)} \left[\mathcal{A}^{\mathsf{PRF.Eval}_K(\cdot)}(1^\lambda) = 1\right] - \Pr_{f \leftarrow F_\lambda} \left[\mathcal{A}^{f(\cdot)}(1^\lambda) = 1\right] \right| \\
&\leq \mathsf{neg}(\lambda),
\end{aligned}$$

for all sufficiently large $\lambda \in \mathbb{N}$, where F_λ is the set of all functions that map $\mathcal{X}_\lambda$ into $\mathcal{Y}_\lambda$.

In addition to the standard notion of a pseudorandom function family, we rely on the seemingly stronger (yet existentially equivalent) notion of a *puncturable* pseudorandom function family [KPT+13, BW13, SW14, BGI14]. In terms of syntax, this notion asks for an additional probabilistic polynomial-time algorithm, $\mathsf{PRF.Punc}$, that takes as input a key $K \in \mathcal{K}_\lambda$ and a set $S \subseteq \mathcal{X}_\lambda$ and outputs a "punctured" key K_S. The properties required by such a puncturing algorithm are captured by the following definition.

Definition 2.2 (Puncturable PRF)**.** A pseudorandom function family $\mathcal{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$ is *puncturable* if the following properties are satisfied:

1. **Functionality:** For all sufficiently large $\lambda \in \mathbb{N}$, for every set $S \subseteq \mathcal{X}_\lambda$, and for every $x \in \mathcal{X}_\lambda \setminus S$ it holds that

$$\Pr_{\substack{K \leftarrow \mathsf{PRF.Gen}(1^\lambda);\\ K_S \leftarrow \mathsf{PRF.Punc}(K,S)}} [\mathsf{PRF.Eval}_K(x) = \mathsf{PRF.Eval}_{K_S}(x)] = 1.$$

2. **Pseudorandomness at punctured points:** Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be any probabilistic polynomial-time algorithm such that $\mathcal{A}_1(1^\lambda)$ outputs a set $S \subseteq \mathcal{X}_\lambda$, a value $x \in S$, and state information state. Then, for any such $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\begin{aligned} &\mathsf{Adv}_{\mathcal{PRF},\mathcal{A}}(\lambda) \\ &\quad \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{A}_2(K_S, \mathsf{PRF.Eval}_K(x), \mathsf{state}) = 1\right] - \Pr\left[\mathcal{A}_2(K_S, y, \mathsf{state}) = 1\right]\right| \\ &\quad \leq \mathsf{neg}(\lambda) \end{aligned}$$

for all sufficiently large $\lambda \in \mathbb{N}$, where $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $(S, x, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda)$, $K_S = \mathsf{PRF.Punc}(K, S)$, and $y \leftarrow \mathcal{Y}_\lambda$.

For our constructions we rely on pseudorandom functions that need to be punctured only at a single point (i.e., in both parts of Definition 2.2 it holds that $S = \{x^*\}$ for some $x^* \in \mathcal{X}_\lambda$). As observed by [KPT+13, BW13, SW14, BGI14] the GGM construction [GGM86] of PRFs from any one-way function can be easily altered to yield such a puncturable pseudorandom function family.

Augmented evaluation. When dealing with pseudorandom functions that need to be punctured only at a single point, we find it natural to consider an "augmented" evaluation algorithm that outputs a pre-determined value y^* at the punctured point. That is, we extend the functionality of $\mathsf{PRF.Eval}$ such that given an augmented key of the form $(K_{x^*}, (x^*, y^*))$, it holds that

$$\mathsf{PRF.Eval}_{(K_{x^*},(x^*,y^*))}(x) = \begin{cases} y^*, & \text{if } x = x^* \\ \mathsf{PRF.Eval}_{K_{x^*}}(x), & \text{if } x \neq x^* \end{cases} .$$

2.2 Private-Key Encryption with Pseudorandom Ciphertexts

A private-key encryption scheme over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ is a triplet $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms. The key-generation algorithm KG takes as input the unary representation 1^λ of the security parameter $\lambda \in \mathbb{N}$ and outputs a secret key k. The encryption algorithm Enc takes as input a secret key k and a message $x \in \mathcal{X}_\lambda$, and outputs a ciphertext c. The decryption algorithm Dec takes as input a secret key k and a ciphertext c, and outputs a message $x \in \mathcal{X}_\lambda$ or the dedicated symbol $\perp$. In terms of correctness we require that for any key k that is produced by $\mathsf{KG}(1^\lambda)$ and for every message $x \in \mathcal{X}_\lambda$ it holds that $\mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, x)) = x$ with probability 1 over the internal randomness of the algorithms Enc and Dec. We also require that a uniformly distributed string does not decrypt to a valid message with overwhelming probability, i.e. $\mathsf{Dec}(\mathsf{k}, \rho) = \perp$ with probability $(1 - \mathsf{neg}(\lambda))$ over the randomness of the key k and a uniformly distributed string ρ of the same length as the ciphertext[4]. In terms of security, we rely on the following standard notion of pseudorandom ciphertexts which can be based on any one-way function (see, for example, [Gol04]).

Definition 2.3 (Pseudorandom ciphertexts)**.** A private-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ has *pseudorandom ciphertexts* if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{PC}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{PC}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{neg}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the random variable $\mathsf{Exp}^{\mathsf{PC}}_{\Pi,\mathcal{A}}(\lambda)$ is defined via the following experiment:

1. $\mathsf{k} \leftarrow \mathsf{KG}(1^\lambda)$, $b \leftarrow \{0,1\}$.
2. $(x^*, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{Enc}(\mathsf{k},\cdot)}(1^\lambda)$, where $x^* \in \mathcal{X}_\lambda$.
3. $c_0^* \leftarrow \mathsf{Enc}(\mathsf{k}, x^*)$, $c_1^* \leftarrow \{0,1\}^{|c_0^*|}$.
4. $b' \leftarrow \mathcal{A}_2^{\mathsf{Enc}(\mathsf{k},\cdot)}(c_b^*, \mathsf{state})$.
5. If $b' = b$ then output 1, and otherwise output 0.

2.3 Public-Key Functional Encryption

A public-key functional encryption scheme over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is a quadruple $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ of probabilistic polynomial-time algorithms. The setup algorithm Setup takes as input the unary representation 1^λ of the security parameter $\lambda \in \mathbb{N}$ and outputs a master-secret key msk and a master-public key mpk. The key-generation algorithm KG takes as input a master-secret key msk and a function $f \in \mathcal{F}_\lambda$, and outputs a functional key sk_f. The encryption algorithm Enc takes as input a master-public key mpk and a message $x \in \mathcal{X}_\lambda$, and outputs a ciphertext ct. In terms of correctness we require that for all sufficiently large $\lambda \in \mathbb{N}$, for every function $f \in \mathcal{F}_\lambda$ and message $x \in \mathcal{X}_\lambda$ it holds that $\mathsf{Dec}(\mathsf{KG}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{mpk}, x)) = f(x)$ with all but a negligible probability over the internal randomness of the algorithms Setup, KG, and Enc.

We rely on the standard indistinguishability-based notion of adaptive security for public-key functional encryption (see, for example, [BSW11, O'N10, BO13, ABS+15]), asking that encryptions of any two messages, x_0 and x_1, are computationally indistinguishable given access to functional keys for any function f such that $f(x_0) = f(x_1)$.

Definition 2.4 (Adaptive security)**.** A functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *adaptively secure* if for any probabilistic polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{FE}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{FE}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{neg}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the random variable $\mathsf{Exp}^{\mathsf{FE}}_{\Pi,\mathcal{A}}(\lambda)$ is defined via the following experiment:

[4] More accurately, since the ciphertext length may (in general) not be a fixed function of the security parameter, the uniform string ρ is sampled as follows: Given the key k, encrypt a fixed message (say, the message 0) to obtain a ciphertext c, and then uniformly sample $\rho \leftarrow \{0,1\}^{|c|}$, where $|c|$ denotes the bit-length of c.

1. $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$.
2. $(x_0^*, x_1^*, \mathsf{state}) \leftarrow \mathcal{A}_1^{\mathsf{KG}(\mathsf{msk},\cdot)}(1^\lambda, \mathsf{mpk})$, where $x_0^*, x_1^* \in \mathcal{X}_\lambda$, and for each function $f \in \mathcal{F}_\lambda$ with which $\mathcal{A}_1$ queries $\mathsf{KG}(\mathsf{msk}, \cdot)$ it holds that $f(x_0^*) = f(x_1^*)$.
3. $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_b^*)$.
4. $b' \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk},\cdot)}(\mathsf{ct}^*, \mathsf{state})$, where for each function $f \in \mathcal{F}_\lambda$ with which $\mathcal{A}_2$ queries $\mathsf{KG}(\mathsf{msk}, \cdot)$ it holds that $f(x_0^*) = f(x_1^*)$.
5. If $b' = b$ then output 1, and otherwise output 0.

In addition to the above notion of adaptive security we consider two natural relaxations: semi-adaptive security, and selective security. Semi-adaptive security is defined via an experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{semiFE}}(\lambda)$ that is obtained from the experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{FE}}(\lambda)$ by asking the adversary to determine the challenge messages before making any key-generation queries (but *after* receiving the public key). Selective security is defined via an experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{selFE}}(\lambda)$ that is obtained from the experiment $\mathsf{Exp}_{\Pi,\mathcal{A}}^{\mathsf{FE}}(\lambda)$ by asking the adversary to determine the challenge messages in advance (i.e., *before* receiving the public key).

Known constructions. General-purpose functional encryption schemes that satisfy the above notion of adaptive security are known to exist based on a variety of assumptions. Ananth et al. [ABS+15] gave a generic transformation from selective security to adaptive security, implying that schemes that are adaptively secure for any number of key-generation queries can be based on indistinguishability obfuscation [GGH+13, Wat15], differing-input obfuscation [BCP14, ABG+13], and multilinear maps [GGH+16]. In addition, schemes that are adaptively secure for a bounded number $B = B(\lambda)$ of key-generation queries can be based on the Learning with Errors (LWE) assumption (where the length of ciphertexts grows with B and with a bound on the depth of allowed functions) [GKP+13], based on any public-key encryption scheme and pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with B and with an upper bound on the circuit size of the functions) [GVW12], and even based on any public-key encryption scheme (for $B = 1$) [GVW12].

3 Hierarchical Functional Encryption

In this section we define the notion of a hierarchical functional encryption scheme and formalize several notions of security for such schemes (based on [ABG+13]). A hierarchical functional encryption scheme is a functional encryption scheme that supports delegation of functional keys: Given a functional key sk_f corresponding to a function f, and given a function g, it is possible to efficiently compute a functional key $\mathsf{sk}_{g\circ f}$ corresponding to the function $g \circ f$ (i.e., the function that first applies f and then applies g). This capability is provided via a delegation algorithm denote $\mathsf{Delegate}$.

Formally, a hierarchical functional encryption scheme over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is a tuple $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ of probabilistic polynomial-time algorithms, where $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is a functional encryption scheme (see Section 2.3), and $\mathsf{Delegate}$ is a delegation algorithm that operates as follows: It takes as input a functional key sk_f (which had been produced either by the key-generation algorithm or by the delegation algorithm itself) corresponding to a function $f \in \mathcal{F}_\lambda$, and a function $g \in \mathcal{F}_\lambda$, and outputs a functional key $\mathsf{sk}_{g\circ f}$.

Correctness. In terms of correctness we require that for every $\lambda \in \mathbb{N}$, for every polynomial $\ell = \ell(\lambda)$, for every sequence of functions $f_1, \ldots, f_\ell \in \mathcal{F}_\lambda$, and for every message $x \in \mathcal{X}_\lambda$, it holds that

$$\mathsf{Dec}(\mathsf{sk}_{f_\ell\circ\cdots\circ f_1}, \mathsf{Enc}(\mathsf{mpk}, x)) = (f_\ell \circ \cdots \circ f_1)(x)$$

with all but a negligible probability over the internal randomness of the algorithms Setup, KG, Enc and $\mathsf{Delegate}$, where $\mathsf{sk}_{f_1} \leftarrow \mathsf{KG}(\mathsf{msk}, f_1)$ and $\mathsf{sk}_{f_{i+1}\circ\cdots\circ f_1} \leftarrow \mathsf{Delegate}(\mathsf{sk}_{f_i\circ\cdots\circ f_1}, f_{i+1})$ for every $i \in [\ell - 1]$. One can also consider schemes that support ℓ delegation levels for some *fixed* polynomial $\ell = \ell(\lambda)$, although we note that our scheme in this paper supports *any* polynomial number of delegation levels.

Security. As in the work of Ananth et al. [ABG+13, Appendix E] we consider the natural extensions of the existing indistinguishability-based definitions of functional encryption [BSW11, O'N10] to the hierarchical setting. Specifically, we consider adversaries that obtain functional keys for various functions of their choice by issuing key-generation queries and delegation queries. We require that such adversaries have only a negligible advantage in distinguishing the encryptions of two challenge messages, x_0^* and x_1^*, of their choice as long as for any function f for which they obtain a functional key it holds that $f(x_0^*) = f(x_1^*)$.

The experiment $\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda)$. Let $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ be a hierarchical public-key functional encryption scheme over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, and let $\mathcal{A}$ be a probabilistic polynomial-time adversary. For each $\lambda \in \mathbb{N}$ we denote by $\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda)$ the random variable that is defined via the following experiment involving the scheme Π, the adversary $\mathcal{A}$, and a challenger:

1. **Setup phase:** The challenger samples $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $b \leftarrow \{0, 1\}$.
2. **Pre-challenge phase:** $\mathcal{A}$ on input $(1^\lambda, \mathsf{mpk})$ adaptively issues queries of the form $(f, \mathsf{parent}, \mathsf{mode})$, where $f \in \mathcal{F}_\lambda$, $\mathsf{parent} \in \mathbb{N} \cup \{0\}$ and $\mathsf{mode} \in \{\mathsf{OutputKey}, \mathsf{StoreKey}\}$. The ith query $(f_i, \mathsf{parent}_i, \mathsf{mode}_i)$ is answered by the challenger as follows:
   (a) If $\mathsf{parent} = 0$ then the challenger generates $\mathsf{hsk}_i \leftarrow \mathsf{KG}(\mathsf{msk}, f)$.
   (b) Else, if $\mathsf{hsk}_{\mathsf{parent}_i}$ had already been generated (and is not $\perp$), then the challenger generates $\mathsf{hsk}_i \leftarrow \mathsf{Delegate}(\mathsf{hsk}_{\mathsf{parent}_i}, f)$. Otherwise set $\mathsf{hsk}_i = \perp$.
   (c) Finally, if $\mathsf{mode}_i = \mathsf{OutputKey}$ then the challenger outputs hsk_i, and if $\mathsf{mode} = \mathsf{StoreKey}$ then the challenger outputs $\perp$.
3. **Challenge phase:** $\mathcal{A}$ outputs $(x_0^*, x_1^*) \in \mathcal{X}_\lambda \times \mathcal{X}_\lambda$, and then the challenger computes $\mathsf{ct}^* \leftarrow \mathsf{Enc}(\mathsf{mpk}, x_b^*)$ and sends it to $\mathcal{A}$.
4. **Post-challenge phase:** $\mathcal{A}$ adaptively issues queries as in the pre-challenge phase.
5. **Output phase:** $\mathcal{A}$ outputs b', and the output of the experiment is 1 if and only if $b' = b$.

Valid adversaries. As standard in functional encryption, we rule out adversaries that can easily distinguish between the two challenge messages, x_0^* and x_1^*, using their queries. Specifically, we say that an adversary is *valid* if for any query $(f_i, \mathsf{parent}_i, \mathsf{mode}_i)$ where $\mathsf{mode}_i = \mathsf{OutputKey}$, it holds that $\tilde{f}_i(x_0^*) = \tilde{f}_i(x_1^*)$, where $\tilde{f}$ is defined recursively by $\tilde{f}_i = f_i \circ \tilde{f}_{\mathsf{parent}_i}$ and $f_0(x) = x$ (if any of these values is not well defined then $\tilde{f}_i(x) \equiv \perp$ for all x). Having defined the experiment $\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda)$ and the notion of a valid adversary, we are now ready to present our notion of adaptive security for hierarchical functional encryption schemes.

Definition 3.1. A hierarchical functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *adaptively secure* if for any probabilistic polynomial-time valid adversary $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{neg}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$.

In addition to our notion of adaptive security we consider two natural relaxations: semi-adaptive security, and selective security. Semi-adaptive security is defined via an experiment $\mathsf{Exp}^{\mathsf{semiHFE}}_{\Pi,\mathcal{A}}(\lambda)$ that is obtained from the experiment $\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda)$ by eliminating the pre-challenge query phase (note that the adversary determines the challenge messages *after* receiving the public key). Selective security is defined via an experiment $\mathsf{Exp}^{\mathsf{selHFE}}_{\Pi,\mathcal{A}}(\lambda)$ that is obtained from the experiment $\mathsf{Exp}^{\mathsf{HFE}}_{\Pi,\mathcal{A}}(\lambda)$ by asking the adversary to determine the challenge messages in advance (i.e., *before* receiving the public key).

Discussion: The delegation capabilities of functional encryption. It is important to point out that given a functional key sk_f, one cannot hope to delegate anything beyond the set of functions $g \circ f$ while maintaining the security properties of functional encryption. To see this, assume towards contradiction that there exists a function h such that h cannot be expressed as $g \circ f$, but sk_h can be derived from sk_f. Since the value of $h(x)$ cannot be inferred just by examining the value of $f(x)$, there must exist two inputs, x_0 and x_1 such that $f(x_0) = f(x_1)$ but $h(x_0) \neq h(x_1)$. Given sk_f, therefore, one should not be able to distinguish encryptions of x_0 and x_1, but by delegating to sk_h, this becomes possible, hence the contradiction.

The above optimality claim may seem a little confusing when we think about special cases such as attribute-based encryption (ABE) or even identity-based encryption (IBE). In ABE for example, each ciphertext contains an attribute x and a message m, and sk_f reveals m if and only if $f(x) = 1$. In hierarchical ABE (HABE) [GVW13, BGG+14], given sk_f, one should be able to derive $\mathsf{sk}_{f \wedge f'}$ for all f'. At first glance, this seems to not be covered by our definition since $f \wedge f'$ cannot be expressed as $g \circ f$. However, we notice that in fact when thinking about HABE as a special case of functional encryption, it must be the case that what we call sk_f, is in fact a functional key for the function $f^+(x, m) = ((f(x) = 1)?(x, m) : \perp)$ (i.e., the function that takes (x, m) as input, and if $f(x) = 1$ it returns (x, m) and otherwise it returns $\perp$). This is because if $f(x) = 1$ then x can always be recovered by considering delegated keys that fix the value of each bit of x to 0 or 1, and check if decryption still works. It is clear from this viewpoint that $(f \wedge f')^+$ can be seen as $g \circ f^+$ for an appropriate g. Therefore, our definition and construction are fully compatible also with the more restricted settings of HABE and HIBE.

4 Our Generic Transformation

In this section we show how to transform any general-purpose public-key functional encryption scheme into a hierarchical one. Our construction relies on the following building blocks:

1. A general-purpose public-key functional encryption scheme $\mathcal{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KG}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$.
2. A private-key encryption scheme $\mathcal{SKE} = (\mathsf{SKE.KG}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.
3. A puncturable pseudorandom function family $\mathcal{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$.

Our hierarchical scheme $\mathcal{HFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is defined as follows.

- **The setup algorithm.** On input the security parameter 1^λ the setup algorithm samples and outputs $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$.
- **The encryption algorithm.** On input the public key mpk and a message x, the encryption algorithm first samples a PRF key $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$. Then, it computes and outputs $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, (x, K, \perp))$. (Note that the message space of the resulting scheme is thus smaller than that of the original scheme.)

- **The key-generation algorithm.** On input the master secret key msk and a function f, the key-generation algorithm first generates a fresh key pair $(\mathsf{msk}', \mathsf{mpk}') \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and uniformly samples a tag $t \leftarrow \{0,1\}^\lambda$. Then, it uniformly samples $c \leftarrow \{0,1\}^\ell$ where $\ell = \ell(\lambda)$ is the length of an $\mathcal{SKE}$ encryption of an $\mathcal{FE}$ ciphertext.[5] Finally, it computes $\mathsf{sk}_f \leftarrow \mathsf{FE.KG}(\mathsf{msk}, \mathsf{ReEnc}_{f,t,\mathsf{mpk}',c})$, where ReEnc is defined in Figure 1, and outputs $\mathsf{hsk}_f = (\mathsf{sk}_f, \mathsf{msk}')$.
- **The delegation algorithm.** On input a (possibly delegated) functional key of the form $\mathsf{hsk}_{f_i \circ \cdots \circ f_1} = (\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_i}, \mathsf{msk}')$ for some integer $i \geq 1$, and a function f_{i+1}, the delegation algorithm uses the key-generation algorithm described above to compute $(\mathsf{sk}_{f_{i+1}}, \mathsf{msk}'') \leftarrow \mathcal{HFE}.\mathsf{KG}(\mathsf{msk}', f_{i+1})$, and outputs $\mathsf{hsk}_{f_{i+1} \circ \cdots \circ f_1} = (\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_i}, \mathsf{sk}_{f_{i+1}}, \mathsf{msk}'')$.
- **The decryption algorithm.** On input a functional key $\mathsf{hsk}_{f_i \circ \cdots \circ f_1} = (\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_i}, \mathsf{msk}')$ for some integer $i \geq 1$, and a ciphertext ct, the decryption algorithm first sets $\mathsf{ct}_0 = \mathsf{ct}$ and computes $\mathsf{ct}_j \leftarrow \mathsf{FE.Dec}(\mathsf{sk}_{f_j}, \mathsf{ct}_{j-1})$ for $j = 1, \ldots, i$. Then, ct_i is decrypted by using msk' for generating a functional key for the identity function $\mathsf{ID} \in \mathcal{F}$:

$$w \leftarrow \mathsf{FE.Dec}(\mathsf{FE.KG}(\mathsf{msk}', \mathsf{ID}), \mathsf{ct}_i).$$

  Finally, w is parsed as a triplet $w = (y, \cdot, \cdot)$, of which the first element y is returned as output.

$\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}(x, K, \mathsf{k})$

1. Compute $\mathsf{ct} \leftarrow \mathsf{SKE.Dec}(\mathsf{k}, c)$, $(s, r) = \mathsf{PRF.Eval}(K, t)$, and $K' = \mathsf{PRF.Gen}(1^\lambda; s)$.
2. If $\mathsf{ct} \neq \perp$ then output ct, and otherwise output $\mathsf{FE.Enc}(\mathsf{mpk}', (f(x), K', \perp); r)$.

Figure 1: The function $\mathsf{ReEnc}_{f,t,\mathsf{mpk}',c}$.

In what follows we first discuss the correctness of our resulting scheme, then discuss its parameters and overhead, and then state and prove its security based on that of its underlying building blocks.

Correctness. The correctness of our scheme follows easily by induction on the delegation depth i. Let $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$, and fix a message $x \in \mathcal{X}_\lambda$ and a sequence of functions $f_1, \ldots, f_i \in \mathcal{F}_\lambda$.

For $i = 1$ the correctness of decrypting a ciphertext $\mathsf{ct}_0 \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, (x, K_0, \perp))$ using a key $\mathsf{hsk}_{f_1} = (\mathsf{sk}_{f_1}, \mathsf{msk}_1) \leftarrow \mathsf{KG}(\mathsf{msk}, f_1)$ follows from that of the underlying scheme $\mathcal{FE}$. Specifically, the decryption algorithm first computes $\mathsf{ct}_1 \leftarrow \mathsf{FE.Dec}(\mathsf{sk}_{f_1}, \mathsf{ct}_0)$, and by the correctness of $\mathcal{FE}$ with an overwhelming probability it holds that $\mathsf{ct}_1 = \mathsf{FE.Enc}(\mathsf{mpk}_1, (f_1(x), K_1, \perp); r_1)$, where $(s_1, r_1) = \mathsf{PRF.Eval}(K_0, t)$ for some t chosen during the key generation, $K_1 = \mathsf{PRF.Gen}(1^\lambda; s_1)$, and mpk_1 is a master public key that is sampled together with msk_1. Next, the decryption algorithm decrypts ct_1 with msk_1, and noting that r_1 is pseudorandom given the triplet $(\mathsf{mpk}_1, f_1(x), K_1)$ we can once again rely on the correctness of the underlying scheme $\mathcal{FE}$ and argue that the decryption algorithm outputs $f_1(x)$ with an overwhelming probability.[6]

[5] To be accurate, ℓ is also a function of the message space of the scheme and of the specific properties of the master secret key. We refrain from mentioning these implicit parameters to avoid cluttering of notation. We note however that this imposes an a-priori bound on the length of the ciphertext and thus also on the message space of our resulting scheme. Lifting this restriction is an interesting research direction.

[6] If we further assume that either the underlying functional encryption scheme is perfectly correct, or that the underlying pseudorandom function produces outputs whose marginal distribution is uniform, the argument significantly simplifies and there is no need to argue that r_1 is pseudorandom given the triplet $(\mathsf{mpk}_1, f_1(x), K_1)$.

Assume that the scheme is correct for up to $i-1$ levels of delegation, and consider decrypting a ciphertext $\mathsf{ct}_0 \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, (x, K_0, \perp))$ using a key $\mathsf{hsk}_{f_i \circ \cdots \circ f_1} = (\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_i}, \mathsf{msk}_i) \leftarrow \mathsf{Delegate}(\mathsf{hsk}_{f_{i-1}}, f_i)$ that is generated using i levels of delegation. Then, the correctness for up to $i-1$ levels guarantees that by repeatedly applying the keys $\mathsf{sk}_{f_1}, \ldots, \mathsf{sk}_{f_{i-1}}$ starting with the initial ciphertext ct_0 as prescribed by the decryption algorithm, we obtain with an overwhelming probability a ciphertext $\mathsf{ct}_{i-1} = \mathsf{FE.Enc}(\mathsf{mpk}_{i-1}, ((f_{i-1} \circ \cdots \circ f_1)(x), K_{i-1}, \perp); r_{i-1})$ for some mpk_{i-1}, K_{i-1} and r_{i-1}, where r_{i-1} is pseudorandom given the triplet $(\mathsf{mpk}_{i-1}, (f_{i-1} \circ \cdots \circ f_1)(x), K_{i-1})$. Next, the decryption algorithm computes $\mathsf{ct}_i \leftarrow \mathsf{FE.Dec}(\mathsf{sk}_{f_i}, \mathsf{ct}_{i-1})$, and by the correctness of $\mathcal{FE}$ with an overwhelming probability it holds that $\mathsf{ct}_i = \mathsf{FE.Enc}(\mathsf{mpk}_i, ((f_i \circ \cdots \circ f_1)(x), K_i, \perp); r')$, where $(s_i, r_i) = \mathsf{PRF.Eval}\left(K_{i-1}, t\right)$ for some t chosen during the key generation, $K_i = \mathsf{PRF.Gen}\left(1^\lambda; s_{i-1}\right)$, and mpk_i is a master public key that is sampled together with msk_i. Note that again r_i is pseudorandom given the triplet $(\mathsf{mpk}_i, (f_i \circ \cdots \circ f_1)(x), K_i)$. Therefore, when the decryption algorithm decrypts ct_i with msk_i, it outputs $(f_i \circ \cdots \circ f_1)(x)$ with an overwhelming probability.

Parameters and overhead. We now discuss the parameters that govern the properties that are required of the underlying scheme and thus the overhead of our construction. We address two parameters of the hierarchy: The *width* which is the maximal number of delegated keys that are derived from each key at the previous level, and the *depth* which is the maximal number of successive derivations.[7] The functionality and security of our scheme hold for arbitrary and a-priori unbounded width and depth. However, if the underlying scheme is restricted in some way, then this restriction could propagate through our reduction. For example, if the underlying scheme only supports bounded collusion, then the maximal width will be restricted. Furthermore, since the ReEnc function produces a functional ciphertext with respect to the next level of the hierarchy, certain instantiations could produce a cascading effect that will increase the overhead. We analyze these restrictions below and show that in some cases they can be overcome completely and in others they can be managed.

Define the *compactness parameter* of a (standard) FE scheme, denoted $C(\lambda, S)$, as the computational complexity of encrypting a message of length λ (or some other fixed length which does not depend on S), while allowing to produce functional keys for size S functions. Note that C is also a bound on the length of the ciphertext, and in the currently-known schemes it also governs the complexity of key generation (see Section 2.3 for the currently-known schemes). Then in our construction, the ciphertext encryption complexity at depth i, which we denote by C_i is at most $C_i \leq C(\lambda, C_{i+1} \cdot \mathrm{poly}(\lambda))$. This relation follows immediately from the description of the scheme.

For a scheme which only allows bounded collusion, the compactness is $C(\lambda, S, B)$, where B is the bound on the number of collusions. In this case, the width factors in as well such that for a scheme with width w it holds that $C_i \leq C(\lambda, C_{i+1} \cdot \mathrm{poly}(\lambda), w)$.

In particular, in the known schemes with unbounded collusion [GGH+13, Wat15, GGH+16], the encryption complexity is independent of S and therefore instantiating our construction with such a scheme will support arbitrary polynomial depth and width while keeping the encryption complexity polynomial. In fact, one can show, via a little modification of [ABS+15], that any scheme that supports unbounded collusions can be modified using randomized encodings to one where the compactness is independent of S.

For known schemes with bounded collusion, such as those based on public-key encryption [GVW12] or on LWE [GKP+13], the compactness is $C(\lambda, S, B) \leq \mathrm{poly}(\lambda) \cdot S \cdot B$, which implies that C_i is bounded by $C_{i+1} \cdot \mathrm{poly}(\lambda) \cdot w$. If we intend to support a total depth d, then unfolding the reduction, the bound we have is $C_0 \leq w^d \cdot \lambda^{O(d)}$. This means that if we wish to keep the encryption

[7] One could consider a more fine-grained view of the parameters, e.g. that the maximal width itself depends on the depth of the key. Such analyses follow the same principles presented here.

complexity polynomial in λ, we can only allow $d = O(1)$ and $w = \text{poly}(\lambda)$. Furthermore, we must know w ahead of time in order to instantiate the parameters of the scheme.

Security. The following theorem captures the security of our resulting scheme. We note that the assumptions stated in the theorem are all known to be implied by the existence of any (selectively-secure) general-purpose public-key functional encryption scheme (see Section 2 for formal descriptions of our building blocks and their known instantiations).

Theorem 4.1. *Assuming that (1) $\mathcal{FE}$ is semi-adaptively (resp., selectively) secure (2) $\mathcal{SKE}$ has pseudorandom ciphertexts, and (3) $\mathcal{PRF}$ is a puncturable pseudorandom function family, then $\mathcal{HFE}$ is a semi-adaptively-secure (resp., selectively-secure) hierarchical functional encryption scheme.*

Proof. For ease of exposition we focus here on the case where the underlying scheme $\mathcal{FE}$ is semi-adaptively secure. The proof for the case where $\mathcal{FE}$ is only selectively secure is identical, except for requiring the adversary to provide the challenge messages prior to receiving the public parameters. Let $\mathcal{A}$ be a valid probabilistic polynomial-time adversary (as defined in Section 3). We present a sequence of experiments and upper bound $\mathcal{A}$'s advantage in distinguishing each two consecutive experiments. The first experiment is the experiment $\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda)$ and the last experiment is completely independent of the bit b. This enables us to prove that there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\mathsf{Adv}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2} \right| \leq \mathsf{neg}(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

How to read this proof. To read our proof, one starts from the first hybrid and proceeds in order to the next, each adjacent hybrid is shown to be computationally indistinguishable from its predecessor. When a loop is encountered, this means that a sequence of hybrids is now being defined, one hybrid for each "iteration" of the loop. The hybrid defined in the first iteration needs to be indistinguishable from the last hybrid before the loop, and all hybrids except the first need to be indistinguishable from the hybrid of the previous iteration. In a *nested loop*, each iteration of the external loop represents a generation of many hybrids, as many as the internal loop generates. In such case, in the first iteration of the external loop, and the first iteration of the internal loop, the hybrid being defined needs to be indistinguishable from the one preceding the loop. However, in the next execution of the external loop, the first iteration of the internal needs to be indistinguishable with the *last* iteration of the internal loop that have been carried out in the previous iteration of the external loop. For example, say that the external loop iterates for $i = 1, \ldots, S$ and the internal loop iterates for $j = 1, \ldots, T$. Then what we prove for $\mathcal{H}^{(i,j)}$ is that: $\mathcal{H}^{(1,1)}$ is indistinguishable from the last hybrid before the loop, $\mathcal{H}^{(i,1)}$ for $i > 1$ is indistinguishable from $\mathcal{H}^{(i-1,T)}$, and for $i, j > 1$ that $\mathcal{H}^{(i,j)}$ is indistinguishable from $\mathcal{H}^{(i,j-1)}$.

In order to explain the purpose of the different steps in the proof, we also include *invariants* which are properties of the distribution of the current experiment. The invariant holds *only* at that point in the proof where it appears and does not necessarily hold in following hybrids. An invariant inside a loop holds whenever the flow of the proof reaches that point in the loop. Namely, going back to our nested loop example from above, an invariant that appears after the "for $i = 1, \ldots, S$" statement, holds for the experiment immediately preceding the loop, and for all hybrids $\mathcal{H}^{(i,T)}$, except $\mathcal{H}^{(S,T)}$. An invariant that appears after the "for $j = 1, \ldots, T$" statement, should hold for the hybrid immediately preceding the loop, as well as for all $\mathcal{H}^{(i,T)}$, except $\mathcal{H}^{(S,T)}$.

We advise the reader to read our proof as if it was an execution of a computer program. We believe that while this proof writing method is still not very widely used, it is quite beneficial in writing complicated proofs, and will find additional uses. In what follows we first describe the notation used throughout the proof, and then describe the experiments.

Notation. Let $Q = Q(\lambda)$ denote a polynomial upper bound on the number of queries that are made by $\mathcal{A}$ in the experiment $\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda)$. We denote these queries by $\{(f_i, \mathsf{parent}_i, \mathsf{mode}_i)\}_{i\in[Q]}$ and we also consider an implicit "zeroth" query which generates the master key pair $(\mathsf{msk}, \mathsf{mpk})$ of the scheme. This allows us to define the *depth* of the ith query, denoted $\mathsf{d}(i)$, s.t. $\mathsf{d}(0) = 0$ and $\mathsf{d}(i) = \mathsf{d}(\mathsf{p}(i)) + 1$ for $i > 0$, where we use $\mathsf{p}(i)$ as shorthand for parent_i. Thus we can view $\mathcal{A}$'s queries as a tree rooted by the zeroth query, where each query $(f_i, \mathsf{parent}_i, \mathsf{mode}_i)$ is the child of the query $\mathsf{p}(i)$ and has depth $\mathsf{d}(i)$ in the tree.

For any query $i \in \{0, \ldots, Q\}$, we define a function $\tilde{f}_i$ as follows: $\tilde{f}_0$ is the identity function, and for all $i > 0$ we define $\tilde{f}_i = f_i \circ \tilde{f}_{\mathsf{p}(i)}$. In other words, the ith query $(f_i, \mathsf{parent}_i, \mathsf{mode}_i)$ generates a delegated key that allows to compute the function $\tilde{f}_i(x)$ given an encryption of x. We say that the ith query is *observable* if $\tilde{f}_i(x_0^*) = \tilde{f}_i(x_1^*)$, and *unobservable* otherwise. We note that if the ith query is unobservable then necessarily $\mathsf{mode}_i = \mathsf{StoreKey}$.

We let $(\mathsf{msk}_i, \mathsf{mpk}_i)$ denote the key pair generated by the challenger while answering the ith query, and let $(\mathsf{msk}_0, \mathsf{mpk}_0)$ be the master key pair $(\mathsf{msk}, \mathsf{mpk})$ that is generated by the setup algorithm. Similarly, we let t_i denote the tag that is sampled while answering the ith query.

We denote by x_0^* and x_1^* the challenge messages that are chosen by $\mathcal{A}$, and by K^* the PRF key that is used for computing the challenge ciphertext. We further define $K_0^* = K^*$, and for all $i > 0$ we define K_i^*, s_i^*, and r_i^* as follows: $(s_i^*, r_i^*) = \mathsf{PRF.Eval}(K^*_{\mathsf{p}(i)}, t_i)$, and $K_i^* = \mathsf{PRF.Gen}(1^\lambda; s_i^*)$. Note that these are exactly the values that are computed by the ReEnc function produced in the ith query, when evaluated on the challenge ciphertext.

Finally, throughout the proof we find it convenient to denote by $\$$ a fresh value that is sampled uniformly and independently of all other existing values.

Experiment $\mathcal{H}_0$. This is the experiment $\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda)$ (see Section 3).

Experiment $\mathcal{H}_1$. This experiment is obtained from the experiment $\mathcal{H}_0$ by having the challenger sample in advance the tags and the key pairs that are used for replying to $\mathcal{A}$'s queries. In fact, we will sample these values in a redundant manner so that we prepare several such triplets for each query, and the choice of which triplet to use is determined by the depth of the query. We thus have the following claim:

Specifically, at the beginning of the experiment, for all $i, d \in [Q]$ the challenger samples $t_{i,d}^{(\mathrm{o})}, t_{i,d}^{(\mathrm{u})} \leftarrow \{0,1\}^\lambda$ and $(\mathsf{msk}_{i,d}, \mathsf{mpk}_{i,d}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$. Then, the experiment proceeds exactly as in $\mathcal{H}_1$, and whenever the challenger needs to sample t_i and $(\mathsf{msk}_i, \mathsf{mpk}_i)$ for replying to the ith query, it will use $t_i = t_{i,\mathsf{d}(i)}^{(\mathrm{o})}$ if i is an observable query, and $t_i = t_{i,\mathsf{d}(i)}^{(\mathrm{u})}$ otherwise. It will further use $(\mathsf{msk}_i, \mathsf{mpk}_i) = (\mathsf{msk}_{i,\mathsf{d}(i)}, \mathsf{mpk}_{i,\mathsf{d}(i)})$.

looking ahead, this experiment allows the challenger to know in advance, for every possible depth, a polynomial superset of the tags and key pairs that will be produced for replying to queries of this depth. The view of the adversary in this experiment is distributed identically to its view in the experiment $\mathcal{H}_0$, yielding the following observation:

Observation 4.2. *For all $\lambda \in \mathbb{N}$ it holds that*

$$\Pr[\mathcal{H}_0(\lambda) = 1] = \Pr[\mathcal{H}_1(\lambda) = 1]\,.$$

Experiment $\mathcal{H}_2$. This experiment is obtained from the experiment $\mathcal{H}_1$ as follows. After the generation of the tags $t_{i,d}^{(\mathrm{o})}$ and $t_{i,d}^{(\mathrm{u})}$, and before interacting with the adversary, the challenger checks if any of the values $t^{(\mathrm{o})}_{i,d}$ or $t^{(\mathrm{u})}_{i,d}$ for some $(i,d) \in [Q]^2$ appears more than once. In such case the output of the experiment is defined as $\perp$, and otherwise the experiment is identical to the experiment $\mathcal{H}_1$. A standard union bound implies that the experiments $\mathcal{H}_1$ and $\mathcal{H}_2$ differ with probability at most $2(Q+1)^4 \cdot 2^{-\lambda} = \mathsf{neg}(\lambda)$, yielding the following observation:

Observation 4.3. *For all $\lambda \in \mathbb{N}$ it holds that*

$$|\Pr[\mathcal{H}_1(\lambda) = 1] - \Pr[\mathcal{H}_2(\lambda) = 1]| \leq \frac{2(Q+1)^4}{2^\lambda}.$$

Experiment $\mathcal{H}_3$. This experiment is obtained from the experiment $\mathcal{H}_2$ by sampling a sequence $\mathsf{k}_0, \ldots, \mathsf{k}_{Q-1} \leftarrow \mathsf{SKE.KG}(1^\lambda)$ of symmetric keys (one for each possible depth – recall that Q is always an upper bound on the maximal depth), and modifying the symmetric ciphertext c that is generated by the key-generation algorithm when replying to each query as follows: When replying to the ith query $(f_i, \mathsf{parent}_i, \mathsf{mode}_i)$, instead of sampling c uniformly, the key-generation algorithm computes

$$\boxed{c_i = \mathsf{SKE.Enc}\left(\mathsf{k}_{\mathsf{d}(i)-1}, \mathsf{ct}_i;\ \$\ \right)}$$

where $\mathsf{ct}_i = \mathsf{FE.Enc}(\mathsf{mpk}_i, (\tilde{f}_i(x^*_b), K^*_i, \perp); r^*_i)$ (recall that throughout the proof we find it convenient to denote by $\$$ a fresh value that is sampled uniformly and independently of all other existing values).

Note that ct_i is exactly the same as the "ct_i" value that is computed in the process of decrypting the challenge ciphertext using the ith functional key (and is also computed as an intermediate value when decrypting the challenge ciphertext with any descendant of the ith key). See the decryption algorithm above.

It thus makes sense to extend our notation and denote the challenge ciphertext by ct_0 (as in the decryption algorithm). Note that while ct_0 is encrypted with true randomness and includes a properly generated PRF key, all other ct_i's are encrypted using pseudorandomness and contain PRF keys that were generated pseudorandomly. We further say that ct_i is observable if the ith query is an observable query and unobservable otherwise.

To see why the adversary's view in $\mathcal{H}_3$ is indistinguishable from $\mathcal{H}_2$, we note that in $\mathcal{H}_3$, the symmetric keys $\mathsf{k}_0, \ldots, \mathsf{k}_{Q-1}$ are used only for generating the c_i's. In other words, this experiment can be carried out given only access to an encryption oracle $\mathsf{SKE.Enc}(\mathsf{k}_d, \cdot)$ for each $d \in \{0, \ldots, Q-1\}$ (instead of explicit access to the actual keys $\mathsf{k}_0, \ldots, \mathsf{k}_{Q-1}$). This enables us to use the ciphertext pseudorandomness of $\mathcal{SKE}$ to prove computational indistinguishability from $\mathcal{H}_2$, yielding the following claim in a rather straightforward manner:

Claim 4.4. *Assuming that $\mathcal{SKE}$ has pseudorandom ciphertexts, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$|\Pr[\mathcal{H}_2(\lambda) = 1] - \Pr[\mathcal{H}_3(\lambda) = 1]| \leq \mathsf{neg}(\lambda)$$

for all sufficiently large $\lambda \in \mathbb{N}$.

<u>`For` $\boldsymbol{d = 0, \ldots, Q}$:</u>

Invariant 4.5. In the previous experiment, it should hold that all ciphertexts ct_i that correspond to *unobservable* queries (i.e., queries for which $\tilde{f}_i(x^*_0) \neq \tilde{f}_i(x^*_1)$) such that $\mathsf{d}(i) < d$ are of the form $\mathsf{FE.Enc}(\mathsf{mpk}_i, (\perp, K^*_i, \mathsf{k}_{\mathsf{d}(i)});\ \$\ )$, and all such ciphertext such that $\mathsf{d}(i) = d$ are of the form $\mathsf{ct}_i = \mathsf{FE.Enc}(\mathsf{mpk}_i, (\tilde{f}_i(x^*_b), K^*_i, \perp);\ \$\ )$. Further, if $\mathsf{d}(i) \leq d$ then $K^*_i = \mathsf{PRF.Gen}(1^\lambda;\ \$\ )$. More specifically:

- If i is such that $\mathsf{d}(i) < d$ and $\tilde{f}_i(x_0^*) \neq \tilde{f}_i(x_1^*)$, then it holds that $\mathsf{ct}_i = \mathsf{FE.Enc}\left(\mathsf{mpk}_i, \left(\perp, K_i^*, \mathsf{k}_{\mathsf{d}(i)}\right); \$\right)$ and $K_i^* = \mathsf{PRF.Gen}(1^\lambda; \$)$.
- If i is such that $\mathsf{d}(i) = d$ and $\tilde{f}_i(x_0^*) \neq \tilde{f}_i(x_1^*)$, then it holds that $\mathsf{ct}_i = \mathsf{FE.Enc}\left(\mathsf{mpk}_i, \left(\tilde{f}_i(x_b^*), K_i^*, \perp\right); \$\right)$ and $K_i^* = \mathsf{PRF.Gen}(1^\lambda; \$)$.
- If i is such that $\mathsf{d}(i) > d$ or $\tilde{f}_i(x_0^*) = \tilde{f}_i(x_1^*)$, then it holds that $\mathsf{ct}_i = \mathsf{FE.Enc}\left(\mathsf{mpk}_i, \left(\tilde{f}_i(x_b^*), K_i^*, \perp\right); r_i^*\right)$ and $K_i^* = \mathsf{PRF.Gen}(1^\lambda; s_i^*)$.

We note that this indeed holds for $d = 0$ in experiment $\mathcal{H}_3$.

<u>`For` $\boldsymbol{i = 0, \ldots, Q}$:</u>

Experiment $\mathcal{H}_4^{(i,d)}$. In this experiment, the challenger changes the way ct_i is computed as follows. Before generating ct_i, the challenger checks if both $\mathsf{d}(i) = d$ and ct_i is unobservable ($\tilde{f}_i(x_0^*) \neq \tilde{f}_i(x_1^*)$). If both conditions hold then it sets

$$\mathsf{ct}_i = \mathsf{FE.Enc}(\mathsf{mpk}_i, (\perp, K_i^*, \mathsf{k}_d); \$) \ .$$

Otherwise ct_i is computed as in the previous experiment.

To see why the adversary's view in this experiment is indistinguishable from the previous hybrid, we note that for any child of i, i.e., j such that $\mathsf{p}(j) = i$,

$$\mathsf{ReEnc}_{f_j, t_j, \mathsf{mpk}_j, c_j}(\tilde{f}_i(x_b^*), K_i^*, \perp) = \mathsf{ReEnc}_{f_j, t_j, \mathsf{mpk}_j, c_j}(\perp, K_i^*, \mathsf{k}_d) = \mathsf{ct}_j \ .$$

This is because necessarily $\mathsf{d}(j) = d + 1 > d$ and due to Invariant 4.5. Thus, the security of the $(\mathsf{msk}_{i,d}, \mathsf{mpk}_{i,d})$ key pair guarantees that this hybrid is indistinguishable from the previous one: Since ct_i is unobservable, then necessarily the adversary cannot access $\mathsf{msk}_{i,d}$ directly, but rather only via further delegation (i.e., via functional keys to $\mathsf{ReEnc}_{f_j, t_j, \mathsf{mpk}_j, c_j}$). This yields the following claim in rather straightforward manner:

Claim 4.6. *Assuming that $\mathcal{FE}$ is semi-adaptively secure, there exists a negligible function $\mathsf{neg}(\cdot)$ such that*

$$\left| \Pr\left[\mathcal{H}_4^{(0,0)}(\lambda) = 1\right] - \Pr[\mathcal{H}_3(\lambda) = 1] \right| \leq \mathsf{neg}(\lambda)$$

and

$$\left| \Pr\left[\mathcal{H}_4^{(i,d)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}_4^{(i-1,d)}(\lambda) = 1\right] \right| \leq \mathsf{neg}(\lambda)$$

for all $d \in \{0, \ldots, Q\}$ and $i \in \{1, \ldots, Q\}$, and for all sufficiently large $\lambda \in \mathbb{N}$.

<u>`End For` $\boldsymbol{i}$.</u>

Invariant 4.7. In the previous experiment, it should hold that all ciphertexts ct_i corresponding to unobservable queries such that $\mathsf{d}(i) \leq d$ are of the form $\mathsf{FE.Enc}(\mathsf{mpk}_i, (\perp, K_i^*, \mathsf{k}_{\mathsf{d}(i)}); \$)$ and further $K_i^* = \mathsf{PRF.Gen}(1^\lambda, \$)$.

Recall that our goal is to restore Invariant 4.5 for value $(d+1)$. To this end, we next need to replace r_j^* and s_j^* for all j such that $\mathsf{d}(j) = d + 1$, with random values (rather than values that are generated from $K_{\mathsf{p}(j)}^*$).

<u>`For` $\boldsymbol{j = 0, \ldots, Q}$:</u>

Invariant 4.8. This is similar to Invariant 4.7, but for all ciphertexts $\mathsf{ct}_{j'}$ corresponding to unobservable queries such that $j' < j$ and $\mathsf{d}(j') = d+1$ it holds that $r^*_{j'}$ and $s^*_{j'}$ had already been replaced with random.

<u>`For` $\boldsymbol{i = 0, \ldots, Q}$:</u>

Experiment $\mathcal{H}_5^{(i,j,d)}$. In this hybrid, we again change ct_i as follows. If ct_i is unobservable and $\mathsf{d}(i) = d$, then define $K^*_i = \mathsf{PRF.Gen}(1^\lambda, \$)$ (as before), $K^{\circledast}_i = \mathsf{PRF.Punc}(K^*_i, t^{(\mathrm{u})}_{j,d+1})$, $y_{i,j,d+1} = \mathsf{PRF.Eval}(K^*_i, t^{(\mathrm{u})}_{j,d+1})$. We now set:

$$\mathsf{ct}_i = \mathsf{FE.Enc}\left(\mathsf{mpk}_i, \left(\perp, \left(K^{\circledast}_i, (t^{(\mathrm{u})}_{j,d+1}, y_{i,j,d+1})\right), \mathsf{k}_{\mathsf{d}(i)}\right); \$\right) .$$

Namely, we replace the PRF key with a punctured key at the point $t^{(\mathrm{u})}_{j,d+1}$, and supply the value at that point[8]. We note that the functionality of $\mathsf{PRF.Eval}((K^{\circledast}_i, (t^{(\mathrm{u})}_{j,d+1}, y_{i,j,d+1})), \cdot)$ is identical to $\mathsf{PRF.Eval}(K^*_i, \cdot)$. The security of the key pair $(\mathsf{msk}_{i,d}, \mathsf{mpk}_{i,d})$ guarantees the indistinguishability of this hybrid (again relying on ct_i being unobservable and thus $\mathsf{msk}_{i,d}$ is not given to the adversary). This yields the following claim in rather straightforward manner:

Claim 4.9. *Assuming that $\mathcal{FE}$ is semi-adaptively secure, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{H}_5^{(0,0,d)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}_4^{Q,d}(\lambda) = 1\right]\right| \leq \mathsf{neg}(\lambda)$$

for all $d \in \{0, \ldots, Q\}$*, and*

$$\left|\Pr\left[\mathcal{H}_5^{(i,j,d)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}_5^{(i-1,j,d)}(\lambda) = 1\right]\right| \leq \mathsf{neg}(\lambda)$$

for all $d, j \in \{0, \ldots, Q\}$ *and* $i \in \{1, \ldots, Q\}$*, and for all sufficiently large* $\lambda \in \mathbb{N}$.

<u>`End For` $\boldsymbol{i}$.</u>

Invariant 4.10. In the current experiment, it holds that the PRF key for all depth-d ciphertexts which are unobservable had been punctured at point $t^{(\mathrm{u})}_{j,d+1}$, namely at the point on which it will be evaluated if indeed ct_j is of level $d+1$.

<u>`For` $\boldsymbol{i = 0, \ldots, Q}$:</u>

Experiment $\mathcal{H}_6^{(i,j,d)}$. In this hybrid, we again change ct_i in the case where ct_i is unobservable and $\mathsf{d}(i) = d$. The change from the previous experiment is only that now $y_{i,j,d+1} \leftarrow \$$, namely sampled randomly. We notice that already in the previous hybrid we never use K^* for unobservable queries, only the respective $K^{\circledast}$ and y values. Therefore swapping the y value to a completely random will be indistinguishable to the adversary by the punctured PRF property. This yields the following claim in rather straightforward manner:

[8] As discussed in Section 2.1, we find it natural to consider an "augmented" evaluation algorithm that outputs a pre-determined value at the punctured point. That is, the augmented evaluation algorithm is given an augmented key $(K^{\circledast}_i, (t^{(\mathrm{u})}_{j,d+1}, y_{i,j,d+1}))$, where $K^{\circledast}_i$ is punctured at $t^{(\mathrm{u})}_{j,d+1}$, and given an input t it outputs $\mathsf{PRF.Eval}_{K^{\circledast}_i}(t)$ if $t \neq t^{(\mathrm{u})}_{j,d+1}$, and it outputs $y_{i,j,d+1}$ if $t = t^{(\mathrm{u})}_{j,d+1}$.

Claim 4.11. *Assuming that $\mathcal{PRF}$ is a puncturable pseudorandom function, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{H}_6^{(0,j,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_5^{Q,j,d}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

and

$$\left|\Pr\left[\mathcal{H}_6^{(i,j,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_6^{(i-1,j,d)}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

for all $d,j\in\{0,\ldots,Q\}$ and $i\in\{1,\ldots,Q\}$, and for all sufficiently large $\lambda\in\mathbb{N}$.

<u>`End For` $\boldsymbol{i}$.</u>

Invariant 4.12. In the current experiment, it holds that the PRF key for all depth-d ciphertexts which are unobservable had been punctured at point $t_{j,d+1}$, and further that the punctured value had been substituted with random.

Experiment $\boldsymbol{\mathcal{H}_7^{(j,d)}}$. In this hybrid, we (finally) change the way ct_j is generated in the case where ct_j is unobservable and $\mathsf{d}(j)=d+1$ (if these conditions don't hold then we proceed as in the previous experiment). In particular, we change the way the randomness for ct_j and K_j^* is generated. Note that if ct_j is unobservable then it must be the case that $\mathsf{ct}_{\mathsf{p}(j)}$ is also unobservable (since $\tilde{f}_j=f_j\circ\tilde{f}_{\mathsf{p}(j)}$). In the previous experiment, we had

$$(s_j^*,r_j^*)=\mathsf{PRF.Eval}\left(\left(K_{\mathsf{p}(j)}^{\circledast},(t_{j,d+1}^{(\mathrm{u})},y_{\mathsf{p}(j),j,d+1})\right),t_{j,d+1}^{(\mathrm{u})}\right)\ .$$

We now define instead $(s_j',r_j')=y_{\mathsf{p}(j),j,d+1}$. We set $K_j^*=\mathsf{PRF.Gen}(1^\lambda,s_j')$ and

$$\mathsf{ct}_j=\mathsf{FE.Enc}\left(\mathsf{mpk}_j,\left(\tilde{f}_j(x_b^*),K_j^*,\perp\right);r_j'\right)\ .$$

The view of the adversary here remains exactly the same, since $(s_j',r_j')=(s_j^*,r_j^*)$. However, conceptually this means that (s_j^*,r_j^*) are detached from the value that is embedded in $\mathsf{ct}_{\mathsf{p}(i)}$. As we will see in the next experiment, we will remove $y_{i,j,d+1}$ from ct_i, but (s_j',r_j') will still be well defined. This yields the following observation:

Observation 4.13. *For all $\lambda\in\mathbb{N}$ it holds that*

$$\Pr\left[\mathcal{H}_7^{(j,d)}(\lambda)=1\right]=\Pr\left[\mathcal{H}_6^{Q,j,d}(\lambda)=1\right]$$

for all $d,j\in\{0,\ldots,Q\}$.

<u>`For` $\boldsymbol{i=0,\ldots,Q}$:</u>

Experiment $\boldsymbol{\mathcal{H}_8^{(i,j,d)}}$. In this hybrid, we again change ct_i in the case where ct_i is unobservable and $\mathsf{d}(i)=d$. We will now undo the puncturing of the PRF keys.

$$\mathsf{ct}_i=\mathsf{FE.Enc}\left(\mathsf{mpk}_i,\left(\perp,K_i^*,\mathsf{k}_{\mathsf{d}(i)}\right);\ \$\right)\ .$$

Indistinguishability holds since in all positions except $t_{j,d+1}^{(\mathrm{u})}$ the new and old keys, K_i^* and $\left(K_i^{\circledast},(t_{j,d+1},y_{i,j,d+1})\right)$ are functionally equivalent. Furthermore, the function $\mathsf{PRF.Eval}$ is never evaluated at $t_{j,d+1}^{(\mathrm{u})}$ (since if ct_j is unobservable then (r_j',s_j') are used instead of (r_j^*,s_j^*)). The functional encryption security of $(\mathsf{msk}_{i,d},\mathsf{mpk}_{i,d})$ therefore implies indistinguishability. This yields the following claim in rather straightforward manner:

Claim 4.14. *Assuming that $\mathcal{FE}$ is semi-adaptively secure, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{H}_8^{(0,j,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_7^{j,d}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

for all $d,j\in\{0,\ldots,Q\}$, and

$$\left|\Pr\left[\mathcal{H}_8^{(i,j,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_8^{(i-1,j,d)}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

for all $d,j\in\{0,\ldots,Q\}$ and $i\in\{1,\ldots,Q\}$, and for all sufficiently large $\lambda\in\mathbb{N}$.

`End For` $\boldsymbol{i}$.

The proof of the following claim is almost identical to that of Claim 4.9 and is therefore omitted:

Claim 4.15. *Assuming that $\mathcal{FE}$ is semi-adaptively secure, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{H}_8^{(Q,j,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_5^{(0,j+1,d)}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

for all $d\in\{0,\ldots,Q\}$ and $j\in\{0,\ldots,Q-1\}$, and for all sufficiently large $\lambda\in\mathbb{N}$.

`End For` $\boldsymbol{j}$.

The proof of the following claim is almost identical to that of Claim 4.6 and is therefore omitted:

Claim 4.16. *Assuming that $\mathcal{FE}$ is semi-adaptively secure, there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\left|\Pr\left[\mathcal{H}_8^{(Q,Q,d)}(\lambda)=1\right]-\Pr\left[\mathcal{H}_4^{(0,d+1)}(\lambda)=1\right]\right|\leq \mathsf{neg}(\lambda)$$

for all $d\in\{0,\ldots,Q-1\}$, and for all sufficiently large $\lambda\in\mathbb{N}$.

`End For` $\boldsymbol{d}$.

We now notice that the proof is practically finished, since the last hybrid $\mathcal{H}_8^{(Q,Q,Q)}$ is completely independent of the bit b. To see this, note that the only values that depend on b in the experiment are the values $\tilde{f}_i(x_b^*)$ that appear inside the ciphertexts ct_i (in particular inside the challenge ciphertext $\mathsf{ct}^*=\mathsf{ct}_0$). We first point out that the value $\tilde{f}_i(x_b^*)$ is in fact *independent* of b in observable ciphertexts, since by definition $\tilde{f}_i(x_0^*)=\tilde{f}_i(x_1^*)$. As for unobservable ciphertexts, in $\mathcal{H}_8^{(Q,Q,Q)}$ none of them contains $\tilde{f}_i(x_b^*)$ at all, as this value had been replaced by $\perp$. This yields the following observation:

Observation 4.17. *For all $\lambda\in\mathbb{N}$ it holds that*

$$\Pr\left[\mathcal{H}_8^{(Q,Q,Q)}(\lambda)=1\right]=\frac{1}{2}\ .$$

We presented a sequence of polynomially-many experiments starting with the experiment $\mathcal{H}_0=\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}$ and ending with the experiment $\mathcal{H}_8^{(Q,Q,Q)}$ which is completely independent of the bit b. We showed that the output distributions of each two consecutive experiments are negligibly close, which implies that there exists a negligible function $\mathsf{neg}(\cdot)$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda)&\stackrel{\mathsf{def}}{=}\left|\Pr\left[\mathsf{Exp}^{\mathsf{semiHFE}}_{\mathcal{HFE},\mathcal{A}}(\lambda)=1\right]-\frac{1}{2}\right|\\&=\left|\Pr\left[\mathcal{H}_0(\lambda)=1\right]-\Pr\left[\mathcal{H}_8^{(Q,Q,Q)}(\lambda)=1\right]\right|\\&\leq\ \mathsf{neg}(\lambda)\end{aligned}$$

for all sufficiently large $\lambda\in\mathbb{N}$, as required. ■

References

[ABB10a] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *Advances in Cryptology – EUROCRYPT '10*, pages 553–572, 2010.

[ABB10b] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *Advances in Cryptology – CRYPTO '10*, pages 98–115, 2010.

[ABG+13] P. Ananth, D. Boneh, S. Garg, A. Sahai, and M. Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013.

[ABS+15] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. From selective to adaptive security in functional encryption. In *Advances in Cryptology – CRYPTO '15*, pages 657–677, 2015.

[AGV+13] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption: New perspectives and lower bounds. In *Advances in Cryptology – CRYPTO '13*, pages 500–518, 2013.

[AJ15] P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *Advances in Cryptology – CRYPTO '15*, pages 308–326, 2015.

[AS15] G. Asharov and G. Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 191–209, 2015.

[BB04] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT '04*, pages 223–238, 2004.

[BBG05] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EUROCRYPT '05*, pages 440–456, 2005.

[BCP14] E. Boyle, K. Chung, and R. Pass. On extractability obfuscation. In *Proceedings of the 11th Theory of Cryptography Conference*, pages 52–73, 2014.

[BF03] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Preliminary version in *Advances in Cryptology – CRYPTO '01*, pages 213–229, 2001.

[BGG+14] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology – EUROCRYPT '14*, pages 533–556, 2014.

[BGI+12] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6, 2012.

[BGI14] E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*, pages 501–519, 2014.

[BKS16] Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In *Advances in Cryptology – EUROCRYPT '16*, pages 852–880, 2016.

[BO13] M. Bellare and A. O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *Proceedings of the 12th International Conference on Cryptology and Network Security*, pages 218–234, 2013.

[BS15] Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. In *Proceedings of the 12th Theory of Cryptography Conference*, pages 306–324, 2015.

[BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Proceedings of the 8th Theory of Cryptography Conference*, pages 253–273, 2011.

[BSW12] D. Boneh, A. Sahai, and B. Waters. Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

[BV15] N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science*, pages 171–190, 2015.

[BW06] X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – CRYPTO '06*, pages 290–307, 2006.

[BW13] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT '13*, pages 280–300, 2013.

[CGJ+15] N. Chandran, V. Goyal, A. Jain, and A. Sahai. Functional encryption: Decentralised and delegatable. Cryptology ePrint Archive, Report 2015/1017, 2015.

[CHK+12] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, 2012.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, 2001.

[CW14] J. Chen and H. Wee. Semi-adaptive attribute-based encryption and improved delegation for Boolean formula. In *Proceedings of the 9th International Conference on Security and Cryptography for Networks*, pages 277–297, 2014.

[DIJ+13] A. De Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano. On the achievability of simulation-based security for functional encryption. In *Advances in Cryptology – CRYPTO '13*, pages 519–535, 2013.

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*, pages 40–49, 2013.

[GGH+16] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Functional encryption without obfuscation. In *Proceedings of the 1th Theory of Cryptography Conference*, pages 480–511, 2016.

[GGM86] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[GH09] C. Gentry and S. Halevi. Hierarchical identity based encryption with polynomially many levels. In *Proceedings of the 6th Theory of Cryptography Conference*, pages 437–456, 2009.

[GKP+13] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 555–564, 2013.

[Gol04] O. Goldreich. Foundations of Cryptography – Volume 2: Basic Applications. Cambridge University Press, 2004.

[GS02] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology – ASIACRYPT '02*, pages 548–566, 2002.

[GVW12] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology – CRYPTO '12*, pages 162–179, 2012.

[GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, pages 545–554, 2013.

[HL02] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology – EUROCRYPT '02*, pages 466–481, 2002.

[KPT+13] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 20th Annual ACM Conference on Computer and Communications Security*, pages 669–684, 2013.

[KSY15] I. Komargodski, G. Segev, and E. Yogev. Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In *Proceedings of the 12th Theory of Cryptography Conference*, pages 352–377, 2015.

[LOS+10] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in Cryptology – EUROCRYPT '10*, pages 62–91, 2010.

[LW10] A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *Poceedings of the 7th Theory of Cryptography Conference*, pages 455–479, 2010.

[LW11] A. B. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. In *Advances in Cryptology – EUROCRYPT '11*, pages 547–567, 2011.

[LW14] A. B. Lewko and B. Waters. Why proving HIBE systems secure is difficult. In *Advances in Cryptology – EUROCRYPT '14*, pages 58–76, 2014.

[O'N10] A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – CRYPTO '84*, pages 47–53, 1984.

[SW08] A. Sahai and B. Waters. Slides on functional encryption. Available at `http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt`, 2008.

[SW14] A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 475–484, 2014.

[Wat09] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology – CRYPTO '09*, pages 619–636, 2009.

[Wat15] B. Waters. A punctured programming approach to adaptively secure functional encryption. In *Advances in Cryptology – CRYPTO '15*, pages 678–697, 2015.