Function-Private Functional Encryption in the Private-Key Setting

Zvika Brakerski* Gil Segev†

Abstract

Functional encryption supports restricted decryption keys that allow users to learn specific functions of the encrypted messages. Although the vast majority of research on functional encryption has so far focused on the privacy of the encrypted *messages*, in many realistic scenarios it is crucial to offer privacy also for the *functions* for which decryption keys are provided.

Whereas function privacy is inherently limited in the public-key setting, in the private-key setting it has a tremendous potential. Specifically, one can hope to construct schemes where encryptions of messages $\mathsf{m}_1, \ldots, \mathsf{m}_T$ together with decryption keys corresponding to functions $f_1, \ldots, f_T$, reveal essentially no information other than the values $\{f_i(\mathsf{m}_j)\}_{i,j \in [T]}$. Despite its great potential, the known function-private private-key schemes either support rather limited families of functions (such as inner products), or offer somewhat weak notions of function privacy.

We present a generic transformation that yields a *function-private* functional encryption scheme, starting with any *non-function-private* scheme for a sufficiently rich function class. Our transformation preserves the message privacy of the underlying scheme, and can be instantiated using a variety of existing schemes. Plugging in known constructions of functional encryption schemes, we obtain function-private schemes based either on the Learning with Errors assumption, on obfuscation assumptions, on simple multilinear-maps assumptions, and even on the existence of any one-way function (offering various trade-offs between security and efficiency).

*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 Seventh Framework Programme (FP7) via a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), and by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11).

1 Introduction

The most classical cryptographic scenario, dating back thousands of years, consists of two parties who wish to secretly communicate in the presence of an eavesdropper. This classical scenario has traditionally led the cryptographic community to view the security provided by encryption schemes as an *all-or-nothing* guarantee: The encrypted data can be fully recovered using the decryption key, but it is completely useless without it. In a wide variety of modern scenarios, however, a more fine-grained approach is urgently needed. Starting with the seminal notion of identity-based encryption [Sha84, BF03, Coc01], this need has recently motivated the cryptographic community to develop a vision of *functional encryption* [SW08, BSW11, O'N10], allowing tremendous flexibility when accessing encrypted data.

Functional encryption supports restricted decryption keys that allow users to learn specific functions of the encrypted data and nothing else. More specifically, 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 some message m, can compute $f(\mathsf{m})$ but cannot learn any additional information about the message m. Extensive research has recently been devoted to studying the security of functional encryption schemes and to constructing such schemes (see, for example, [SW08, BSW11, O'N10, GVW12, AGV+13, BO13, BCP14, DIJ+13, GGH+13, GKP+13a] and the references therein).

Function privacy in functional encryption. The vast majority of research on functional encryption to date has focused on the privacy of encrypted *messages*. In various scenarios, however, one should consider not only privacy for the encrypted messages but also privacy for the *functions* for which functional keys are provided. Consider, for example, a user who subscribes to an on-line storage service for storing her files. For protecting her privacy, the user locally encrypts her files using a functional encryption scheme prior to uploading them to the service. The user can then remotely query her data by providing the service with a functional key sk_f corresponding to any query f. Without any additional privacy guarantees, the functional key sk_f may entirely reveal the user's query f to the service, which is clearly undesirable whenever the query itself contains sensitive information.

Scenarios of such flavor have motivated the research of *function privacy* in functional encryption, first in the private-key setting by Shen, Shi and Waters [SSW09], and very recently in the public-key setting by Boneh, Raghunathan and Segev [BRS13a, BRS13b] followed by Agrawal et al. [AAB+13]. Intuitively, function privacy requires that functional keys reveal no unnecessary information on their functionality.

The extent to which function privacy can be satisfied differs dramatically between the settings of private-key and public-key encryption. Specifically, in the public-key setting, where anyone can encrypt messages, only a limited form of function privacy can be satisfied. This is because given a functional key sk_f and the public key pk of the scheme, a malicious user can learn information about the function f by evaluating it on any point m of his choosing (by first encrypting m and then using sk_f to decrypt $f(\mathsf{m})$). This process reveals non-trivial information about f and in some cases may entirely leak the function's description (unless additional restrictions are imposed, see [BRS13a, BRS13b, AAB+13] for more details). As a result, function-private functional encryption schemes in the public-key setting are quite restricted, and furthermore such have only been presented respective to limited function families (e.g., point functions and inner products).

Our work: Function privacy in the private-key setting. In this work, we focus on function

privacy in the private-key setting. In this setting, function privacy has significantly more potential than in the public-key setting, both as a stand-alone feature and as a very useful building block (see Section 1.2 for subsequent work). Specifically, one can hope to achieve the following notion of privacy (stated informally): Any user that obtains encryptions of messages $\mathsf{m}_1, \ldots, \mathsf{m}_T$, and functional keys corresponding to functions $f_1, \ldots, f_T$, learns essentially no information other than the values $\{f_i(\mathsf{m}_j)\}_{i,j\in[T]}$. This is a strong notion of privacy which has great (realistic) potential for a wide variety of applications.

Despite its great potential, the known function-private private-key schemes either support only the inner-product functionality for attribute-based encryption [SSW09, AAB+13], or offer only somewhat weak notions of function privacy [GKP+13a, GGG+14]. We refer the reader to Section 1.3 for a detailed discussion of the known function private schemes. This state of affairs has motivated us to explore the following fundamental question:

> Can we construct private-key functional encryption schemes that support *rich and expressive* families of functions while offering *strong notions* of function privacy?

1.1 Our Contributions

Our work provides a positive answer to the above fundamental question. We present private-key functional encryption schemes that support rich and expressive families of functions, while offering strong notions of function privacy.

Specifically, we put forward a generic transformation that yields a *function-private* private-key functional encryption scheme based on any (possibly *non-function-private*) private-key functional encryption scheme that supports all functions that are computable by bounded-size circuits. In particular, our transformation can be instantiated by the recently developed functional encryption scheme of Goldwasser et al. [GKP+13a] that is based on the LWE assumption, by the schemes of Garg et al. [GGH+13], Boyle et al. [BCP14], Ananth et al. [ABG+13], and Waters [Wat14] that are based on obfuscation assumptions, by the scheme of Garg et al. [GGH+14] that is based on simple assumptions on multilinear maps, and even by the scheme of Gorbunov et al. [GVW12] which is somewhat less efficient but can be based on any one-way function (we refer the reader to Section 2.2.1 for more details). Although most of these constructions are in fact public-key schemes, they are in particular private-key ones (i.e., in some sense, these schemes currently seem significantly more powerful than what is required for our transformation).

The notions of function privacy that are satisfied by our transformation are the strongest notions that have been proposed so far (we refer the reader to Section 3 for a detailed discussion of these notions). In addition, the resulting scheme inherits the message privacy of the underlying scheme (i.e., full vs. selective security, and one-key vs. many-keys security), and supports all functions that are computable by circuits whose size is slightly smaller than those supported by the underlying scheme. Finally, we note that our transformation is in fact oblivious to the computational model that is supported by the underlying scheme and to its representation (e.g., circuits vs. Turing machines), as long as the scheme supports a universal function for the model and a few additional basic operations (see Section 1.4 below).

1.2 Subsequent Work

Our generic construction and proof techniques have already been proved fruitful by Komargodski et al. [KSY15] and by Ananth et al. [ABS+14] beyond the context of function-private functional encryption as its own primitive. Komargodski et al. presented a construction of a private-key functional encryption scheme for *randomized* functions based on *any* private-key functional encryption scheme

for *deterministic* functions that is sufficiently expressive. Their work follows-up on the work of Goyal et al. [GJK+13] who put forward the notion of functional encryption for randomized functionalities, and constructed a public-key functional encryption scheme for randomized functionalities based on the (seemingly significantly stronger) assumption of indistinguishability obfuscation. Ananth et al. presented a construction of a *fully-secure* private-key functional encryption scheme from any *selectively-secure* private-key functional encryption scheme that is sufficiently expressive. Previous constructions of fully-secure schemes were based on assumptions that seem significantly stronger, such as obfuscation and multilinear maps assumptions [BCP14, ABG+13, Wat14, GGH+14].

One of the key insights underlying both of these works is that in the private-key setting, where encryption is performed honestly by the owner of the master secret key, the power of obfuscation may not be needed. Instead, they observed that in some cases one can rely on the weaker notion of function privacy. More specifically, both Komargodski et al. and Ananth et al. showed that any sufficiently expressive functional encryption scheme may be appropriately utilized via our function-privacy techniques for implementing some of the proof techniques that were so far implemented based on obfuscation (including, for example, a variant of the punctured programming approach of Sahai and Waters [SW14]).

1.3 Additional Related Work

Function privacy. As mentioned above, Shen, Shi and Waters [SSW09] initiated the research on predicate privacy in attribute-based encryption in the private-key setting. They constructed a predicate-private inner-product encryption scheme in the private-key setting. Boneh, Raghunathan and Segev [BRS13a, BRS13b] initiated the research on function privacy in the public key setting. They constructed function-private public-key functional encryption schemes for point functions (equivalently, anonymous IBE) and for subspace membership. Since their work is in the public-key setting, their framework assumes that the functions come from a distribution of sufficient entropy, as otherwise it seems that no realistic notion of function privacy can be satisfied.

Agrawal et al. [AAB+13] then presented a general framework for function-private functional encryption both in the private-key setting and in the public-key setting, and explore their plausibility. Most relevant to our work, they presented the *full security* notion for function-private functional encryption in the private-key setting and presented improved constructions for the inner-product functionality in this model. We note that we refer to their notion of full security as *full function privacy* (see Definition 3.2).

Reusable garbled circuits. The related notion of *reusable garbled circuits* (ruGC) is defined as follows. Given a secret key, two procedures can be carried out: Garbling a circuit C (which corresponds to generating a function key) and encoding of an input x (which corresponds to an encryption of a message). Given a garbled C and an encoded input x, it is possible to publicly compute $C(x)$. The security requirement is that an adversary that chooses C to be garbled and then a sequence of inputs $x_1, \ldots, x_t$ to be encoded cannot learn more than $C(x_1), \ldots, C(x_t)$. Security is formalized in a simulation-based model: The simulator is required to simulate the garbled circuit without knowing C, and then it is fed with $C(x_i)$ in turn and is required to simulate the encoded inputs. Goldwasser et al. [GKP+13a, GKP+13b] constructed a simulation-secure functional encryption scheme (without function privacy) and showed how ruGC follows from that primitive[1]. The similarity to function-private functional encryption is apparent, but there are some significant differences. It follows from the result of [AGV+13] that ruGC, much like simulation secure functional encryption,

[1] Additional constructions were presented by Boneh et al. [BGG+14] who were able to reduce the garbling overhead from multiplicative to additive in either the size of the circuit or the size of the encoded input.

cannot securely support an a-priori unbounded number of circuits, whereas we are able to guarantee function privacy for any polynomial (a-priori unknown) number of function keys. A very similar argument shows that the situation where C is chosen after the inputs x_i is also impossible in the context of ruGC (at least under a natural definition of the simulation process), whereas we would like the inputs and functions to be adaptively introduced in arbitrary order. On the flip side, ruGC provides simulation based security which seems to be a stronger notion than indistinguishability-based security achieved by our construction.

Multi-input functional encryption. Goldwasser et al. [GGG+14] have recently introduced the notion of *multi-input functional encryption* (MIFE) schemes. As the name suggests, MIFE allows functional keys to correspond to multi-input functions which can be evaluated on tuples of ciphertexts. Slightly simplified, the dream notion of security (specifically, indistinguishability-based security in the private-key setting, which is most relevant to this work) is that of an adversary that is allowed to make functional key queries and also message queries containing pairs of messages (m_0, m_1), and in response it gets an encryption of m_b, where b is a secret bit. We would like the adversary to not be able to guess b unless it obtained a key to a function that behaves differently on the m_0's and on the m_1's. This dream version of security, even just for two inputs, implies function-private private-key functional encryption: We will use the first input coordinate to encrypt the description of the function and the second to encrypt the input, and provide a function key for the universal 2-input function. However, Goldwasser et al. [GGG+14] fall short of achieving this dream version, since their adversary is not allowed to make message queries adaptively. Furthermore, their construction relies on strong notions of obfuscation (indistinguishability obfuscation and differing input obfuscation), whereas the construction in this paper only relies on private-key function encryption (which is currently known to be implied by obfuscation, but no reverse derivation is known and it is quite possible that they can be constructed under milder assumptions – see Section 2.2.1).

1.4 Overview of Our Approach

In this section we provide a high-level overview of our approach and techniques. We begin with a brief description of the notions of function privacy that we consider in this work, and then describe the main ideas underlying our construction.

Function privacy. Our notion of function privacy is that of Agrawal et al. [AAB+13, Def. 2.7] (generalizing Shen, Shi and Waters [SSW09]), which considers the privacy of functional keys and the privacy of encrypted messages in a completely symmetric manner. Specifically, we consider adversaries that issue both key-generation queries of the form (f_0, f_1) and encryption queries of the form $(\mathsf{m}_0, \mathsf{m}_1)$. These queries are answered by providing a functional key for f_b and an encryption of m_b, where all queries are answered using the same bit $b \in \{0, 1\}$. We allow adversaries to adaptively issue any polynomial number of such queries (this number does not have to be bounded in advance), and their goal is to distinguish the experiment where $b = 0$ and the experiment where $b = 1$. Our only requirement from such adversaries is that for all key-generation queries (f_0, f_1) and for all encryption queries $(\mathsf{m}_0, \mathsf{m}_1)$ it holds that $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$. In addition to this notion, we also consider two "selective" relaxations, and we refer the reader to Section 3 for more details on our notions of function privacy.

A failed attempt. Our starting point is any given private-key functional encryption scheme without function privacy. A natural approach towards achieving function privacy is to modify its key-generation algorithm so that it provides functional keys containing only *encrypted* descriptions of the associated functions. Namely, for generating a functional key for a function f, we will

first encrypt the description of f using a symmetric encryption scheme $\mathcal{SKE} = (\mathsf{SKE.KG}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$ to obtain a ciphertext $c_f \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f)$, where $\mathsf{SKE.k}$ is a key for the scheme $\mathcal{SKE}$ (which does not change throughout the lifespan of the scheme). Then, the key-generation algorithm would provide a functional key for the function $U_{c_f}(\mathsf{m}, k) \stackrel{\mathsf{def}}{=} (\mathsf{SKE.Dec}(k, c_f))(\mathsf{m})$ (that is, the function that first decrypts c_f using the candidate key k, and then applies the resulting function on m). The semantic security of $\mathcal{SKE}$ guarantees that the function U_{c_f} hides the description of f, as long as the key $\mathsf{SKE.k}$ is not known. In order to maintain the functionality, the message encryption algorithm must also change: Rather than encrypting the message m alone using the underlying functional encryption scheme, we will now encrypt the pair $(m, \mathsf{SKE.k})$. One can verify that the functionality of the scheme still holds since clearly $U_{c_f}(\mathsf{m}, \mathsf{SKE.k}) = f(\mathsf{m})$.

One could hope to prove that this construction is function private. Indeed, Goldwasser et al. [GKP+13a] used this exact scheme to construct reusable garbled circuits. This approach by itself, however, is insufficient for our purposes. On one hand, during the proof of security we would like to rely on the semantic security of $\mathcal{SKE}$ for arguing that the function U_{c_f} hides the description of f. This implies that the key $\mathsf{SKE.k}$ should be kept completely secret. On the other hand, the functionality of the scheme must be preserved even during the proof of security. Thus, in order to allow adversaries to use the functional key for the function U_{c_f}, the key $\mathsf{SKE.k}$ must be used while encrypting messages as above. This conflict is the main challenge that our construction overcomes.

We note that also in the construction of reusable garbled circuits of Goldwasser et al. [GKP+13a] this conflict arises. However, they consider only a *selective single-function* security notion asking adversaries to specify a challenge function f prior to receiving any encryptions. Within such a selective framework, the conflict is easily resolved: During the proof of security one can preserve the functionality by modifying the encryption algorithm to encrypt $f(\mathsf{m})$ instead of encrypting m itself. Thus, the description of the function f is in fact not needed, but only the value $f(\mathsf{m})$ is needed for each encrypted message m (note that $f(\mathsf{m})$ is anyway known to the adversary). This approach, however, seems inherently limited to a selective framework, whereas we would like to allow adversaries to adaptively query the key-generation and encryption oracles, at any point in time, and for any polynomial number of queries[2].

Our scheme. To get around the aforementioned obstacle, we show that the Naor-Yung "double encryption" methodology [NY90] can be adapted to our setting. Instead of encrypting the description of f only once, we encrypt it twice using two independent symmetric keys. For preserving the functionality of the system, only one out of the two keys will be explicitly needed, and this allows us to attack the other key. Combined with the message privacy of the underlying functional encryption scheme, this approach enables us to prove the security of our scheme.

More specifically, the master secret key of our scheme consists of a master secret key msk for the underlying functional encryption scheme, and two keys, $\mathsf{SKE.k}$ and $\mathsf{SKE.k}'$, for a symmetric-key CPA-secure scheme. In order to generate a functional key for a function f, we first generate two symmetric encryptions $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f)$ of the description of f. Then, we issue a functional key for the function $U_{c,c'}$ which is defined as follows on inputs of the form $(\mathsf{m}, \mathsf{m}', k, k')$: If $k \neq \perp$ then decrypt c using k for obtaining a function f, and output $f(\mathsf{m})$. Otherwise, if $k = \perp$, then decrypt c' using k' for obtaining a function f', and output $f(\mathsf{m}')$. In order to encrypt a message m, we will use the encryption scheme of the underlying functional encryption

[2]The approach of Goldwasser et al. can be extended to deal with any a-priori bounded number of functions, as long as they are specified in advance (this is done using [GVW12]). In this case, the length of ciphertexts in their scheme would be linear in the number of functions. This is in fact inherent to their setting, as they consider a simulation-based notion of security [AGV+13]. We consider indistinguishability-based notions of security, and would like to inherit the (either full or selective) security of the underlying functional encryption scheme.

scheme to encrypt $(\mathsf{m}, \perp, \mathsf{SKE.k}, \perp)$ using its master secret key msk. Note that this scheme works quite similarly to the aforementioned intuitive idea, only it has "placeholders" for elements that will be used in the proof.

Towards illustrating some of the ideas underlying the proof of security, consider an adversary that makes just one encryption query $(\mathsf{m}_0, \mathsf{m}_1)$, and just one key-generation query (f_0, f_1) in some *arbitrary* order (recall that we require $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$). The view of this adversary consists of an encryption of m_b and a functional key for f_b for a uniformly chosen bit $b \in \{0,1\}$. The proof starts by modifying the functional key: Instead of computing $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_b)$ we compute $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$. Note that since the key $\mathsf{SKE.k}'$ is in fact not being used, and since the functionality of the functional key is not hurt (c' is anyway not used for decryption), the CPA-security of the symmetric scheme implies that this goes unnoticed. Next, we modify the encryption algorithm to encrypt $(\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}')$ instead of $(\mathsf{m}_b, \perp, \mathsf{SKE.k}, \perp)$. This time the adversary will not notice the change due to the message-privacy of the underlying functional encryption scheme, since the new and old ciphertexts will decrypt to the same value $f_b(\mathsf{m}_b) = f_1(\mathsf{m}_1)$ under the modified functional key. Finally, we modify the functional key once again: Instead of computing $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_b)$ we compute $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_1)$. As before, since the key $\mathsf{SKE.k}$ is in fact not being used, and since the functionality of the functional key is not hurt, then the CPA-security of the symmetric scheme implies that this goes unnoticed. At this point, we observe that the view of the adversary is in fact completely independent of the choice of the bit b, and the result follows. We refer the reader to Section 4 for the formal description and proof of our scheme.

1.5 Open Problems

Our work raises various open problems on the feasibility and the design of functional encryption schemes. Some of these are outlined below.

Private-key vs. public-key functional encryption. Our construction relies on any private-key functional encryption schemes, but in fact, most of the existing constructions of such schemes are secure even in the public-key setting (see Section 2.2.1). Clearly, any functional encryption scheme that is secure in the public-key setting is also secure in the private-key one. However, the existing constructions either apply in a restricted setting or rely on somewhat strong assumptions that are related to program obfuscation. Our work provides additional motivation for studying private-key FE in hope for achieving constructions with better efficiency or under improved assumptions. Alternatively, perhaps it is possible to construct a public-key functional encryption scheme based on any private-key scheme.

Simulation-based function privacy. Following Shen, Shi and Waters [SSW09] and Boneh, Raghunathan and Segev [BRS13a, BRS13b], we consider indistinguishability-based notions of function privacy. As already observed [BSW11, O'N10], in some cases indistinguishability-based notions do not provide realistic security guarantees for functional encryption schemes. A (somewhat relaxed) simulation-based notion of function privacy was recently formalized by Agrawal et al. [AAB+13], and an interesting open problem is to further explore its relation to our notions and to our construction.

Relying on restricted function families. Our construction relies on any private-key functional encryption scheme that supports a sufficiently rich function class. Although, as discussed above, various such schemes are known to exist, an interesting open problem is to construct a function-private scheme based on any scheme that supports more restricted function classes (e.g., inner products or subspace membership).

1.6 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce the basic notation and tools underlying our construction. In Section 3 we introduce the notions of function privacy that are considered in this work. In Section 4 we present and prove the security of our generic construction of a function-private 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 $\{1, \ldots, n\}$. A real function over the natural numbers is *negligible* if it vanishes faster than the inverse of any polynomial.

We rely on the following standard notion of a *left-or-right oracle* when formalizing the security of encryption schemes:

Definition 2.1 (Left-or-right oracle)**.** Let $\mathcal{O}$ be a probabilistic two-input functionality. For each $b \in \{0,1\}$ we denote by $\mathcal{O}_b$ the probabilistic three-input functionality $\mathcal{O}_b(k, x_0, x_1) \stackrel{\mathsf{def}}{=} \mathcal{O}(k, x_b)$.

2.1 Private-Key Encryption

A private-key encryption scheme over a message space $\mathcal{M}$ is a triplet $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ 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 $\mathsf{m} \in \mathcal{M}$, and outputs a ciphertext c. The decryption algorithm Dec takes as input a secret key k and a ciphertext c, and outputs a message m 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 $\mathsf{m} \in \mathcal{M}$ it holds that $\mathsf{Dec}(\mathsf{k}, \mathsf{Enc}(\mathsf{k}, \mathsf{m})) = \mathsf{m}$ with probability 1 over the internal randomness of the algorithms Enc and Dec.

In terms of security, we rely on the standard notion of a CPA-secure private-key encryption scheme that is formulated using a left-or-right encryption oracle. Recall (Definition 2.1) that for an encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and for any $b \in \{0,1\}$ we denote by Enc_b the left-or-right encryption oracle $\mathsf{Enc}_b(\mathsf{k}, \mathsf{m}_0, \mathsf{m}_1) \stackrel{\mathsf{def}}{=} \mathsf{Enc}(\mathsf{k}, \mathsf{m}_b)$.

Definition 2.2 (CPA security)**.** A private-key encryption scheme $\Pi = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is *CPA-secure* if for any probabilistic polynomial-time adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{CPA}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left| \Pr\left[\mathcal{A}^{\mathsf{Enc}_0(\mathsf{k},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Enc}_1(\mathsf{k},\cdot,\cdot)}(\lambda) = 1\right] \right| \leq \nu(\lambda),$$

where the probability is taken over the choice of $\mathsf{k} \leftarrow \mathsf{KG}(1^\lambda)$ and over the randomness of $\mathcal{A}$.

2.2 Private-Key Functional Encryption

We rely on the standard indistinguishability-based notions of full security and selective security for functional encryption schemes (see, for example, [BSW11, O'N10, BO13]), by adapting them to the private-key setting. In this paper, as we consider security notions for both messages and keys, we refer to the standard notions of security and selective security as *message privacy* and *selective-message privacy*.

A private-key functional encryption scheme over a message space $\mathcal{M}$ and a function space $\mathcal{F}$ is a quadruple $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ 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. The key-generation algorithm KG takes as input a master-secret key msk and a function $f \in \mathcal{F}$, and outputs a functional key sk_f. The encryption algorithm Enc takes as input a master-secret key msk and a message $\mathsf{m} \in \mathcal{M}$, and outputs a ciphertext c. In terms of correctness we require that for every function $f \in \mathcal{F}$ and message $\mathsf{m} \in \mathcal{M}$ it holds that $\mathsf{Dec}(\mathsf{KG}(\mathsf{msk}, f), \mathsf{Enc}(\mathsf{msk}, \mathsf{m})) = f(\mathsf{m})$ with all but a negligible probability over the internal randomness of the algorithms $\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}$, and Dec.

In terms of message privacy we require that encryptions of any two adversarially-chosen messages, m_0 and m_1, are computationally indistinguishable for any adversary that may adaptive obtain functional keys for any function $f \in \mathcal{F}$ as long as $f(\mathsf{m}_0) = f(\mathsf{m}_1)$. This is formalized via the following definitions. Recall (Definition 2.1) that for a private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ and for any $b \in \{0,1\}$ we denote by Enc_b the left-or-right encryption oracle $\mathsf{Enc}_b(\mathsf{msk}, \mathsf{m}_0, \mathsf{m}_1) \stackrel{\mathsf{def}}{=} \mathsf{Enc}(\mathsf{msk}, \mathsf{m}_b)$.

Definition 2.3 (Valid message-privacy adversary)**.** A probabilistic polynomial-time algorithm $\mathcal{A}$ is a *valid message-privacy adversary* if for all private-key functional encryption schemes $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, for all $\lambda \in \mathbb{N}$ and $b \in \{0,1\}$, and for all f and $(\mathsf{m}_0, \mathsf{m}_1)$ with which $\mathcal{A}$ queries the oracles KG and Enc_b, respectively, it holds that $f(\mathsf{m}_0) = f(\mathsf{m}_1)$.

Definition 2.4 (Full message privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ is *fully message private* if for any valid message-privacy adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{MP}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{A}^{\mathsf{KG}(\mathsf{msk},\cdot),\mathsf{Enc}_0(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}(\mathsf{msk},\cdot),\mathsf{Enc}_1(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right]\right| \leq \nu(\lambda),$$

where the probability is taken over the choice of $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ and over the randomness of $\mathcal{A}$.

Definition 2.5 (Selective-message message privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ is *T-selective-message message private*, where $T = T(\lambda)$, if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda, 1) = 1\right]\right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda, b)$ is defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $((\mathsf{m}_{0,1}, \ldots, \mathsf{m}_{0,T}), (\mathsf{m}_{1,1}, \ldots, \mathsf{m}_{1,T}), \mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$, where $\mathsf{m}_{0,i}, \mathsf{m}_{1,i} \in \mathcal{M}_\lambda$ for all $i \in [T]$.
3. $\mathsf{c}^*_i \leftarrow \mathsf{Enc}(\mathsf{msk}, \mathsf{m}_{b,i})$ for all $i \in [T]$.
4. $b' \leftarrow \mathcal{A}^{\mathsf{KG}(\mathsf{msk},\cdot)}(\mathsf{c}^*_1, \ldots, \mathsf{c}^*_T, \mathsf{state})$, where for each of $\mathcal{A}$'s queries f to $\mathsf{KG}(\mathsf{msk}, \cdot)$ it holds that $f(\mathsf{m}_{0,i}) = f(\mathsf{m}_{1,i})$ for all $i \in [T]$.
5. Output b'.

Such a scheme Π is *selective-message message private* if it is T-selective-message message private for all polynomials $T = T(\lambda)$.

Definition 2.6 (Selective-function message privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *T-selective-function message private*, where $T = T(\lambda)$, if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{sfMP}}_{\Pi,\mathcal{A},T}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{sfMP}}_{\Pi,\mathcal{A},T}(\lambda,0)=1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{sfMP}}_{\Pi,\mathcal{A},T}(\lambda,1)=1\right]\right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{\mathsf{sfMP}}_{\Pi,\mathcal{A},T}(\lambda,b)$ is defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $(f_1,\ldots,f_T,\mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$, where $f_i \in \mathcal{F}_\lambda$ for all $i \in [T]$.
3. $\mathsf{sk}_{f_i} \leftarrow \mathsf{KG}(\mathsf{msk}, f_i)$ for all $i \in [T]$.
4. $b' \leftarrow \mathcal{A}^{\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)}(\mathsf{sk}_{f_1},\ldots,\mathsf{sk}_{f_T},\mathsf{state})$, where for each of $\mathcal{A}$'s queries $(\mathsf{m}_0,\mathsf{m}_1)$ to $\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)$ it holds that $f_i(\mathsf{m}_0) = f_i(\mathsf{m}_1)$ for all $i \in [T]$.
5. Output b'.

Such a scheme Π is *selective-function message private* if it is T-selective-function message private for all polynomials $T = T(\lambda)$.

2.2.1 Known Instantiations

Private-key functional encryption schemes that satisfy the notions presented in Definitions 2.4–2.6 (and support circuits of any a-priori bounded polynomial size) are known to exist based on various assumptions. Most of the known schemes are in fact *public-key* schemes, which are in particular private-key ones[3]. Each of these scheme can be used to instantiate our generic transformation.

Specifically, a scheme that satisfies our notion of selective-message message privacy was constructed by Garg et al. [GGH+13] based on indistinguishability obfuscation. Schemes that satisfy the stronger notion of full message privacy (Definition 2.4) were constructed by Boyle et al. [BCP14] and by Ananth et al. [ABG+13] based on differing-input obfuscation, by Waters [Wat14] based on indistinguishability obfuscation, and by Garg et al. [GGH+14] based on multilinear maps. Moreover, a generic transformation from selective-message message privacy to full message privacy was recently showed by Ananth et al. [ABS+14].

A scheme that satisfies the notion of 1-selective-function message privacy was constructed by Gorbunov, Vaikuntanathan and Wee [GVW12] under the sole assumption that public-key encryption exists. In the private-key setting, their transformation can in fact rely on any private-key encryption scheme (and thus on any one-way function). By assuming, in addition, the existence of a pseudorandom generator computable by small-depth circuits (which is known to be implied by most concrete intractability assumptions), they construct a scheme that satisfies the notion of T-selective-function message privacy for any predetermined polynomial $T = T(\lambda)$. However, the length of the ciphertexts in their scheme grows linearly with T and with an upper bound on the circuit size of the functions that the scheme allows (which also has to be known ahead of time). Goldwasser et al. [GKP+13a] showed that based on the Learning with Errors (LWE) assumption, T-selective-function message privacy can be achieved where the ciphertext size grows with T and with a bound on the depth of allowed functions.

[3]For indistinguishability-based message privacy in the public-key setting, considering one challenge is equivalent to considering a left-or-right encryption oracle [GVW12]. Therefore, as public-key schemes are also private-key ones, in our indistinguishability-based definitions we directly consider left-or-right encryption oracles.

3 Modeling Function Privacy in the Private-Key Setting

In this section we introduce the notions of function privacy that are considered in this work. We consider three notions: *full function privacy*, *selective-message function privacy*, and *selective-function function privacy*. These are indistinguishability-based notions whose goal is to guarantee that functional keys reveal no unnecessary information on their functionality. Specifically, these notions ask that any efficient adversary that obtains encryptions of messages $\mathsf{m}_1, \ldots, \mathsf{m}_T$, and functional keys corresponding to functions $f_1, \ldots, f_T$, learns essentially no information other than the values $\{f_i(\mathsf{m}_j)\}_{i,j\in[T]}$. Our notions generalize the standard message-privacy notions for functional encryption (see Section 2.2) by taking into account function privacy *in addition* to message privacy.

Full function privacy. The strongest notion that we consider, which we refer to as *full function privacy*, was recently put forward by Agrawal et al. [AAB+13] who generalized the notion of Shen, Shi and Waters [SSW09] for predicate-privacy in attribute-based encryption. This notion considers both privacy of functional keys and privacy of encrypted messages in a completely symmetric manner. Specifically, we consider adversaries that interact with a left-or-right key-generation oracle $\mathsf{KG}_b(\mathsf{msk}, \cdot, \cdot)$, and with a left-or-right encryption oracle $\mathsf{Enc}_b(\mathsf{msk}, \cdot, \cdot)$ (where both oracles operate using the same bit b)[4]. We allow adversaries to adaptively interact with these oracles for any polynomial number of queries (which does not have to be bounded in advance), and the adversaries' goal is to distinguish the cases $b = 0$ and $b = 1$. Our only requirement from such adversaries is that for all (f_0, f_1) and $(\mathsf{m}_0, \mathsf{m}_1)$ with which they query the oracles KG_b and Enc_b, respectively, it holds that $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$. We note that this is clearly an inherent requirement.

Definition 3.1 (Valid function-privacy adversary)**.** A probabilistic polynomial-time algorithm $\mathcal{A}$ is a *valid function-privacy adversary* if for all private-key functional encryption schemes $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$, for all $\lambda \in \mathbb{N}$ and $b \in \{0, 1\}$, and for all (f_0, f_1) and $(\mathsf{m}_0, \mathsf{m}_1)$ with which $\mathcal{A}$ queries the oracles KG_b and Enc_b, respectively, the following three conditions hold:

1. $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$.
2. The messages m_0 and m_1 have the same length.
3. The descriptions of the functions f_0 and f_1 have the same length.

Definition 3.2 (Full function privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *fully function private* if for any valid function-privacy adversary $\mathcal{A}$, there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{FP}}_{\Pi,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{A}^{\mathsf{KG}_0(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_0(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}_1(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_1(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right]\right| \leq \nu(\lambda),$$

where the probability is taken over the choice of $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ and over the randomness of $\mathcal{A}$.

Selective notions. We consider two relaxations of our notion of full function privacy from Definition 3.2. The first, which we refer to as *selective-message function privacy* restricts the access that adversaries have to the left-or-right encryption oracle. Specifically, this notion asks that adversaries choose in advance their set of encryption queries. We note that adversaries are still given oracle access to the left-or-right key-generation oracle, with which they can interact in an adaptive manner for any polynomial number of queries. The second, which we refer to as *selective-function function*

[4]Recall (Definition 2.1) that for a probabilistic two-input functionality $\mathcal{O}$ and for $b \in \{0, 1\}$, we denote by $\mathcal{O}_b$ the probabilistic three-input functionality $\mathcal{O}_b(k, x_0, x_1) \stackrel{\mathsf{def}}{=} \mathcal{O}(k, x_b)$.

privacy restricts the access that adversaries have to the left-or-right key-generation oracle. Specifically, this notion asks that adversaries choose in advance their set of key-generation queries. We note that adversaries are still given oracle access to the left-or-right encryption oracle, with which they can interact in an adaptive manner for any polynomial number of queries. In addition, we note that our definition of a valid function-privacy adversary (Definition 3.1) naturally extends to the selective setting.

Definition 3.3 (Selective-message function privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *T-selective-message function private*, where $T = T(\lambda)$, if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{smFP}}_{\Pi,\mathcal{A},T}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{smFP}}_{\Pi,\mathcal{A},T}(\lambda,0)=1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{smFP}}_{\Pi,\mathcal{A},T}(\lambda,1)=1\right]\right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda,b)$ is defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $((\mathsf{m}_{0,1},\ldots,\mathsf{m}_{0,T}),(\mathsf{m}_{1,1},\ldots,\mathsf{m}_{1,T}),\mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$, where $\mathsf{m}_{0,i},\mathsf{m}_{1,i} \in \mathcal{M}_\lambda$ for all $i \in [T]$.
3. $\mathsf{c}^*_i \leftarrow \mathsf{Enc}(\mathsf{msk},\mathsf{m}_{b,i})$ for all $i \in [T]$.
4. $b' \leftarrow \mathcal{A}^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot)}(\mathsf{c}^*_1,\ldots,\mathsf{c}^*_T,\mathsf{state})$, where for each of $\mathcal{A}$'s queries (f_0,f_1) to $\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot)$ it holds that $f_0(\mathsf{m}_{0,i}) = f_1(\mathsf{m}_{1,i})$ for all $i \in [T]$.
5. Output b'.

Such a scheme Π is *selective-message function private* if it is T-selective-message function private for all polynomials $T = T(\lambda)$.

Definition 3.4 (Selective-function function privacy)**.** A private-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is *T-selective-function function private*, where $T = T(\lambda)$, if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\nu(\lambda)$ such that

$$\mathsf{Adv}^{\mathsf{sfFP}}_{\Pi,\mathcal{A},T}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Expt}^{\mathsf{sfFP}}_{\Pi,\mathcal{A},T}(\lambda,0)=1\right] - \Pr\left[\mathsf{Expt}^{\mathsf{sfFP}}_{\Pi,\mathcal{A},T}(\lambda,1)=1\right]\right| \leq \nu(\lambda),$$

where for each $b \in \{0,1\}$ and $\lambda \in \mathbb{N}$ the experiment $\mathsf{Expt}^{\mathsf{sMP}}_{\Pi,\mathcal{A},T}(\lambda,b)$ is defined as follows:

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$.
2. $((f_{0,1},\ldots,f_{0,T}),(f_{1,1},\ldots,f_{1,T}),\mathsf{state}) \leftarrow \mathcal{A}(1^\lambda)$, where $f_{0,i},f_{1,i} \in \mathcal{F}_\lambda$ for all $i \in [T]$.
3. $\mathsf{sk}^*_i \leftarrow \mathsf{KG}(\mathsf{msk},f_{b,i})$ for all $i \in [T]$.
4. $b' \leftarrow \mathcal{A}^{\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)}(\mathsf{sk}^*_1,\ldots,\mathsf{sk}^*_T,\mathsf{state})$, where for each of $\mathcal{A}$'s queries $(\mathsf{m}_0,\mathsf{m}_1)$ to $\mathsf{Enc}_b(\mathsf{msk},\cdot,\cdot)$ it holds that $f_{0,i}(\mathsf{m}_0) = f_{1,i}(\mathsf{m}_1)$ for all $i \in [T]$.
5. Output b'.

Such a scheme Π is *selective-function function private* if it is T-selective-function function private for all polynomials $T = T(\lambda)$.

Finally, we observe that due to the symmetry between the roles of the encryption oracle and the key-generation oracle in these two selective notions, they are in fact equivalent when switching between the encryption algorithm and key-generation algorithm of any given scheme. That is, a private-key functional encryption scheme $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is selective-message function private

if and only if the scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KG}, \mathsf{Dec})$ is selective-function function private. To be a little more accurate, replacing the roles of functions f and message m may require some standard "type casting" to represent a message as function and function as message. This is done using universal machines: To cast a function f as a message, we consider its description as the message to be encrypted. This means that if Enc only takes bounded length messages, then the new scheme will only support functions with bounded description lengths. To cast a message m as a function, we consider a universal function that accepts a description of a function f and outputs $f(\mathsf{m})$. Again, depending on the computational model under consideration, this may impose some restrictions. For example, if working over circuits then the universal circuit imposes an upper bound on the size of the functions supported by the new scheme, whereas that may not have been required in the original scheme before the switch (however, in this example, if the function size was a-priori unbounded in the original scheme, then the message space after the switch will become a-priori length unbounded).

4 Our Function-Private Scheme

In this section we present our generic construction of a function-private private-key functional encryption scheme. Our construction relies on the following two building blocks:

- A private-key functional encryption scheme $\mathcal{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KG}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$.
- A private-key encryption scheme $\mathcal{SKE} = (\mathsf{SKE.KG}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.[5]

Our new functional encryption scheme $\mathcal{FPE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows.

The setup algorithm. On input the security parameter 1^λ the setup algorithm Setup samples $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$, $\mathsf{SKE.k} \leftarrow \mathsf{SKE.KG}(1^\lambda)$, and $\mathsf{SKE.k}' \leftarrow \mathsf{SKE.KG}(1^\lambda)$. Then, it outputs $\mathsf{msk} = (\mathsf{FE.msk}, \mathsf{SKE.k}, \mathsf{SKE.k}')$.

The key-generation algorithm. On input the master secret key $\mathsf{msk} = (\mathsf{FE.msk}, \mathsf{SKE.k}, \mathsf{SKE.k}')$ and a function f, the key-generation algorithm KG first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f)$. Then, it computes $\mathsf{FE.SK}_{U_{c,c'}} \leftarrow \mathsf{FE.KG}(\mathsf{FE.msk}, U_{c,c'})$, where the function $U_{c,c'}$ is described in Figure 1. Finally, it outputs $\mathsf{sk}_f = \mathsf{FE.SK}_{U_{c,c'}}$.

The encryption algorithm. On input the master secret key $\mathsf{msk} = (\mathsf{FE.msk}, \mathsf{SKE.k}, \mathsf{SKE.k}')$ and a message m, the encryption algorithm Enc outputs $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, (\mathsf{m}, \perp, \mathsf{SKE.k}, \perp))$.

The decryption algorithm. On input a functional key sk_f and a ciphertext c, the decryption algorithm Dec outputs $\mathsf{FE.Dec}(\mathsf{sk}_f, \mathsf{c})$.

$U_{c,c'}(\mathsf{m}, \mathsf{m}', k, k')$

1. If $k \neq \perp$, compute $f \leftarrow \mathsf{SKE.Dec}(k, c)$ and output $f(\mathsf{m})$.
2. Else, if $k' \neq \perp$, compute $f' \leftarrow \mathsf{SKE.Dec}(k', c')$ and output $f'(\mathsf{m}')$.
3. Else, output $\perp$.

Figure 1: The function $U_{c,c'}$.

Note that if the underlying scheme $\mathcal{FE}$ supports functions that are computable by circuits of size at most s, for some sufficiently large polynomial $s = s(n)$, then our new scheme $\mathcal{FPE}$ supports functions that are computable by circuits of size $\Omega(s)$. Specifically, a functional key sk_f for a function

[5] To be absolutely formal, this building block is implied by the former in an obvious way.

f in the new scheme consists of a functional key for the function $U_{c,c'}$ in the underlying scheme. The function $U_{c,c'}$ is computable in a straightforward manner by a circuit that contains two copies of a circuit for computing f, and two copies of $\mathcal{SKE}$'s decryption circuit. The security of our construction is captured by the following theorem:

Theorem 4.1. *Assuming that the scheme $\mathcal{SKE}$ is CPA-secure the following hold:*

1. *If the scheme $\mathcal{FE}$ is fully message private then the scheme $\mathcal{FPE}$ is fully function private.*
2. *If the scheme $\mathcal{FE}$ is selective-message message private (resp. T-selective-message message private) then the scheme $\mathcal{FPE}$ is selective-message function private (resp. T-selective-message function private).*
3. *If the scheme $\mathcal{FE}$ is selective-function message private (resp. T-selective-function message private) then the scheme $\mathcal{FPE}$ is selective-function function private (resp. T-selective-function function private).*

As discussed in Section 2.2.1, Theorem 4.1 can be instantiated based on a variety of known functional encryption schemes (e.g., [GVW12, ABG+13, GGH+13, GKP+13a, BCP14]) that offer full message privacy, selective-message message privacy, and selective-function message privacy.

Proof of Theorem 4.1. For concreteness we first prove the theorem for the case where the scheme $\mathcal{FE}$ is fully message private, and then explain the minor adjustments that are needed for the case where the scheme $\mathcal{FE}$ is selectively message private. Let $\mathcal{A}$ be a valid function-privacy adversary for the scheme $\mathcal{FPE}$ (recall Definition 3.1). We prove that there exist a probabilistic-polynomial time adversary $\mathcal{B}_1$ attacking the CPA security of $\mathcal{SKE}$, and a probabilistic polynomial-time adversary $\mathcal{B}_2$ attacking the message privacy of $\mathcal{FE}$, such that

$$\mathsf{Adv}^{\mathsf{FP}}_{\mathcal{FPE},\mathcal{A}}(\lambda) \leq 2 \cdot \left(\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda)\right).$$

We present a sequence of five hybrid experiments, denoted $\mathcal{H}^{(0)},\ldots,\mathcal{H}^{(4)}$, and prove that each two consecutive experiments are computationally indistinguishable from $\mathcal{A}$'s point of view. Each such experiment $\mathcal{H}^{(i)}$ is completely characterized by its key-generation oracle (denoted $\mathsf{KG}^{(i)}$) and its encryption oracle (denoted $\mathsf{Enc}^{(i)}$). The differences between these experiments are summarized in Table 1.

Experiment	Encryption oracle	Key-generation oracle
$\mathcal{H}^{(0)}$	$\mathsf{FE.Enc}(\mathsf{FE.msk},(\mathsf{m}_0,\bot,\mathsf{SKE.k},\bot))$	$c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k},f_0)$, $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}',f_0)$
$\mathcal{H}^{(1)}$	$\mathsf{FE.Enc}(\mathsf{FE.msk},(\mathsf{m}_0,\bot,\mathsf{SKE.k},\bot))$	$c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k},f_0)$, $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}',f_1)$
$\mathcal{H}^{(2)}$	$\mathsf{FE.Enc}(\mathsf{FE.msk},(\bot,\mathsf{m}_1,\bot,\mathsf{SKE.k}'))$	$c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k},f_0)$, $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}',f_1)$
$\mathcal{H}^{(3)}$	$\mathsf{FE.Enc}(\mathsf{FE.msk},(\bot,\mathsf{m}_1,\bot,\mathsf{SKE.k}'))$	$c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k},f_1)$, $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}',f_1)$
$\mathcal{H}^{(4)}$	$\mathsf{FE.Enc}(\mathsf{FE.msk},(\mathsf{m}_1,\bot,\mathsf{SKE.k},\bot))$	$c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k},f_1)$, $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}',f_1)$

Table 1: The differences between the experiments $\mathcal{H}^{(0)},\ldots,\mathcal{H}^{(4)}$. Adjacent experiments that differ on the generation of c or c' are proven indistinguishable using the CPA security of $\mathcal{SKE}$. Adjacent experiments that differ on the input to $\mathsf{FE.Enc}$ are proven indistinguishable using the message privacy of $\mathcal{FE}$.

Experiment $\mathcal{H}^{(0)}(\lambda)$. This experiment is the experiment $\mathcal{A}^{\mathsf{KG}_0(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_0(\mathsf{msk},\cdot,\cdot)}(\lambda)$ where $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$ (see Definition 3.2). That is, we let $\mathsf{KG}^{(0)} = \mathsf{KG}_0$ and $\mathsf{Enc}^{(0)} = \mathsf{Enc}_0$.

Experiment $\mathcal{H}^{(1)}(\lambda)$. This experiment is obtained from the experiment $\mathcal{H}^{(0)}(\lambda)$ by considering a modified key-generation oracle $\mathsf{KG}^{(1)}(\mathsf{msk},\cdot,\cdot)$ that is defined as follows: On input (f_0, f_1) it first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_0)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', \boxed{f_1})$. Then, it computes $\mathsf{FE.SK}_{U_{c,c'}} \leftarrow \mathsf{FE.KG}(\mathsf{FE.msk}, U_{c,c'})$, where the function $U_{c,c'}$ is described in Figure 1. Finally, it outputs $\mathsf{sk}_{f_0} = \mathsf{FE.SK}_{U_{c,c'}}$. The encryption oracle is not modified (i.e., $\mathsf{Enc}^{(1)} = \mathsf{Enc}^{(0)} = \mathsf{Enc}_0$).

Claim 4.2. *There exists a probabilistic polynomial-time adversary $\mathcal{B}_1$ such that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(0)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(0)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(1)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(1)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda).$$

The above claim states that the CPA security of the scheme $\mathcal{SKE}$ implies that the experiments $\mathcal{H}^{(0)}$ and $\mathcal{H}^{(1)}$ are computationally indistinguishable. Specifically, the difference between these experiments is that in $\mathcal{H}^{(0)}$ the key-generation oracle computes $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', \boxed{f_0})$ whereas in $\mathcal{H}^{(1)}$ it computes $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', \boxed{f_1})$. The claim then follows from the fact that the key $\mathsf{SKE.k}'$ is not being used for any other purpose in these experiments. We the refer the reader to Appendix A for the complete proof of Claim 4.2.

Experiment $\mathcal{H}^{(2)}(\lambda)$. This experiment is obtained from the experiment $\mathcal{H}^{(1)}(\lambda)$ by considering a modified encryption oracle $\mathsf{Enc}^{(2)}(\mathsf{msk},\cdot,\cdot)$ that is defined as follows: On input $(\mathsf{m}_0, \mathsf{m}_1)$ it outputs $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}')})$. The key-generation oracle is not modified (i.e., $\mathsf{KG}^{(2)} = \mathsf{KG}^{(1)}$).

Claim 4.3. *There exists a probabilistic polynomial-time adversary $\mathcal{B}_2$ such that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(1)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(1)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(2)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(2)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda).$$

The above claim states that the message privacy of the scheme $\mathcal{FE}$ implies that the experiments $\mathcal{H}^{(1)}$ and $\mathcal{H}^{(2)}$ are computationally indistinguishable. Specifically, the difference between these experiments is that in $\mathcal{H}^{(1)}$ the encryption oracle computes $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\mathsf{m}_0, \bot, \mathsf{SKE.k}, \bot)})$ whereas in $\mathcal{H}^{(2)}$ it computes $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}')})$. Note that for each functional key $\mathsf{FE.SK}_{U_{c,c'}}$ that is produced by the key-generation algorithm in these experiments it holds that $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_0)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$. Therefore, by the fact that $\mathcal{A}$ is a valid function-privacy adversary we know that $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$ and therefore

$$U_{c,c'}(\mathsf{m}_0, \bot, \mathsf{SKE.k}, \bot) = f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1) = U_{c,c'}(\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}').$$

We the refer the reader to Appendix A for the complete proof of Claim 4.3.

Experiment $\mathcal{H}^{(3)}(\lambda)$. This experiment is obtained from the experiment $\mathcal{H}^{(2)}(\lambda)$ by considering a modified key-generation oracle $\mathsf{KG}^{(3)}(\mathsf{msk},\cdot,\cdot)$ that is defined as follows: On input (f_0, f_1) it first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, \boxed{f_1})$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$. Then, it computes $\mathsf{FE.SK}_{U_{c,c'}} \leftarrow \mathsf{FE.KG}(\mathsf{FE.msk}, U_{c,c'})$, where the function $U_{c,c'}$ is described in Figure 1. Finally, it outputs $\mathsf{sk}_{f_1} = \mathsf{FE.SK}_{U_{c,c'}}$. The encryption oracle is not modified (i.e., $\mathsf{Enc}^{(3)} = \mathsf{Enc}^{(2)}$).

Claim 4.4. *There exists a probabilistic polynomial-time adversary $\mathcal{B}_1$ such that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(2)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(2)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(3)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(3)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda).$$

The proof of the above claim is essentially identical to the proof of Claim 4.2. Specifically, the difference between experiments $\mathcal{H}^{(2)}$ and $\mathcal{H}^{(3)}$ is that in $\mathcal{H}^{(2)}$ the key-generation oracle computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, \boxed{f_0})$ whereas in $\mathcal{H}^{(3)}$ it computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, \boxed{f_1})$. The claim then follows from the fact that the key $\mathsf{SKE.k}$ is not being used for any other purpose in these experiments. We the refer the reader to Appendix A for the complete proof of Claim 4.4.

Experiment $\mathcal{H}^{(4)}(\lambda)$. This experiment is obtained from the experiment $\mathcal{H}^{(3)}(\lambda)$ by considering a modified encryption oracle $\mathsf{Enc}^{(4)}(\mathsf{msk}, \cdot, \cdot)$ that is defined as follows: On input $(\mathsf{m}_0, \mathsf{m}_1)$ it outputs $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\mathsf{m}_1, \bot, \mathsf{SKE.k}, \bot)})$. The key-generation oracle is not modified (i.e., $\mathsf{KG}^{(4)} = \mathsf{KG}^{(3)}$). Note that $\mathsf{Enc}^{(4)} = \mathsf{Enc}_1$ and $\mathsf{KG}^{(4)} = \mathsf{KG}_1$, and therefore this experiment is in fact the experiment $\mathcal{A}^{\mathsf{KG}_1(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_1(\mathsf{msk},\cdot,\cdot)}(\lambda)$.

Claim 4.5. *There exists a probabilistic polynomial-time adversary $\mathcal{B}_2$ such that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(3)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(3)}(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(4)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(4)}(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda).$$

The proof of the above claim is essentially identical to the proof of Claim 4.3 (it is in fact somewhat simpler). Specifically, the difference between experiments $\mathcal{H}^{(3)}$ and $\mathcal{H}^{(4)}$ is that in $\mathcal{H}^{(3)}$ the encryption oracle computes $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}')})$ whereas in $\mathcal{H}^{(4)}$ it computes $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, \boxed{(\mathsf{m}_1, \bot, \mathsf{SKE.k}, \bot)})$. Note that for each functional key $\mathsf{FE.SK}_{U_{c,c'}}$ that is produced by the key-generation algorithm in these experiments it holds that $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_1)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$, and therefore

$$U_{c,c'}(\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}') = f_1(\mathsf{m}_1) = U_{c,c'}(\mathsf{m}_1, \bot, \mathsf{SKE.k}, \bot).$$

We refer the reader to Appendix A for the complete proof of Claim 4.5.

We conclude the proof by observing that Claims 4.2–4.5 imply that there exist polynomial-time adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{FP}}_{\Pi,\mathcal{A}}(\lambda) &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{A}^{\mathsf{KG}_0(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_0(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}_1(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_1(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right]\right| \\ &\leq 2 \cdot \left(\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda)\right).\end{aligned}$$

This settles the proof of the theorem for the case where the scheme $\mathcal{FE}$ is message private, showing that the scheme $\mathcal{FPE}$ is function private. Finally, in the case where the scheme $\mathcal{FE}$ is selectively-message private, an almost identical proof shows that the scheme $\mathcal{FPE}$ offers the same flavor of selective function private. The only difference is that the modifications to the left-or-right encryption and key-generation oracles in our hybrids should be applied at the beginning of the experiments (depending on the flavor of the selective notion). ■

Acknowledgments

We thank Shweta Agrawal and Vinod Vaikuntanthan for insightful discussions.

References

[AAB+13] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai. Function private functional encryption and property preserving encryption: New definitions and positive results. Cryptology ePrint Archive, Report 2013/744, 2013.

[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+14] P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. The trojan method in functional encryption: From selective to adaptive security, generically. Cryptology ePrint Archive, Report 2014/917, 2014.

[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.

[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.

[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.

[BRS13a] D. Boneh, A. Raghunathan, and G. Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *Advances in Cryptology – CRYPTO '13*, pages 461–478, 2013.

[BRS13b] D. Boneh, A. Raghunathan, and G. Segev. Function-private subspace-membership encryption and its applications. In *Advances in Cryptology – ASIACRYPT '13*, pages 255–275, 2013.

[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.

[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.

[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.

[GGG+14] S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *Advances in Cryptology – EUROCRYPT '14*, pages 578–602, 2014. Merge of [GGJ+13] and [GKL+13].

[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+14] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Fully secure functional encryption without obfuscation. Cryptology ePrint Archive, Report 2014/666, 2014.

[GGJ+13] S. Goldwasser, V. Goyal, A. Jain, and A. Sahai. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/727, 2013.

[GJK+13] V. Goyal, A. Jain, V. Koppula, and A. Sahai. Functional encryption for randomized functionalities. Cryptology ePrint Archive, Report 2013/729, 2013.

[GKL+13] S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013.

[GKP+13a] 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.

[GKP+13b] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology – CRYPTO '13*, pages 536–553, 2013.

[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.

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

[NY90] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, 1990.

[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.

[SSW09] E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Proceedings of the 6th Theory of Cryptography Conference*, pages 457–473, 2009.

[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.

[Wat14] B. Waters. A punctured programming approach to adaptively secure functional encryption. Cryptology ePrint Archive, Report 2014/588, 2014.

A Proofs of Claims 4.2–4.5

In this section we provide the complete proofs of Claims 4.2–4.5 from Section 4.

Proof of Claim 4.2. Consider the following adversary $\mathcal{B}_1$ that is given as input the security parameter 1^λ, and has access to the oracle $\mathsf{SKE.Enc}_b(\mathsf{SKE.k}', \cdot, \cdot)$, where $b \in \{0,1\}$ and $\mathsf{SKE.k}' \leftarrow \mathsf{SKE.KG}(1^\lambda)$. The adversary $\mathcal{B}_1$ first samples $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{SKE.k} \leftarrow \mathsf{SKE.KG}(1^\lambda)$, and then invokes the adversary $\mathcal{A}$ on the security parameter 1^λ. Next, $\mathcal{B}_1$ simulates to $\mathcal{A}$ the left-or-right key-generation and encryption oracles of the scheme $\mathcal{FPE}$ as follows:

- **Simulating the encryption oracle.** On input $(\mathsf{m}_0, \mathsf{m}_1)$, the adversary $\mathcal{B}_1$ outputs $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, (\mathsf{m}_0, \bot, \mathsf{SKE.k}, \bot))$. Note that $\mathcal{B}_1$ knows the keys $\mathsf{FE.msk}$ and $\mathsf{SKE.k}$.
- **Simulating the key-generation oracle.** On input (f_0, f_1), the adversary $\mathcal{B}_1$ first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f)$ (note that $\mathcal{B}_1$ knows the key $\mathsf{SKE.k}$). Then, it queries its own encryption oracle $\mathsf{SKE.Enc}_b(\mathsf{SKE.k}', \cdot, \cdot)$ with (f_0, f_1) for obtaining a ciphertext c'. Next, $\mathcal{B}_1$ computes and outputs $\mathsf{FE.SK}_{U_{c,c'}} \leftarrow \mathsf{FE.KG}(\mathsf{FE.msk}, U_{c,c'})$ (note that $\mathcal{B}_1$ knows the key $\mathsf{FE.msk}$).

Finally, $\mathcal{B}_1$ outputs the output of $\mathcal{A}$. We observe that if $b = 0$ then $\mathcal{B}_1$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(0)}$ and $\mathsf{KG}^{(0)}$, and if $b = 1$ then $\mathcal{B}_1$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(1)}$ and $\mathsf{KG}^{(1)}$. Therefore,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda) &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{B}_1^{\mathsf{SKE.Enc}_0(\mathsf{SKE.k}',\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{B}_1^{\mathsf{SKE.Enc}_1(\mathsf{SKE.k}',\cdot,\cdot)}(\lambda)=1\right]\right| \\ &= \left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(0)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(0)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(1)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(1)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right]\right|.\end{aligned}$$

■

Proof of Claim 4.3. Consider the following adversary $\mathcal{B}_2$ that is given as input the security parameter 1^λ, and has access to the oracles $\mathsf{FE.Enc}_b(\mathsf{FE.msk}, \cdot, \cdot)$ and $\mathsf{FE.KG}(\mathsf{FE.msk}, \cdot)$, where $b \in \{0,1\}$ and $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$. The adversary $\mathcal{B}_2$ first samples $\mathsf{SKE.k} \leftarrow \mathsf{SKE.KG}(1^\lambda)$ and $\mathsf{SKE.k}' \leftarrow \mathsf{SKE.KG}(1^\lambda)$, and then invokes the adversary $\mathcal{A}$ on the security parameter 1^λ. Next, $\mathcal{B}_2$ simulates to $\mathcal{A}$ the left-or-right key-generation and encryption oracles of the scheme $\mathcal{FPE}$ as follows:

- **Simulating the encryption oracle.** On input $(\mathsf{m}_0, \mathsf{m}_1)$, the adversary $\mathcal{B}_2$ queries its own encryption oracle $\mathsf{FE.Enc}_b(\mathsf{FE.msk}, \cdot, \cdot)$ with $((\mathsf{m}_0, \bot, \mathsf{SKE.k}, \bot), (\bot, \mathsf{m}_1, \bot, \mathsf{SKE.k}'))$ for obtaining a ciphertext c. It then outputs c.
- **Simulating the key-generation oracle.** On input (f_0, f_1), the adversary $\mathcal{B}_2$ first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_0)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$. Next, it queries its own key-generation oracle $\mathsf{FE.KG}(\mathsf{FE.msk}, \cdot)$ with $U_{c,c'}$ for obtaining a functional key $\mathsf{FE.SK}_{U_{c,c'}}$. It then outputs $\mathsf{FE.SK}_{U_{c,c'}}$.

Finally, $\mathcal{B}_2$ outputs the output of $\mathcal{A}$. We observe that if $b = 0$ then $\mathcal{B}_2$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(1)}$ and $\mathsf{KG}^{(1)}$, and if $b = 1$ then $\mathcal{B}_2$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(2)}$ and $\mathsf{KG}^{(2)}$. Therefore,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda) &\stackrel{\mathsf{def}}{=} \Big|\Pr\left[\mathcal{B}_2^{\mathsf{FE.KG}(\mathsf{FE.msk},\cdot),\mathsf{FE.Enc}_0(\mathsf{FE.msk},\cdot,\cdot)}(\lambda)=1\right] \\ &\qquad - \Pr\left[\mathcal{B}_2^{\mathsf{FE.KG}(\mathsf{FE.msk},\cdot),\mathsf{FE.Enc}_1(\mathsf{FE.msk},\cdot,\cdot)}(\lambda)=1\right]\Big| \\ &= \left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(1)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(1)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(2)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(2)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\right]\right|.\end{aligned}$$

Therefore, it only remains to prove that $\mathcal{B}_2$ is a valid-message privacy adversary (see Definition 2.3). That is, it remains to prove that for all functions $U_{c,c'}$ with which it queries its key-generation oracle, and for all pairs of messages $((\mathsf{m}_0, \perp, \mathsf{SKE.k}, \perp), (\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}'))$ with which it queries its left-or-right encryption oracle it holds that $U_{c,c'}(\mathsf{m}_0, \perp, \mathsf{SKE.k}, \perp) = U_{c,c'}(\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}')$. By the definition of $\mathcal{B}_2$ for each such function $U_{c,c'}$ it holds that $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_0)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$, where (f_0, f_1) is a query made by $\mathcal{A}$ to the left-or-right key-generation oracle of the scheme $\mathcal{FPE}$. The fact that $\mathcal{A}$ is a valid function-privacy adversary (see Definition 3.1) guarantees that $f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1)$, and therefore by the definition of the function $U_{c,c'}$ and the perfect correctness of $\mathcal{SKE}$, it holds that

$$U_{c,c'}(\mathsf{m}_0, \perp, \mathsf{SKE.k}, \perp) = f_0(\mathsf{m}_0) = f_1(\mathsf{m}_1) = U_{c,c'}(\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}').$$

■

Proof of Claim 4.4. Consider the following adversary $\mathcal{B}_1$ that is given as input the security parameter 1^λ, and has access to the oracle $\mathsf{SKE.Enc}_b(\mathsf{SKE.k}, \cdot, \cdot)$, where $b \in \{0,1\}$ and $\mathsf{SKE.k} \leftarrow \mathsf{SKE.KG}(1^\lambda)$. The adversary $\mathcal{B}_1$ first samples $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$ and $\mathsf{SKE.k}' \leftarrow \mathsf{SKE.KG}(1^\lambda)$, and then invokes the adversary $\mathcal{A}$ on the security parameter 1^λ. Next, $\mathcal{B}_1$ simulates to $\mathcal{A}$ the left-or-right key-generation and encryption oracles of the scheme $\mathcal{FPE}$ as follows:

- **Simulating the encryption oracle.** On input $(\mathsf{m}_0, \mathsf{m}_1)$, the adversary $\mathcal{B}_1$ outputs $\mathsf{c} \leftarrow \mathsf{FE.Enc}(\mathsf{FE.msk}, (\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}'))$. Note that $\mathcal{B}_1$ knows the keys $\mathsf{FE.msk}$ and $\mathsf{SKE.k}'$.
- **Simulating the key-generation oracle.** On input (f_0, f_1), the adversary $\mathcal{B}_1$ first queries its own encryption oracle $\mathsf{SKE.Enc}_b(\mathsf{SKE.k}, \cdot, \cdot)$ with (f_0, f_1) for obtaining a ciphertext c. Then, it computes $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$ (note that $\mathcal{B}_1$ knows the key $\mathsf{SKE.k}'$). Next, $\mathcal{B}_1$ computes and outputs $\mathsf{FE.SK}_{U_{c,c'}} \leftarrow \mathsf{FE.KG}(\mathsf{FE.msk}, U_{c,c'})$ (note that $\mathcal{B}_1$ knows the key $\mathsf{FE.msk}$).

Finally, $\mathcal{B}_1$ outputs the output of $\mathcal{A}$. We observe that if $b = 0$ then $\mathcal{B}_1$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(2)}$ and $\mathsf{KG}^{(2)}$, and if $b = 1$ then $\mathcal{B}_1$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(3)}$ and $\mathsf{KG}^{(3)}$. Therefore,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{CPA}}_{\mathcal{SKE},\mathcal{B}_1}(\lambda) &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{B}_1^{\mathsf{SKE.Enc}_0(\mathsf{SKE.k},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{B}_1^{\mathsf{SKE.Enc}_1(\mathsf{SKE.k},\cdot,\cdot)}(\lambda) = 1\right]\right| \\ &= \left|\Pr\left[\mathcal{A}^{\mathsf{KG}^{(2)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(2)}(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{KG}^{(3)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(3)}(\mathsf{msk},\cdot,\cdot)}(\lambda) = 1\right]\right|.\end{aligned}$$

■

Proof of Claim 4.5. Consider the following adversary $\mathcal{B}_2$ that is given as input the security parameter 1^λ, and has access to the oracles $\mathsf{FE.Enc}_b(\mathsf{FE.msk}, \cdot, \cdot)$ and $\mathsf{FE.KG}(\mathsf{FE.msk}, \cdot)$, where $b \in \{0,1\}$ and $\mathsf{FE.msk} \leftarrow \mathsf{FE.Setup}(1^\lambda)$. The adversary $\mathcal{B}_2$ first samples $\mathsf{SKE.k} \leftarrow \mathsf{SKE.KG}(1^\lambda)$ and $\mathsf{SKE.k}' \leftarrow \mathsf{SKE.KG}(1^\lambda)$, and then invokes the adversary $\mathcal{A}$ on the security parameter 1^λ. Next, $\mathcal{B}_2$ simulates to $\mathcal{A}$ the left-or-right key-generation and encryption oracles of the scheme $\mathcal{FPE}$ as follows:

- **Simulating the encryption oracle.** On input $(\mathsf{m}_0, \mathsf{m}_1)$, the adversary $\mathcal{B}_2$ queries its own encryption oracle $\mathsf{FE.Enc}_b(\mathsf{FE.msk}, \cdot, \cdot)$ with $((\perp, \mathsf{m}_1, \perp, \mathsf{SKE.k}'), (\mathsf{m}_1, \perp, \mathsf{SKE.k}, \perp))$ for obtaining a ciphertext c. It then outputs c.
- **Simulating the key-generation oracle.** On input (f_0, f_1), the adversary $\mathcal{B}_2$ first computes $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_1)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$. Next, it queries its own key-generation oracle $\mathsf{FE.KG}(\mathsf{FE.msk}, \cdot)$ with $U_{c,c'}$ for obtaining a functional key $\mathsf{FE.SK}_{U_{c,c'}}$. It then outputs $\mathsf{FE.SK}_{U_{c,c'}}$.

Finally, $\mathcal{B}_2$ outputs the output of $\mathcal{A}$. We observe that if $b = 0$ then $\mathcal{B}_2$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(3)}$ and $\mathsf{KG}^{(3)}$, and if $b = 1$ then $\mathcal{B}_2$ provides a perfect simulation of the oracles $\mathsf{Enc}^{(4)}$ and $\mathsf{KG}^{(4)}$. Therefore,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{MP}}_{\mathcal{FE},\mathcal{B}_2}(\lambda) &\stackrel{\mathsf{def}}{=} \Big|\Pr\Big[\mathcal{B}_2^{\mathsf{FE.KG}(\mathsf{FE.msk},\cdot),\mathsf{FE.Enc}_0(\mathsf{FE.msk},\cdot,\cdot)}(\lambda)=1\Big] \\ &\qquad - \Pr\Big[\mathcal{B}_2^{\mathsf{FE.KG}(\mathsf{FE.msk},\cdot),\mathsf{FE.Enc}_1(\mathsf{FE.msk},\cdot,\cdot)}(\lambda)=1\Big]\Big| \\ &= \Big|\Pr\Big[\mathcal{A}^{\mathsf{KG}^{(3)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(3)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{KG}^{(4)}(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}^{(4)}(\mathsf{msk},\cdot,\cdot)}(\lambda)=1\Big]\Big|.\end{aligned}$$

Therefore, it only remains to prove that $\mathcal{B}_2$ is a valid-message privacy adversary (see Definition 2.3). That is, it remains to prove that for all functions $U_{c,c'}$ with which it queries its key-generation oracle, and for all pairs of messages $((\perp,\mathsf{m}_1,\perp,\mathsf{SKE.k}'),(\mathsf{m}_1,\perp,\mathsf{SKE.k},\perp))$ with which it queries its left-or-right encryption oracle it holds that $U_{c,c'}(\perp,\mathsf{m}_1,\perp,\mathsf{SKE.k}') = U_{c,c'}(\mathsf{m}_1,\perp,\mathsf{SKE.k},\perp)$. By the definition of $\mathcal{B}_2$ for each such function $U_{c,c'}$ it holds that $c \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}, f_1)$ and $c' \leftarrow \mathsf{SKE.Enc}(\mathsf{SKE.k}', f_1)$, where (f_0, f_1) is a query made by $\mathcal{A}$ to the left-or-right key-generation oracle of the scheme $\mathcal{FPE}$. Therefore, by the definition of the function $U_{c,c'}$ it holds that

$$U_{c,c'}(\perp,\mathsf{m}_1,\perp,\mathsf{SKE.k}') = f_1(\mathsf{m}_1) = U_{c,c'}(\mathsf{m}_1,\perp,\mathsf{SKE.k},\perp).$$

■