Multi-input Functional Encryption in the Private-Key Setting: Stronger Security from Weaker Assumptions

Zvika Brakerski[1], Ilan Komargodski[1](✉), and Gil Segev[2]

[1] Weizmann Institute of Science, 76100 Rehovot, Israel
{zvika.brakerski,ilan.komargodski}@weizmann.ac.il
[2] Hebrew University of Jerusalem, 91904 Jerusalem, Israel
segev@cs.huji.ac.il

Abstract. We construct a general-purpose *multi-input* functional encryption scheme in the private-key setting. Namely, we construct a scheme where a functional key corresponding to a function f enables a user holding encryptions of $x_1, \ldots, x_t$ to compute $f(x_1, \ldots, x_t)$ but nothing else. This is achieved starting from any general-purpose private-key *single-input* scheme (without any additional assumptions), and is proven to be *adaptively secure* for any constant number of inputs t. Moreover, it can be extended to a super-constant number of inputs assuming that the underlying single-input scheme is sub-exponentially secure.

Instantiating our construction with existing single-input schemes, we obtain multi-input schemes that are based on a variety of assumptions (such as indistinguishability obfuscation, multilinear maps, learning with errors, and even one-way functions), offering various trade-offs between security and efficiency.

Previous and concurrent constructions of multi-input functional encryption schemes either rely on stronger assumptions and provided weaker security guarantees (Goldwasser et al. [EUROCRYPT '14], and Ananth and Jain [CRYPTO '15]), or relied on multilinear maps and could be proven secure only in an idealized generic model (Boneh et al. [EUROCRYPT '15]). In comparison, we present a general transformation that simultaneously relies on weaker assumptions and guarantees stronger security.

Z. Brakerski—Supported by the Israel Science Foundation (Grant No. 468/14) and by the Alon Young Faculty Fellowship.

I. Komargodski—Research supported in part by a grant from the Israel Science Foundation, the I-CORE Program of the Planning and Budgeting Committee, BSF and the Israeli Ministry of Science and Technology.

G. Segev—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.

M. Fischlin and J.-S. Coron (Eds.): EUROCRYPT 2016, Part II, LNCS 9666, pp. 852–880, 2016.
DOI: 10.1007/978-3-662-49896-5_30

1 Introduction

The emerging vision of functional encryption [14,31,32] extends the traditional "all-or-nothing" view of encryption schemes. Specifically, functional encryption schemes offer additional flexibility by supporting restricted decryption keys. These keys allow users to learn specific functions of the encrypted data, without learning any additional information. Building upon the early examples of functional encryption schemes for restricted function families (such as identity-based encryption [11,20,34]), extensive research is currently devoted to the construction of functional encryption schemes offering a variety of expressive families of functions (see, for example, [2,4,5,9,10,14,16,19,21,22,25,26,30–32,36]).

Until very recently, research on functional encryption has focused on the case of *single-input* functions. In a single-input functional encryption scheme, a functional key sk_f corresponding to a function f enables a user holding an encryption of a value x to compute $f(x)$, while not revealing any additional information on x. In many scenarios, however, dealing only with single-input functions is insufficient, and a more general framework allowing *multi-input* functions is required.

Goldwasser et al. [24] recently introduced the notion of a *multi-input* functional encryption scheme. In such a scheme, a functional key corresponding to a t-input function f enables a user holding encryptions of $x_1, \ldots, x_t$ to compute $f(x_1, \ldots, x_t)$ without learning any additional information on the x_i's. The work of Goldwasser et al. and their new notion are very well-motivated by a wide range of applications based on mining aggregate information from several different data sources. These include, for example, running SQL queries on encrypted databases, computing over encrypted data streams, non-interactive differentially-private data release, and order-revealing encryption (all of which are relevant in both the public-key setting and the private-key one [12]).

Goldwasser et al. presented a rigorous framework for capturing the security of multi-input schemes in the public-key setting and in the private-key one. In addition, relying on indistinguishability obfuscation and one-way functions [8,21,29], they constructed the first multi schemes. In terms of functionality, their schemes are extremely expressive, supporting all multi-input functions that are computable by bounded-size circuits. In terms of security, however, their private-key scheme satisfies a weak selective notion, which does not allow the adversary to access an encryption oracle (which is quite crippling in the private-key setting), and requires an a-priori bound on the number of challenge ciphertexts (the ciphertext length in their scheme depends on the number of challenge ciphertexts).

Following the work of Goldwasser et al. [24], a private-key multi-input functional encryption scheme that satisfies a more standard notion of security (one that allows access to an encryption oracle) was constructed by Boneh et al. [12]. Their scheme is based on multilinear maps, and is proven secure in the idealized generic multilinear map model. In addition, in an independent and concurrent work, Ananth and Jain [5] constructed a *selectively-secure* multi-input private-key functional encryption scheme based on any general-purpose *public-key* functional encryption scheme (as an intermediate step in constructing an indistinguishability obfuscator).

Thus, constructions of multi-input functional encryption schemes in the private-key setting have so far either relied on stronger assumptions and provided weaker security guarantees [5,24][1], or could be proven secure only in an idealized generic model [12].

1.1 Our Contributions

In this paper we present a construction of private-key multi-input functional encryption from *any* general-purpose *private-key single-input* functional encryption scheme (without introducing any additional assumptions). The resulting scheme supports any set of efficiently-computable functions, and provides adaptive security in the standard model for any constant number of inputs. We prove the following theorem:

Theorem 1.1. *Assuming the existence of any private-key single-input selectively-secure functional encryption scheme, for any constant $t \geq 2$ there exists a private-key t-input adaptively-secure functional encryption scheme.*

Moreover, assuming that the underlying private-key single-input scheme is sub-exponentially secure, our resulting scheme provides adaptive security for a *super-constant* number of inputs (we refer the reader to Sect. 1.3 for more details). Following [1,19], our scheme provides not only message privacy, but in fact a unified notion that captures both message privacy and function privacy (this notion is known as *full security* – see Sect. 2.3 for more details).

Instantiations. Instantiating our construction with existing private-key single-input schemes, we obtain new multi-input schemes based on a variety of assumptions in the standard model. Specifically, we obtain schemes that are secure for an unbounded number of encryption and key-generation queries based on indistinguishability obfuscation or multilinear maps. In addition, if the number of encryption and key-generation queries is a-priori bounded, we can rely on much milder assumptions such as learning with errors [25] or even the existence of one-way functions or low-depth pseudorandom generators [26]. See Sect. 2.2 for further discussion.

Comparison with Previous and Concurrent Work. Compared to the previous work of Goldwasser et al. [24] and Boneh et al. [12], our work yields stronger security guarantees and at the same time relies solely on a necessary assumption. Specifically, whereas Goldwasser et al. and Boneh et al. rely on indistinguishability obfuscation and multilinear maps, respectively, we rely on the existence of any general-purpose private-key single-input scheme, which is obviously necessary. Moreover, whereas the scheme of Goldwasser et al. provides a selective notion of security which, in addition, does not allow adversaries to access an

[1] In terms of assumptions, the recent work of Asharov and Segev [7] shows that indistinguishability obfuscation and public-key functional encryption are significantly stronger primitives than private-key functional encryption. We refer the reader to Sect. 1.1 for a more elaborate discussion.

encryption oracle and requires an a-priori bound on the number of challenge ciphertexts, and the scheme of Boneh et al. is proved secure only in an idealized generic model that does not properly capture real-world adversaries, our scheme provides adaptive security in the standard model for any number of challenge ciphertexts.

Compared to the concurrent work of Ananth and Jain [5], our work again yields stronger security guarantees while relying on a weaker assumption. Specifically, whereas the construction of Ananth and Jain relies on *public-key* functional encryption and guarantees *selective* security (where, in addition, the adversary is not allow to access an encryption oracle), our construction relies on *private-key* functional encryption and guarantees *full* security. From the technical point of view, the scheme of Ananth and Jain is essentially "Step 1" of our approach (see Sect. 1.3), which was sufficient (together with additional techniques and assumptions) for constructing their obfuscator. The vast majority of our efforts in this paper are devoted for providing better security while simultaneously relying on weaker assumptions, as mentioned above.

In terms of assumptions, the recent work of Asharov and Segev [7] shows that private-key functional encryption is much weaker than *any* public-key primitive (in particular, it is much weaker than public-key functional encryption). Specifically, they show that using the currently-known techniques it is impossible to use a private-key functional encryption scheme for constructing even a key-agreement protocol (and therefore, in particular, it is impossible to construct a public-key encryption scheme or a public-key functional encryption scheme).

Finally, we note that in addition to introducing the notion of a multi-input functional encryption scheme, Goldwasser et al. [24] introduced the more general notion of a *multi-client* multi-input functional encryption scheme. In such a scheme, each input coordinate is associated with its own encryption key, and security should be satisfied for all coordinates whose encryption keys are not known to the adversary. In this paper we do not consider this more general notion, and an interesting open problem is to extend our approach to the multi-client setting.

1.2 Additional Related Work

Extensive research has been devoted to the study of functional encryption, and for concreteness we focus here only on those previous efforts that are directly relevant to the techniques used in this paper.

Function-Private Functional Encryption. The security guarantees of functional encryption typically focus on *message privacy.* Intuitively, message privacy asks that a functional key sk_f does not help in distinguishing encryptions of two messages, m_0 and m_1, as long as $f(m_0) = f(m_1)$. In various cases, however, it is also useful to consider *function privacy* [1,13,19,35], asking that a functional key sk_f does not reveal any unnecessary information on the function f. Specifically, in the private-key setting, function privacy asks that an encryption of a message m does not help in distinguishing two functional keys, sk_{f_0} and sk_{f_1}, as long as

$f_0(m) = f_1(m)$. Brakerski and Segev [19] recently showed that any private-key functional encryption scheme can be generically transformed into one that satisfies a unified notion of security, referred to as *full security*, which considers both message privacy and function privacy.

Other than being a useful notion for various applications, function privacy was found useful as a building block in the construction of several functional encryption schemes [4,30]. One of the key insights that we utilize in this work is that function-private functional encryption allows to successfully apply proof techniques "borrowed" from the indistinguishability obfuscation literature (including, for example, a variant of the punctured programming approach of Sahai and Waters [33]).

Key-Encapsulation Techniques in Functional Encryption. Key encapsulation (also known as "hybrid encryption") is an extremely useful approach in the design of encryption schemes, both for improved efficiency and for improved security. Specifically, key encapsulation typically means that instead of encrypting a message m under a fixed key sk, one can instead sample a random key k, encrypt m under k and then encrypt k under sk. Recently, Ananth et al. [4] showed that key encapsulation is useful also in the setting of functional encryption. They showed that it can be used to transform any selectively-secure functional encryption scheme into an adaptively-secure one (in both the public-key setting and the private-key one). Their construction and proof technique hint that key encapsulation techniques may in fact be a general tool that is useful in the design of functional encryption schemes. Our constructions incorporate key encapsulation techniques, and exhibit additional strengths of this technique in the context of functional encryption schemes. Specifically, as discussed in Sect. 1.3, we use key encapsulation techniques to create "sufficient independence" between combinations of different ciphertexts, a crucial ingredient in our constructions (see Sect. 1.3 for a detailed comparison between our technique and that of Ananth et al.).

Multi-input Functional Encryption Schemes and Obfuscation. An important aspect in studying multi-input functional encryption schemes is its tight connection to indistinguishability obfuscation. Goldwasser et al. [24] showed that the following three primitives are equivalent: (1) selectively-secure *private*-key multi-input functional encryption scheme with polynomially many inputs, (2) selectively-secure *public*-key two-input functional encryption scheme, and (3) indistinguishability obfuscation. The works of Ananth and Jain [5] and Ananth, Jain and Sahai [6] show how to construct a selectively-secure private-key multi-input functional encryption scheme with polynomially many inputs (and thereby an indistinguishability obfuscator) from any sub-exponentially-secure *public-key* single-input functional encryption scheme.[2]

[2] Bitansky and Vaikuntanathan [10] achieved the same result (an indistinguishability obfuscator) as [5] using a similar construction (at least conceptually) while relying essentially on the same assumptions. However, they construct an indistinguishability obfuscator directly without going through the equivalence to multi-input functional encryption schemes.

1.3 Overview of Our Constructions and Techniques

In this section we provide a high-level overview of our constructions. For concreteness, we focus here mainly on two-input schemes, and then briefly discuss the generalization of our approach to more than two inputs (we refer the reader to Appendix A for the generalization to t-input schemes for $t \geq 2$). In what follows, we start by briefly describing the functionality and security properties of two-input schemes in the private-key setting. Then, we explain the main ideas underlying our constructions. We emphasize that the forthcoming overview is very high-level and ignores many technical details. For the full details we refer to Sects. 3 and 4.

Functionality and Security. In a private-key two-input functional encryption scheme, the master secret key msk of the scheme is used for encrypting any messages x and y (separately) to the first and second coordinates, respectively, and for generating functional keys for two-input functions. A functional key sk_f corresponding to a function f enables to compute $f(x, y)$ given $\mathsf{Enc}(x)$ and $\mathsf{Enc}(y)$. Building upon the previous notions of security for private-key multi-input functional encryption schemes [12,24], we consider a strengthened notion of security that combines both message privacy and function privacy (as in [1,19] for single-input schemes), to which we refer as *full security.*[3] Specifically, we consider *adaptive* adversaries that are given access to "left-or-right" key-generation and encryption oracles. These oracles operate in one out of two modes corresponding to a randomly-chosen bit b. The key-generation oracle receives as input pairs of the form (f_0, f_1) and outputs a functional key for f_b. The encryption oracle receives as input pairs of the form (x_0, x_1) for the first coordinate, or (y_0, y_1) for the second coordinate, and outputs an encryption of x_b or y_b. We require that no efficient adversary can guess the bit b with probability noticeably higher than 1/2, as long as for each such three queries (f_0, f_1), (x_0, x_1) and (y_0, y_1) it holds that $f_0(x_0, y_0) = f_1(x_1, y_1)$.

Intuition: Input Aggregation. Given a two-input function $f(\cdot, \cdot)$, one can view f as a single-input function, f^*, that takes a tuple (x, y), which we denote by $x \| y$ to avoid confusion, and computes $f^*(x \| y) = f(x, y)$. Using a single-input scheme, we can generate a functional key for the function f^*. We thus remain with the problem of *aggregating the input.* That is, we need to be able to encrypt inputs x and y, such that given $\mathsf{Enc}(x)$ and $\mathsf{Enc}(y)$ it is possible to compute $\mathsf{Enc}(x \| y)$. At a very high-level, this is achieved by having the encryption of x be an "aggregator": To encrypt x, we will generate a functional key for the

[3] We consider a unified notion capturing both message privacy and function privacy not only as a useful feature for various applications. In fact, the function privacy of the resulting two-input scheme plays a crucial role when extending our results to more than two inputs.

function $\mathsf{AGG}_x(\cdot)$, that on input y outputs an encryption of $x\|y$.[4] There are many technical difficulties in realizing this intuition, as we explain in the remainder of this section.

Step 1: Functional Keys as Ciphertexts. Given any private-key single-input functional encryption scheme, $\mathsf{1FE}$, the first step in our transformation is to use both its ciphertexts and its functional keys as ciphertexts for a two-input scheme $\mathsf{2FE}$: An encryption of a message x to the first coordinate is a functional key sk_x corresponding to a certain functionality that depends on x, and an encryption of a message y to the second coordinate is simply an encryption of y. Intuitively, the hope is that the function privacy of $\mathsf{1FE}$ will hide x, and that the message privacy of $\mathsf{1FE}$ will hide y. More specifically, a first attempt towards realizing this intuition is as follows:

1. The master secret key consists of two keys, $\mathsf{msk_{in}}$ and $\mathsf{msk_{out}}$, for the single-input scheme $\mathsf{1FE}$. The key $\mathsf{msk_{in}}$ is used for encryption, and the key $\mathsf{msk_{out}}$ is used to decryption.
2. An encryption of a message x to the first coordinate is a functional key $\mathsf{sk}_{x,\mathsf{msk_{out}}}$ that is generated using $\mathsf{msk_{in}}$ and corresponds to the following functionality: Given an input y, it outputs an encryption $\mathsf{Enc}_{\mathsf{msk_{out}}}(x\|y)$ of x concatenated with y under $\mathsf{msk_{out}}$. An encryption of a message y to the second coordinate is simply an encryption $\mathsf{Enc}_{\mathsf{msk_{in}}}(y)$ of y under $\mathsf{msk_{in}}$.
3. A functional key for a two-input function f is a functional key that is generated using $\mathsf{msk_{out}}$ for the function f when viewed as a single-input function.
4. Given a functional key for a function f, and two encryptions $\mathsf{sk}_{x,\mathsf{msk_{out}}}$ and $\mathsf{Enc}_{\mathsf{msk_{in}}}(y)$, we first apply $\mathsf{sk}_{x,\mathsf{msk_{out}}}$ on $\mathsf{Enc}_{\mathsf{msk_{in}}}(y)$ to obtain $\mathsf{Enc}_{\mathsf{msk_{out}}}(x\|y)$, and then apply the functional key for f on $\mathsf{Enc}_{\mathsf{msk_{out}}}(x\|y)$.

It is straightforward to verify that the above scheme indeed provides the required functionality of a two-input scheme. Proving its security, however, does not seem to go through: When "attacking" the key $\mathsf{msk_{out}}$, we clearly cannot embed it in the encryptions $\mathsf{sk}_{x,\mathsf{msk_{out}}}$ generated to the first coordinate. A typical approach for dealing with such a difficulty (e.g., [4,19,30]) is to embed all possibly-needed encryptions under $\mathsf{msk_{out}}$ inside the ciphertexts of the two-input scheme (so that the key $\mathsf{msk_{out}}$ will not be explicitly needed). Note, however, that when an adversary makes T encryption queries there may be roughly T^2 different pairs of the form (x, y), and these T^2 pairs cannot be embedded into T ciphertexts (we note that $T = T(\lambda)$ may be any polynomial and it is not known in advance).

An additional approach is to use a *public-key* functional encryption scheme for the role played by $\mathsf{msk_{out}}$ (i.e., replacing $\mathsf{sk}_{x,\mathsf{msk_{out}}}$ with $\mathsf{sk}_{x,\mathsf{pk_{out}}}$). Although

[4] A somewhat related functionality was recently considered by Iovino and Zebrowski [27] who introduced the notion of *mergeable* functional encryption, where one can publicly transform encryptions, $\mathsf{Enc}(x)$ and $\mathsf{Enc}(y)$, of two values into an encryption $\mathsf{Enc}(x\|y)$ of their concatenation. They show how to construct such a scheme for two inputs building on the *specific* construction of [21] and assuming strong notions of obfuscation. In comparison, our approach applies to many inputs (as discussed below), and is based on minimal assumptions.

this solution allows to prove security, we view it as a "warm-up solution" as we would like to avoid relying on a stronger primitive than necessary. Specifically, we would like to rely on private-key functional encryption and not on public-key function encryption (as recently shown by Asharov and Segev [7], private-key functional encryption is significantly weaker than any public-key primitive).

Step 2: Selective Security via "One-Sided" Key Encapsulation. Our approach for resolving the difficulty described uses key-encapsulation techniques in functional encryption. Our main idea here is that when encrypting a message x, we sample a fresh key $\mathsf{msk}^\star$ for the single-input scheme, and output two components: $\mathsf{Enc}_{\mathsf{msk}_{\mathsf{out}}}(\mathsf{msk}^\star)$ and $\mathsf{sk}_{x,\mathsf{msk}^\star}$. Given an encryption $\mathsf{Enc}_{\mathsf{msk}_{\mathsf{in}}}(y)$ of a message y, the component $\mathsf{sk}_{x,\mathsf{msk}^\star}$ enables to compute $\mathsf{Enc}_{\mathsf{msk}^\star}(x||y)$. In addition, a functional key for a function f is now generated using $\mathsf{msk}_{\mathsf{out}}$ for the following functionality: Given an input $\mathsf{msk}^\star$, it outputs a functional key for f (viewed as a single-input function) using $\mathsf{msk}^\star$. This enables to compute $f(x, y)$ given $\mathsf{Enc}_{\mathsf{msk}^\star}(x||y)$ and provides the required functionality.

This "one-sided" key encapsulation enables us to prove a selectively-secure variant of our notion of security.[5] In this variant we require adversaries to specify their encryption queries in advance, and they are then given adaptive access to the left-or-right key-generation oracle. The main idea underlying the proof of security is that our one-sided key encapsulation approach yields sufficient independence and allows attacking the x's one by one, by attacking their corresponding encapsulated keys. Focusing on one message x and its encapsulated key msk^*, an adversary that make T encryption queries $y_1, \ldots, y_T$ to the second coordinate induces only T pairs $\{(x, y_i)\}_{i\in[T]}$ (instead of T^2 pairs as above). Moreover, given that the encryption queries are chosen in advance, we can embed an encryption of $x||y_i$ under $\mathsf{msk}^\star$ inside the encryption of each y_i. This way the key $\mathsf{msk}^\star$ is not explicitly needed, and thus can be attacked (while not affecting any of the other x's).

As discussed in Sect. 1.2, key-encapsulation techniques have been introduced into the setting of functional encryption by Ananth et al. [4]. Our approach builds upon and significantly extends their initial observations, and enables us to create "sufficient independence" between combinations of different ciphertexts, a crucial ingredient in our constructions.

This enables us to construct a selectively-secure two-input scheme from any selectively-secure single-input one (we refer the reader to Sect. 3 for the scheme and its proof of security). Note, however, that this approach is limited to selective adversaries: embedding an encryption of $x||y_i$ inside the encryption of y_i requires knowing x before the adversary queries for the encryption of y_i.

Step 3: Adaptive Security via "Two-Sided" Key Encapsulation. Next, we present a general transformation from selective security to adaptive security (in fact, to our stronger notion of full security). Specifically, we rely on two building blocks: (1) any private-key *selectively-secure two-input* scheme, and (2) any

[5] "One-sided" here refers to the fact that the encapsulated key $\mathsf{msk}^\star$ is generated only from the side of the x's.

private-key *adaptively-secure single-input* scheme (recall that in the single-input setting, selective security implies adaptive security [4]). For this transformation we introduce a new technique which we call “two-sided” key encapsulation, where each pair of messages x and y has its own encapsulated key $\mathsf{msk}^\star$. This, more subtle approach, enables us to “attack” a specific pair of messages each time, since each such pair uses a different encapsulated key: If x is known before y then we embed $x||y$ inside the encryption of y, and if x is known after y then we embed $x||y$ inside the encryption of x. This leaves the problem of how to realize this idea of two-sided key encapsulation. Our two-sided key encapsulation works as follows.

1. The master secret key consists of two keys: A master secret key $\mathsf{msk}_{\mathsf{out}}$ for a selectively-secure two-input scheme, and a master secret key $\mathsf{msk}_{\mathsf{in}}$ for an adaptively-secure single-input scheme.
2. An encryption of a message y consists of two components: $\mathsf{Enc}_{\mathsf{msk}_{\mathsf{out}}}(t)$ and $\mathsf{Enc}_{\mathsf{msk}_{\mathsf{in}}}(y, t)$, where t is a fresh random tag.
3. An encryption of a message x consists of two components: $\mathsf{Enc}_{\mathsf{msk}_{\mathsf{out}}}(s)$ and $\mathsf{sk}_{x,s}$, where s is a fresh random tag. The functional key $\mathsf{sk}_{x,s}$ is generated using $\mathsf{msk}_{\mathsf{in}}$ and corresponds to the following functionality: Given an input (y, t), derive $\mathsf{msk}^\star = \mathsf{PRF}(s, t)$,[6] and output $\mathsf{Enc}_{\mathsf{msk}^\star}(x||y)$.
4. A functional key for a function f is generated using $\mathsf{msk}_{\mathsf{out}}$ for the following functionality: Given *two inputs*, s and t, derive $\mathsf{msk}^\star = \mathsf{PRF}(s, t)$, and output a functional key for f (viewed as a single-input function) using $\mathsf{msk}^\star$.

The crucial observation is that the master secret key $\mathsf{msk}_{\mathsf{out}}$ of the two-input selectively-secure scheme is used for encrypting random tags, whereas the plaintext itself is always encrypted using the master secret key $\mathsf{msk}_{\mathsf{in}}$ of the adaptively-secure single-input scheme. This enables us to prove the full security of the resulting scheme (we refer the reader to Sect. 4 for the scheme and its proof of security).

Comparison to the Selective-to-Adaptive Transformation of Ananth et al. [4]. Our two-sided key encapsulation technique shows that the usability of key-encapsulation in the context of functional encryption, demonstrated by Ananth et al. [4], can be significantly extended. Whereas their generic transformation from selective security to adaptive security for single-input scheme uses a rather direct form of key encapsulation, our approach requires a significantly more structured one in which the encapsulated key is not determined at the time of encryption, but rather generated “freshly” (in a pseudorandom manner) for any two messages x and y as above.

Specifically, Ananth et al. encrypted a message m under a selectively-secure key msk, by sampling a fresh master secret key $\mathsf{msk}^\star$ for a “one-time” adaptively-secure scheme, encrypted m under $\mathsf{msk}^\star$ and then encrypted $\mathsf{msk}^\star$ under msk. This direct encapsulation does not seem to extend to the two-input setting,

[6] More accurately, the key $\mathsf{msk}^\star$ is computed by applying the setup algorithm of $\mathsf{1FE}$ with randomness $\mathsf{PRF}(s, t)$.

as applying it independently in each coordinate seems to hurt both the security and the functionality of the scheme. By introducing our two-sided key-encapsulation idea we are able to balance between the need for using key encapsulation in each coordinate and the need for generating sufficient independence between different pairs of messages.

Step 4: Generalization to t-input Schemes. The generalization of our result to t-input schemes, for $t \geq 2$, consists of two components. The first component is a construction that uses any $(t-1)$-input scheme for building a selectively-secure t-input scheme, for any $t \geq 2$. The second component is a construction that uses any selectively-secure t-input scheme and a fully-secure $(t-1)$-input scheme for building a fully-secure t-input scheme. Thus, for obtaining a fully-secure t-input scheme from any single-input scheme, one can iteratively apply both components alternately t times. This is illustrated in Fig. 1 for the case $t = 3$ (and the same illustration generalizes to any $t > 3$ in a straightforward manner).

This iterative application of our components places a restriction on the number of supported inputs. In general, each such application may result in a polynomial blow-up in the parameters of the scheme. Therefore, $t-1$ applications may result in a blow-up of $\lambda^{2^{O(t)}}$ which must be kept polynomial. Without any additional assumptions, this implies that t can be any fixed constant. Assuming, in addition, that the underlying single-input scheme is sub-exponentially secure, the number of inputs can be made super-constant. Specifically, for any constant $0 < \epsilon < 1$, when instantiating the underlying single-input scheme with security parameter $\tilde{\lambda} = 2^{(\log \lambda)^{\epsilon}}$, the first component can be iteratively applied to reach $t = \Theta(\log \log \lambda)$ inputs. Obtaining a generic transformation that supports a super-constant number of inputs without assuming sub-exponential security (or an alternative form of "succinctness") is left as an open problem.

1.4 Paper Organization

The remainder of this paper is organized as follows. In Sect. 2 we provide an overview of the notation, definitions, and tools underlying our constructions. In Sect. 3 we present a construction of a selectively-secure two-input functional encryption scheme from any single-input scheme. In Sect. 4 we present a construction of a fully-secure two-input functional encryption scheme from any selectively-secure one. In Appendix A we generalize our approach to t-input schemes for $t \geq 2$. In the full version [18] we provide the formal proofs of our theorems from Sects. 3 and 4, and from Appendix A.

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 $\mathcal{X}$.

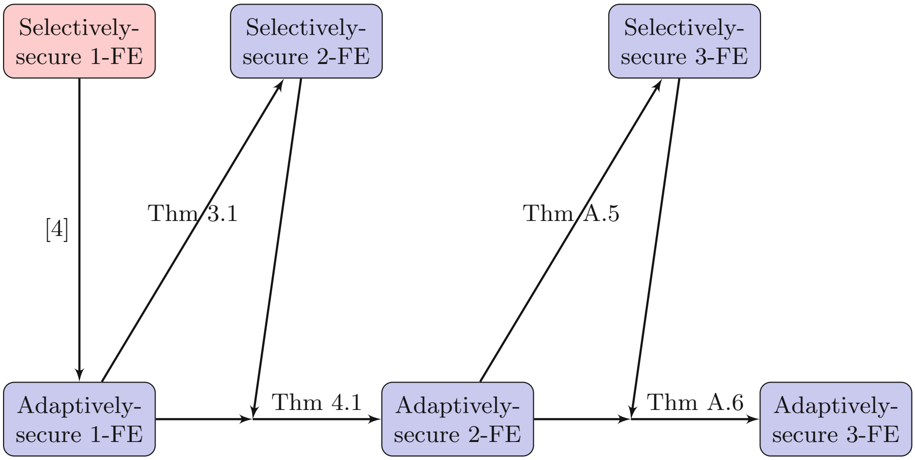

Fig. 1. An illustration of the required iterative applications of our two transformations for obtaining an adaptively-secure three-input scheme based on any selectively-secure single-input scheme.

For a randomized function f and an input $x \in \mathcal{X}$, we denote by $y \leftarrow f(x)$ the process of sampling a value y from the distribution $f(x)$. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. 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 $\lambda \in \mathbb{N}$. Throughout the paper, we denote by λ the security parameter.

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 $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ 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 easy of notation we may denote by $\mathsf{PRF.Eval}_K(\cdot)$ or $\mathsf{PRF}_K(\cdot)$ the function $\mathsf{PRF.Eval}(K, \cdot)$ for $K \in \mathcal{K}_\lambda$. The following is the standard definition of a pseudorandom function family.

Definition 2.1 (Pseudorandomness). *A function family* $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ *is* pseudorandom *if for every probabilistic polynomial-time algorithm*

$\mathcal{A}$ there exits a negligible function $\mathsf{neg}(\cdot)$ *such that*

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

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 [15,17,28,33]. 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* $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$ *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*

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

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

For our constructions we rely on pseudorandom functions that need to be punctured only at one 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 [15,17,28,33] the GGM construction [23] of PRFs from any one-way function can be easily altered to yield such a puncturable pseudorandom function family.

2.2 Private-Key Single-Input Functional Encryption

A private-key single-input 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 $(\mathsf{FE.S}, \mathsf{FE.KG}, \mathsf{FE.E}, \mathsf{FE.D})$ of probabilistic polynomial-time algorithms. The setup algorithm $\mathsf{FE.S}$ 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 $\mathsf{FE.KG}$ takes as input a master-secret key msk and a single-input function $f \in \mathcal{F}_\lambda$, and outputs a functional key sk_f. The encryption algorithm $\mathsf{FE.E}$ takes as input a master-secret key msk 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 $\mathsf{FE.D}(\mathsf{FE.KG}(\mathsf{msk}, f), \mathsf{FE.E}(\mathsf{msk}, x)) = f(x)$ with all but a negligible probability over the internal randomness of the algorithms $\mathsf{FE.S}$, $\mathsf{FE.KG}$, and $\mathsf{FE.E}$.

In terms of security, we rely on the private-key variant of the existing indistinguishability-based notions for message privacy and function privacy. In fact, following [1,19], our notion of security combines both message privacy and function privacy. When formalizing this notion it would be convenient to use the following standard notion of a *left-or-right oracle*.

Definition 2.3 (Left-or-right oracle). *Let $\mathcal{O}(\cdot,\cdot)$ be a probabilistic two-input functionality. For each $b \in \{0,1\}$ we denote by $\mathcal{O}_b$ the probabilistic three-input functionality $\mathcal{O}_b(k, z_0, z_1) \stackrel{\mathsf{def}}{=} \mathcal{O}(k, z_b)$.*

Intuitively, a private-key functional-encryption scheme is secure if encryptions of messages $x_1, \ldots, x_T$ together with functional keys corresponding to functions $f_1, \ldots, f_T$ reveal essentially no information other than the values $\{f_i(x_j)\}_{i,j\in[T]}$. We consider an adaptive notion of security, to which we refer to as *full security*, in which adversaries are given adaptive access to left-or-right encryption and key-generation oracles.

Definition 2.4 (Full security [1,19]). *A private-key single-input functional encryption scheme* $\mathsf{FE} = (\mathsf{FE.S}, \mathsf{FE.KG}, \mathsf{FE.E}, \mathsf{FE.D})$ *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* fully secure *if for any probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{full1FE}}_{\mathsf{FE},\mathcal{A},\mathcal{F}}(\lambda) \stackrel{\mathsf{def}}{=} & \Big| \Pr\left[\mathcal{A}^{\mathsf{KG}_0(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_0(\mathsf{msk},\cdot,\cdot)}(1^\lambda) = 1\right] \\ & \quad - \Pr\left[\mathcal{A}^{\mathsf{KG}_1(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_1(\mathsf{msk},\cdot,\cdot)}(1^\lambda) = 1\right]\Big| \\ \leq & \ \mathsf{neg}(\lambda)\end{aligned}$$

$(f_0, f_1) \in \mathcal{F}_\lambda \times \mathcal{F}_\lambda$ and $(x_0, x_1) \in \mathcal{X}_\lambda \times \mathcal{X}_\lambda$ with which $\mathcal{A}$ queries the left-or-right key-generation and encryption oracles, respectively, it holds that $f_0(x_0) = f_1(x_1)$. Moreover, the probability is taken over the choice of $\mathsf{msk} \leftarrow \mathsf{FE.S}(1^\lambda)$ *and the internal randomness of $\mathcal{A}$.*

Known Constructions. Private-key single-input functional encryption schemes that satisfy the above notion of full security and support circuits of any a-priori bounded polynomial size are known to exist based on a variety of assumptions.

Ananth et al. [4] gave a generic transformation from selective-message (or selective-function) security to full security. Moreover, Brakerski and Segev [19] showed how to transform any message-private functional encryption scheme into a functional encryption scheme which is fully secure, and the resulting scheme inherits the security guarantees of the original one. Therefore, based on [4,19], given any selective-message (or selective-function) message-private functional encryption scheme we can generically obtain a fully-secure scheme. This implies that schemes that are fully secure for any number of encryption and key-generation queries can be based on indistinguishability obfuscation [21,36], differing-input obfuscation [3,16], and multilinear maps [22]. In addition, schemes that are fully secure for a bounded number $T = T(\lambda)$ of encryption and key-generation queries can be based on the Learning with Errors (LWE) assumption (where the length of ciphertexts grows with T and with a bound on the depth of allowed functions) [25], based on pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with T and with an upper bound on the circuit size of the functions) [26], and even based on one-way functions (for $T = 1$) [26].

2.3 Private-Key Two-Input Functional Encryption

In this section we define the functionality and security of private-key *two-input* functional encryption scheme (we refer the reader to Appendix A for the generalization to t-input schemes for any $t \geq 2$). Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$, $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, and $\mathcal{Z} = \{\mathcal{Z}_\lambda\}_{\lambda\in\mathbb{N}}$ be ensembles of finite sets, and let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of finite two-ary function families. For each $\lambda \in \mathbb{N}$, each function $f \in \mathcal{F}_\lambda$ takes as input two strings, $x \in \mathcal{X}_\lambda$ and $y \in \mathcal{Y}_\lambda$, and outputs a value $f(x, y) \in \mathcal{Z}_\lambda$. A private-key two-input functional encryption scheme Π for $\mathcal{F}$ consists of four probabilistic polynomial time algorithm Setup, Enc, KG and Dec, described as follows.

- $\mathsf{Setup}(1^\lambda)$ – The setup algorithm takes as input the security parameter λ, and outputs a master secret key msk.
- $\mathsf{Enc}(\mathsf{msk}, m, \mathsf{i})$ – The encryption algorithm takes as input a master secret key msk, message input m, and an index $\mathsf{i} \in [2]$, where $m \in \mathcal{X}_\lambda$ if $\mathsf{i} = 1$ and $m \in \mathcal{Y}_\lambda$ if $\mathsf{i} = 2$. It outputs a ciphertext ct_i.
- $\mathsf{KG}(\mathsf{msk}, f)$ – The key-generation algorithm takes as input a master secret key msk and a function $f \in \mathcal{F}_\lambda$, and outputs a functional key sk_f.
- $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_1, \mathsf{ct}_2)$ – The (deterministic) decryption algorithm takes as input a functional key sk_f and two ciphertexts ct_1 and ct_2, and outputs a string $z \in \mathcal{Z}_\lambda \cup \{\perp\}$.

Definition 2.5 (Correctness). *A private-key two-input functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KG}, \mathsf{Dec})$ for $\mathcal{F}$ is* correct *if there exists a negligible* *function* $\mathsf{neg}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, for every* $f \in \mathcal{F}_\lambda$*, and for every* $(x, y) \in \mathcal{X}_\lambda \times \mathcal{Y}_\lambda$*, it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{msk}, x, 1), \mathsf{Enc}(\mathsf{msk}, y, 2)) = f(x, y)\right] \geq 1 - \mathsf{neg}(\lambda),$$

where $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$*,* $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{msk}, f)$*, and the probability is taken over the internal randomness of* $\mathsf{Setup}, \mathsf{Enc}$ *and* KG*.*

Intuitively, we say that a two-input scheme is secure if for any two pairs of messages (x_0, x_1) and (y_0, y_1) that are encrypted with respect to indices $\mathsf{i} = 1$ and $\mathsf{i} = 2$, respectively, and for every pair of functions (f_0, f_1), the triplets $(\mathsf{sk}_{f_0}, \mathsf{Enc}(\mathsf{msk}, x_0, 1), \mathsf{Enc}(\mathsf{msk}, y_0, 2))$ and $(\mathsf{sk}_{f_1}, \mathsf{Enc}(\mathsf{msk}, x_1, 1), \mathsf{Enc}(\mathsf{msk}, y_1, 2))$ are computationally indistinguishable as long as $f_0(x_0, y_0) = f_1(x_1, y_1)$ (note that this considers both message privacy and function privacy). The formal notions of security build upon this intuition and capture the fact that an adversary may in fact hold many functional keys and ciphertexts, and may combine them in an arbitrary manner. As in the case of single-input schemes, we formalize our notions of security using left-or-right key-generation and encryption oracles. Specifically, for each $b \in \{0, 1\}$ and $\mathsf{i} \in \{1, 2\}$ we let $\mathsf{KG}_b(\mathsf{msk}, f_0, f_1) \overset{\mathsf{def}}{=} \mathsf{KG}(\mathsf{msk}, f_b)$ and $\mathsf{Enc}_b(\mathsf{msk}, (m_0, m_1), \mathsf{i}) \overset{\mathsf{def}}{=} \mathsf{Enc}(\mathsf{msk}, m_b, \mathsf{i})$. Before formalizing our notions of security we define the notion of a *valid two-input adversary.*

Definition 2.6 (Valid two-input adversary). *A probabilistic polynomial-time algorithm* $\mathcal{A}$ *is a* valid two-input adversary *if for all private-key two-input functional encryption schemes* $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* $\mathcal{X} \times \mathcal{Y} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}} \times \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ *and a function space* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$*, for all* $\lambda \in \mathbb{N}$ *and* $b \in \{0, 1\}$*, and for all* $(f_0, f_1) \in \mathcal{F}_\lambda$*,* $((x_0, x_1), 1) \in \mathcal{X}_\lambda \times \mathcal{X}_\lambda \times \{1\}$ *and* $((y_0, y_1), 1) \in \mathcal{Y}_\lambda \times \mathcal{Y}_\lambda \times \{2\}$ *with which* $\mathcal{A}$ *queries the left-or-right key-generation and encryption oracles, respectively, it holds that* $f_0(x_0, y_0) = f_1(x_1, y_1)$*.*

We consider two notions of security for two-input schemes, both of which combine message privacy and function privacy. The first notion, *full security*, considers adversaries that have adaptive access to both the encryption oracle and the key-generation oracle. The second notion, *selective-message security*, considers adversaries that must specify all of their encryption queries in advance, but can then have adaptive access to the key-generation oracle. Full security clearly implies selective-message security, and our work shows that the two notions are in fact equivalent for multi-input schemes.

Definition 2.7 (Full security). *A private-key two-input functional encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* $\mathcal{X} \times \mathcal{Y} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}} \times \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ *and a function space* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* fully secure *if for any valid two-input adversary* $\mathcal{A}$ *there exists a negligible function* $\mathsf{neg}(\cdot)$ *such that*

$$\mathsf{Adv}^{\mathsf{full2FE}}_{\Pi,\mathcal{F},\mathcal{A}} \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{full2FE}}_{\Pi,\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{neg}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$*, where the random variable* $\mathsf{Exp}^{\mathsf{full2FE}}_{\Pi,\mathcal{F},\mathcal{A}}(\lambda)$ *is defined via the following experiment:*

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$.
2. $b' \leftarrow \mathcal{A}^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_b(\mathsf{msk},(\cdot,\cdot),\cdot)}\left(1^\lambda,\right)$.
3. *If* $b' = b$ *then output 1, and otherwise output 0.*

Definition 2.8 (Selective-message security). *A private-key two-input functional encryption scheme* $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *over a message space* $\mathcal{X} \times \mathcal{Y} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}} \times \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ *and a function space* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *is* selective-message secure *if for any valid two-input adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mathsf{neg}(\lambda)$ *such that*

$$\mathsf{Adv}^{\mathsf{sel2FE}}_{\Pi,\mathcal{F},\mathcal{A}} \overset{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{sel2FE}}_{\Pi,\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{neg}(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, *where the random variable* $\mathsf{Exp}^{\mathsf{sel2FE}}_{\Pi,\mathcal{F},\mathcal{A}}(\lambda)$ *is defined via the following experiment:*

1. $(\vec{x}, \vec{y}, \mathsf{state}) \leftarrow \mathcal{A}_1\left(1^\lambda\right)$, *where* $\vec{x} = ((x_1^0, x_1^1), \ldots, (x_T^0, x_T^1))$ *and* $\vec{y} = ((y_1^0, y_1^1), \ldots, (y_T^0, y_T^1))$.
2. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$.
3. $\mathsf{ct}_{1,i} \leftarrow \mathsf{Enc}(\mathsf{msk}, x_i^b, 1)$ *and* $\mathsf{ct}_{2,i} \leftarrow \mathsf{Enc}(\mathsf{msk}, y_i^b, 2)$ *for* $i \in [T]$.
4. $b' \leftarrow \mathcal{A}_2^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot)}(1^\lambda, \mathsf{ct}_{1,1}, \ldots, \mathsf{ct}_{1,T}, \mathsf{ct}_{2,1} \ldots, \mathsf{ct}_{2,T}, \mathsf{state})$.
5. *If* $b' = b$ *then output 1, and otherwise output 0.*

Our definitions of a two-input functional encryption scheme is inspired by the definition of [12]. It is a natural generalization of the single-input case and gives rise to an order-revealing encryption. Moreover, as a concrete motivation, a t-input scheme according to the above definition is enough to construct indistinguishability obfuscation for circuits with t input bits [24].[7]

Additional natural ways to define two-input functional encryptions schemes exist. Specifically, Goldwasser et al. [24] considered two such definitions. The first allows to encrypt a message m independently of an index $i \in [2]$. Thus, given a key for a two-input function f and encryptions of two messages x and of y, one can compute both $f(x, y)$ and $f(y, x)$. Hence, this definition requires a stronger "validity requirement" (see Definition 2.6), which means it can support less functionalities. A construction which satisfies our (indexed) definition can be easily transformed into one which satisfies the above (non-indexed) definition by encrypting each message with respect to both indices.

The second, referred to as "multi-client", considers each index as a different "client" and gives each of them his own secret key. In this setting, their security game is quite different, and in particular, an adversary is allowed to obtain the secret keys of a subset of the clients of his choice. The approach underlying our schemes does not seem to directly extend to the multi-client setting, and we leave it as an interesting path for future exploration.

[7] Indeed, [5] get a construction of a t-input scheme for any $t \geq 1$ which implies an indistinguishability obfuscator. Our construction falls short from being generalized to such extent (however, it relies on weaker assumptions).

3 A Selectively-Secure Two-Input Scheme from Any Single-Input Scheme

In this section we construct a private-key two-input functional encryption scheme that is selectively secure. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of two-ary functionalities, where for every $\lambda \in \mathbb{N}$ the set $\mathcal{F}_\lambda$ consists of functions of the form $f : \mathcal{X}_\lambda \times \mathcal{Y}_\lambda \to \mathcal{Z}_\lambda$. Our construction relies on the following building blocks:

1. A private-key single-input functional encryption scheme $\mathsf{1FE} = (\mathsf{1FE.S}, \mathsf{1FE.KG}, \mathsf{1FE.E}, \mathsf{1FE.D})$.
2. A pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$.

As discussed in Sect. 1.1, we assume that the scheme $\mathsf{1FE}$ is sufficiently expressive in the sense that $\mathsf{1FE}$ supports the function family $\mathcal{F}$ (when viewed as a family of single-input functions), the evaluation procedure of the pseudorandom function family PRF, the encryption and key-generation procedures of the private-key functional encryption scheme $\mathsf{1FE}$, and a few additional basic operations. Our scheme $\mathsf{2FE}^{\mathsf{sel}} = (\mathsf{2FE}^{\mathsf{sel}}.\mathsf{S}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{KG}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{E}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{D})$ is defined as follows.

- **The setup algorithm.** On input the security parameter 1^λ the setup algorithm $\mathsf{2FE}^{\mathsf{sel}}.\mathsf{S}$ samples $\mathsf{msk}_{\mathsf{out}}, \mathsf{msk}_{\mathsf{in}} \leftarrow \mathsf{1FE.S}(1^\lambda)$ and outputs $\mathsf{msk} = (\mathsf{msk}_{\mathsf{out}}, \mathsf{msk}_{\mathsf{in}})$.
- **The key-generation algorithm.** On input the master secret key msk and a function $f \in \mathcal{F}_\lambda$, the key-generation algorithm $\mathsf{2FE}^{\mathsf{sel}}.\mathsf{KG}$ samples a random string $z \leftarrow \{0,1\}^\lambda$ and outputs $\mathsf{sk}_f \leftarrow \mathsf{1FE.KG}(\mathsf{msk}_{\mathsf{out}}, D_{f,\perp,z,\perp})$, where $D_{f,\perp,z,\perp}$ is a single-input function that is defined in Fig. 2.
- **The encryption algorithm.** On input the master secret key msk, a message m and an index $\mathsf{i} \in [2]$, the encryption algorithm $\mathsf{2FE}^{\mathsf{sel}}.\mathsf{E}$ has two cases:
  - If $(m, \mathsf{i}) = (x, 1)$, it samples a master secret key $\mathsf{msk}^\star \leftarrow \mathsf{1FE.S}(1^\lambda)$, a PRF key $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, and a random string $s \in \{0,1\}^\lambda$, and then outputs a pair $(\mathsf{ct}_1, \mathsf{sk}_1)$ defined as follows:
  $$\mathsf{ct}_1 \leftarrow \mathsf{1FE.E}(\mathsf{msk}_{\mathsf{out}}, (\mathsf{msk}^\star, K, 0))$$
  $$\mathsf{sk}_1 \leftarrow \mathsf{1FE.KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x,\perp,0,s,\mathsf{msk}^\star,K}),$$
  where $\mathsf{AGG}_{x,\perp,0,s,\mathsf{msk}^\star,K}$ is a single-input function that is defined in Fig. 3.
  - If $(m, \mathsf{i}) = (y, 2)$, it samples a random string $t \in \{0,1\}^\lambda$, and outputs
  $$\mathsf{ct}_2 \leftarrow \mathsf{1FE.E}(\mathsf{msk}_{\mathsf{in}}, (y, \perp, t, \perp, \perp)).$$
- **The decryption algorithm.** On input a functional key sk_f and two ciphertexts, $(\mathsf{ct}_1, \mathsf{sk}_1)$ and ct_2, the decryption algorithm $\mathsf{2FE}^{\mathsf{sel}}.\mathsf{D}$ computes $\mathsf{ct}' = \mathsf{1FE.D}(\mathsf{sk}_1, \mathsf{ct}_2)$, $\mathsf{sk}' = \mathsf{1FE.D}(\mathsf{sk}_f, \mathsf{ct}_1)$ and outputs $\mathsf{1FE.D}(\mathsf{sk}', \mathsf{ct}')$.

The correctness of the above scheme with respect to any family of two-ary functionalities follows in a straightforward manner from the correctness of the underlying functional encryption scheme $\mathsf{1FE}$. Specifically, consider any pair of

```
$D_{f_0,f_1,z,u}((\mathsf{msk}^\star, K, w))$:
  1. If $\mathsf{msk}^\star = \bot$, output $u$ and HALT.
  2. Compute $r = \mathsf{PRF.Eval}(K, z)$.
  3. Output $\mathsf{1FE.KG}(\mathsf{msk}^\star, C_{f_w}; r)$.

$C_f((x, y))$:
  1. Output $f(x, y)$.
```

Fig. 2. The single-input functions $D_{f_0,f_1,z,u}$ and C_f.

```
$\mathsf{AGG}_{x_0,x_1,a,s,\mathsf{msk}^\star,K}((y_0, y_1, t, s', v))$:
  1. If $s' = s$ output $v$ and HALT.
  2. Compute $r = \mathsf{PRF.Eval}(K, t)$.
  3. Output $\mathsf{1FE.E}(\mathsf{msk}^\star, (x_a, y_a); r)$.
```

Fig. 3. The single-input function $\mathsf{AGG}_{x_0,x_1,a,s,\mathsf{msk}^\star,K}$.

messages x and y and any function f. The encryption of x with respect to the index i =1 and the encryption of y with respect to the index $\mathsf{i} = 2$ result in ciphertexts $(\mathsf{ct}_1, \mathsf{sk}_1)$ and ct_2, respectively. Using the correctness of the scheme 1FE, by executing $\mathsf{1FE.D}(\mathsf{sk}_1, \mathsf{ct}_2)$ we obtain an encryption ct' of the message (x, y) under the key $\mathsf{msk}^\star$. In addition, by executing $\mathsf{1FE.D}(\mathsf{sk}_f, \mathsf{ct}_1)$ we obtain a functional key sk' for C_f under the key $\mathsf{msk}^\star$. Therefore, executing $\mathsf{1FE.D}(\mathsf{sk}', \mathsf{ct}')$ outputs the value $C_f((x, y)) = f(x, y)$ as required.

The following theorem captures the security of the scheme, stating that under suitable assumptions on the underlying building blocks, the two-input scheme $\mathsf{2FE}^{\mathsf{sel}}$ is selective-message secure (see Definition 2.8). We refer the reader to the full version [18] for the complete proof.

Theorem 3.1. *Assuming that (1)* 1FE *is fully secure, and (2)* PRF *is a pseudorandom function family, then* $\mathsf{2FE}^{\mathsf{sel}}$ *is selective-message secure.*

We note that for proving that $\mathsf{2FE}^{\mathsf{sel}}$ is selective-message secure it suffices to require selective-message security from 1FE. However, given the generic transformations of Ananth et al. [4] (from selective security to adaptive security) and of Brakerski and Segev [19] (from message security to full security), for simplifying the proof of Theorem 3.1 we assume that 1FE is fully secure. In addition, when assuming that 1FE is fully secure, the scheme $\mathsf{2FE}^{\mathsf{sel}}$ can be shown to satisfy a notion of security that seems in between selective-message security and full security. Specifically, this notion considers adversaries that first have adaptive access to encryptions only for the first coordinate, and then have adaptive access to encryptions only for the second coordinate (while having adaptive access to the key-generation oracle throughout the experiment). However, given our generic transformation from selective-message security to full security for multi-input schemes (see Sect. 4), for simplifying the proof of Theorem 3.1 we focus on proving selective-message security.

In addition, for concreteness we focus on the unbounded case where the underlying scheme supports an unbounded (i.e., not fixed in advance) number of key-generation queries and encryption queries. More generally, the proof of Theorem 3.1 shows that if the scheme corresponding to $\mathsf{msk}_{\mathsf{out}}$ supports T_1 encryption queries and T_2 key-generation queries, the scheme corresponding to $\mathsf{msk}_{\mathsf{in}}$ supports T_3 encryption queries and T_4 key-generation queries, and the scheme corresponding to each $\mathsf{msk}^{\star}$ supports T_5 encryption queries and T_6 key-generation queries, then the resulting scheme $\mathsf{2FE}^{\mathsf{sel}}$ supports $\min\{T_1, T_4, T_5\}$ encryption queries with respect to index $\mathsf{i} = 1$, $\min\{T_3, T_5\}$ encryption queries with respect to index $\mathsf{i} = 2$ and $\min\{T_2, T_6\}$ key-generation queries. When the polynomials $T_1, \ldots, T_6$ are known in advance (i.e., do not depend on the adversary), such schemes are known to exist based on the LWE assumption or even only one-way functions (see Sect. 2.2 for a more elaborated discussion of the existing schemes).

4 From Selective to Adaptive Security for Two-Input Schemes

In this section we show how to transform any private-key selective-message secure two-input functional encryption scheme (see Definition 2.8) into a fully secure one (see Definition 2.7). Our construction relies on the following building blocks:

1. A private-key single-input functional encryption scheme $\mathsf{1FE} = (\mathsf{1FE.S}, \mathsf{1FE.KG}, \mathsf{1FE.E}, \mathsf{1FE.D})$.
2. A private-key two-input functional encryption scheme $\mathsf{2FE}^{\mathsf{sel}} = (\mathsf{2FE}^{\mathsf{sel}}.\mathsf{S}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{KG}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{E}, \mathsf{2FE}^{\mathsf{sel}}.\mathsf{D})$.
3. A puncturable pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$.

We assume that the schemes $\mathsf{1FE}$ and $\mathsf{2FE}^{\mathsf{sel}}$ are sufficiently expressive in the sense that they support the function family $\mathcal{F}$ (when viewed as a family of single-input functions), the evaluation procedure of the pseudorandom function family PRF, the setup, encryption and key-generation procedures of the scheme $\mathsf{1FE}$, and a few additional basic operations. The scheme $\mathsf{2FE} = (\mathsf{2FE.S}, \mathsf{2FE.KG}, \mathsf{2FE.E}, \mathsf{2FE.D})$ is defined as follows.

- **The setup algorithm.** On input the security parameter 1^{λ} the setup algorithm $\mathsf{2FE.S}$ samples $\mathsf{msk}_1 \leftarrow \mathsf{1FE.S}(1^{\lambda})$ and $\mathsf{msk}_2 \leftarrow \mathsf{2FE}^{\mathsf{sel}}.\mathsf{S}(1^{\lambda})$ and then outputs $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2)$.
- **The key-generation algorithm.** On input the master secret key msk and a function $f \in \mathcal{F}_{\lambda}$, the key-generation algorithm $\mathsf{2FE.KG}$ outputs $\mathsf{sk}_f \leftarrow \mathsf{2FE}^{\mathsf{sel}}.\mathsf{KG}(\mathsf{msk}_2, D_{f,\perp,1,\perp,\perp,\perp})$, where $D_{f,\perp,1,\perp,\perp,\perp}$ is a two-input function that is defined in Fig. 4.
- **The encryption algorithm.** On input the master secret key msk, a message m and an index $\mathsf{i} \in [2]$, the encryption algorithm $\mathsf{2FE.E}$ has two cases:

- If $(m, \mathsf{i}) = (x, 1)$, it samples $s \leftarrow \{0,1\}^\lambda$ uniformly at random, three PRF keys $K^{\mathsf{enc}}, K^{\mathsf{key}}, K^{\mathsf{msk}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and outputs a pair $(\mathsf{ct}_1, \mathsf{sk}_1)$ defined as follows:

$$\mathsf{ct}_1 \leftarrow \mathsf{2FE}^{\mathsf{sel}}.\mathsf{E}(\mathsf{msk}_2, (K^{\mathsf{msk}}, K^{\mathsf{key}}, s, 0), 1)$$
$$\mathsf{sk}_1 \leftarrow \mathsf{1FE.KG}(\mathsf{msk}_1, \mathsf{AGG}_{x,\perp,0,s,K^{\mathsf{msk}},K^{\mathsf{enc}},\perp,\perp})$$

where the single-input function $\mathsf{AGG}_{x,\perp,0,s,K^{\mathsf{msk}},K^{\mathsf{enc}},\perp,\perp}$ is defined in Fig. 5.
- If $(m, \mathsf{i}) = (y, 2)$, it samples $t \leftarrow \{0,1\}^\lambda$ uniformly at random and outputs a pair $(\mathsf{ct}_2, \mathsf{ct}_3)$ defined as follows:

$$\mathsf{ct}_2 \leftarrow \mathsf{2FE}^{\mathsf{sel}}.\mathsf{E}(\mathsf{msk}_2, (1, t), 2)$$
$$\mathsf{ct}_3 \leftarrow \mathsf{1FE.E}(\mathsf{msk}_1, (y, \perp, 1, t, \perp, \perp)).$$

– **The decryption algorithm.** On input a functional key sk_f and two ciphertexts $(\mathsf{ct}_1, \mathsf{sk}_1)$ and $(\mathsf{ct}_2, \mathsf{ct}_3)$, the decryption algorithm $\mathsf{2FE.D}$ first computes the value $\mathsf{sk}' = \mathsf{2FE}^{\mathsf{sel}}.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1, \mathsf{ct}_2)$, then it computes the value $\mathsf{ct}' = \mathsf{1FE.D}(\mathsf{sk}_1, \mathsf{ct}_3)$, and finally it outputs $\mathsf{1FE.D}(\mathsf{sk}', \mathsf{ct}')$.

```
$D_{f_0,f_1,\mathsf{c},s',t',u}((K^{\mathsf{msk}}, K^{\mathsf{key}}, s, \mathsf{thr}), (\mathsf{c}', t))$:
  1. If $s' = s$ and $t' = t$, output $u$ and HALT.
  2. Compute $r = \mathsf{PRF.Eval}(K^{\mathsf{msk}}, t)$.
  3. Compute $r' = \mathsf{PRF.Eval}(K^{\mathsf{key}}, t)$.
  4. Compute $\mathsf{msk}_{s,t} = \mathsf{1FE.S}(1^\lambda; r)$.
  5. If $\mathsf{c} \leq \mathsf{thr}$ and $\mathsf{c}' \leq \mathsf{thr}$ set $f = f_1$.
  6. Else (if $\mathsf{c} > \mathsf{thr}$ or $\mathsf{c}' > \mathsf{thr}$) set $f = f_0$.
  7. Output $\mathsf{1FE.KG}(\mathsf{msk}_{s,t}, C_f; r')$.

$C_f((x, y))$:
  1. Output $f(x, y)$.
```

Fig. 4. The two-input function $D_{f_0,f_1,\mathsf{c},s',t',u}$ and the single-input function C_f.

The correctness of the above scheme with respect to any family of two-ary functionalities follows in a straightforward manner from the correctness of the underlying functional encryption schemes $\mathsf{1FE}$ and $\mathsf{2FE}^{\mathsf{sel}}$. Specifically, consider any pair of messages x and y and any function f. The encryption of x with respect to the index i =1 and the encryption of y with respect to the index $\mathsf{i} = 2$ result in ciphertexts $(\mathsf{ct}_1, \mathsf{sk}_1)$ and $(\mathsf{ct}_2, \mathsf{ct}_3)$, respectively. Using the correctness of the scheme $\mathsf{2FE}^{\mathsf{sel}}$, by executing $\mathsf{2FE}^{\mathsf{sel}}.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1, \mathsf{ct}_2)$ we obtain a functional key sk' for C_f under the key $\mathsf{msk}_{s,t}$. In addition, by executing $\mathsf{1FE.D}(\mathsf{sk}_1, \mathsf{ct}_3)$ we obtain a an encryption ct' of (x, y) under the key $\mathsf{msk}_{s,t}$. Therefore, executing $\mathsf{1FE.D}(\mathsf{sk}', \mathsf{ct}')$ outputs the value $C_f((x, y)) = f(x, y)$ as required.

```
$\mathbf{AGG}_{x_0,x_1,\mathsf{thr},s,K^{\mathsf{msk}},K^{\mathsf{enc}},t',v'}((y_0,y_1,\mathsf{c},t,s',u'))$:
  1. If $t' = t$ output $v'$ and HALT.
  2. If $s' = s$ output $u'$ and HALT.
  3. Compute $r = \mathsf{PRF.Eval}(K^{\mathsf{msk}}, t)$.
  4. Compute $r' = \mathsf{PRF.Eval}(K^{\mathsf{enc}}, t)$.
  5. Compute $\mathsf{msk}_{s,t} = \mathsf{1FE.S}(1^\lambda; r)$.
  6. If $\mathsf{c} \leq \mathsf{thr}$ set $x = x_1$ and $y = y_1$.
  7. Else (if $\mathsf{c} > \mathsf{thr}$) set $x = x_0$ and $y = y_0$.
  8. Output $\mathsf{1FE.E}(\mathsf{msk}_{s,t}, (x, y); r')$.
```

Fig. 5. The single-input function $\mathsf{AGG}_{x_0,x_1,\mathsf{thr},s,K^{\mathsf{msk}},K^{\mathsf{enc}},t',v'}$.

The following theorem captures the security of the scheme. This theorem states that under suitable assumptions on the underlying building blocks, the two-input scheme $\mathsf{2FE}$ is fully secure (see Definition 2.7). We refer the reader to the full version [18] for the complete proof.

Theorem 4.1. *Assuming that (1)* $\mathsf{1FE}$ *is fully secure, (2)* $\mathsf{2FE}^{\mathsf{sel}}$ *is selective-message secure, and (3)* PRF *is a puncturable pseudorandom function family, then* $\mathsf{2FE}$ *is fully secure.*

As in Sect. 3, for concreteness we focus on the unbounded case where the underlying schemes, $\mathsf{1FE}$ and $\mathsf{2FE}^{\mathsf{sel}}$, support an unbounded (i.e., not fixed in advance) number of key-generation queries and encryption queries. More generally, the proof of Theorem 4.1 shows that if the scheme corresponding to msk_1 supports T_1 encryption queries and T_2 key-generation queries, the scheme corresponding to msk_2 supports $T_3^{(1)}$ encryption queries with respect to index $\mathsf{i} = 1$ and $T_3^{(2)}$ encryption queries with respect to index $\mathsf{i} = 2$, and T_4 key-generation queries, and the scheme corresponding to each $\mathsf{msk}_{s,t}$ supports a *single* encryption query and T_5 key-generation queries, then the resulting scheme $\mathsf{2FE}$ supports $\min\{T_2, T_3^{(1)}\}$ encryption queries with respect to index $\mathsf{i} = 1$, $\min\{T_1, T_3^{(2)}\}$ encryption queries with respect to index $\mathsf{i} = 2$ and $\min\{T_4, T_5\}$ key-generation queries. When the polynomials $T_1, T_2, T_3^{(1)}, T_3^{(2)}, T_4$ and T_5 are known in advance (i.e., do not depend on the adversary), such schemes are known to exist based on the LWE assumption or even only one-way functions (see Sect. 2.2 for a more elaborated discussion of the existing schemes).

Acknowledgments. We thank Eylon Yogev for various insightful discussions and the EUROCRYPT '16 reviewers for their useful comments.

A Generalization to $t \geq 2$ Inputs

In this section we generalize our results to more than two inputs. In Appendix A.1 we generalize the definitions introduced in Sect. 2.3, and in Appendices A.2

and A.3 we generalize the constructions from Sects. 3 and 4, respectively. More precisely, in Appendix A.2 we show how to obtain a *selectively-secure* t-input scheme assuming any fully secure $(t-1)$-input scheme. Then, in Appendix A.3 we show how to obtain a *fully-secure* t-input scheme assuming any fully-secure $(t-1)$-input scheme and a selectively-secure t-input scheme.

A.1 Private-Key t-Input Functional Encryption

In this section we generalize the framework introduced in Sect. 2.3 to the general case of t-input schemes (Sect. 2.3 dealt with the case $t = 2$).

For $i \in [t]$ let $\mathcal{X}_i = \{(\mathcal{X}_i)_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of finite sets, and let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be an ensemble of finite t-ary function families. For each $\lambda \in \mathbb{N}$, each function $f \in \mathcal{F}_\lambda$ takes as input t strings, $x_1 \in (\mathcal{X}_1)_\lambda, \ldots, x_t \in (\mathcal{X}_t)_\lambda$, and outputs a value $f(x_1, \ldots, x_t) \in \mathcal{Z}_\lambda$. A private-key t-input functional encryption scheme Π for $\mathcal{F}$ consists of four probabilistic polynomial time algorithm Setup, Enc, KG and Dec, described as follows. The setup algorithm $\mathsf{Setup}(1^\lambda)$ takes as input the security parameter λ, and outputs a master secret key msk. The encryption algorithm $\mathsf{Enc}(\mathsf{msk}, m, \mathsf{i})$ takes as input a master secret key msk, a message m, and an index $\mathsf{i} \in [t]$, where $m \in (\mathcal{X}_\mathsf{i})_\lambda$, and outputs a ciphertext ct_i. The key-generation algorithm $\mathsf{KG}(\mathsf{msk}, f)$ takes as input a master secret key msk and a function $f \in \mathcal{F}_\lambda$, and outputs a functional key sk_f. The (deterministic) decryption algorithm Dec takes as input a functional key sk_f and t ciphertexts, $\mathsf{ct}_1, \ldots, \mathsf{ct}_t$, and outputs a string $z \in \mathcal{Z}_\lambda \cup \{\perp\}$.

Definition A.1 (Correctness). *A private-key t-input functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{KG}, \mathsf{Dec})$ for $\mathcal{F}$ is* correct *if there exists a negligible function $\mathsf{neg}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, for every $f \in \mathcal{F}_\lambda$, and for every $(x_1, \ldots, x_t) \in (\mathcal{X}_1)_\lambda \times \cdots \times (\mathcal{X}_t)_\lambda$, it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, \mathsf{Enc}(\mathsf{msk}, x_1, 1), \ldots, \mathsf{Enc}(\mathsf{msk}, x_t, t)) = f(x_1, \ldots, x_t)\right] \geq 1 - \mathsf{neg}(\lambda),$$

where $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{msk}, f)$, and the probability is taken over the internal randomness of $\mathsf{Setup}, \mathsf{Enc}$ and KG.

Next, we generalize the security definitions from Sect. 2.3 to the t-input case. As in Sect. 2.3, we start by defining the notion of a *valid t-input adversary.* Then, we define *full security* and *selective-message security.*

Definition A.2 (Valid t-input adversary). *A probabilistic polynomial-time algorithm $\mathcal{A}$ is a* valid t-input adversary *if for all private-key t-input functional encryption schemes $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_t = \{(\mathcal{X}_1)_\lambda\}_{\lambda \in \mathbb{N}} \times \cdots \times \{(\mathcal{X}_t)_\lambda\}_{\lambda \in \mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$, for all $\lambda \in \mathbb{N}$ and $b \in \{0, 1\}$, and for all $(f_0, f_1) \in \mathcal{F}_\lambda$ and $((x_i^0, x_i^1), i) \in \mathcal{X}_i \times \mathcal{X}_i \times \{i\}$ (where $i \in [t]$) with which $\mathcal{A}$ queries the left-or-right key-generation and encryption oracles, respectively, it holds that $f_0(x_1^0, \ldots, x_t^0) = f_1(x_1^1, \ldots, x_t^1)$.*

Definition A.3 (Full security). *A private-key t-input functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_t = \{(\mathcal{X}_1)_\lambda\}_{\lambda\in\mathbb{N}} \times \cdots \times \{(\mathcal{X}_t)_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is* fully secure *if for any valid t-input adversary $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that*

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

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

1. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$.
2. $b' \leftarrow \mathcal{A}^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot),\mathsf{Enc}_b(\mathsf{msk},(\cdot,\cdot),\cdot)}\left(1^\lambda\right)$.
3. *If $b' = b$ then output 1, and otherwise output 0.*

Definition A.4 (Selective-message security). *A private-key t-input functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_t = \{(\mathcal{X}_1)_\lambda\}_{\lambda\in\mathbb{N}} \times \cdots \times \{(\mathcal{X}_t)_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is* selective-message secure *if for any valid t-input adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ there exists a negligible function $\mathsf{neg}(\lambda)$ such that*

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

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

1. $(\vec{x_1}, \ldots, \vec{x_t}, \mathsf{state}) \leftarrow \mathcal{A}_1\left(1^\lambda\right)$, *where* $\vec{x_i} = ((x^0_{i,1}, x^1_{i,1}), \ldots, (x^0_{i,T}, x^1_{i,T}))$ *for* $i \in [t]$.
2. $\mathsf{msk} \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0,1\}$.
3. $\mathsf{ct}_{i,j} \leftarrow \mathsf{Enc}(\mathsf{msk}, x^b_{i,j}, 1)$ *for* $i \in [t]$ *and* $j \in [T]$.
4. $b' \leftarrow \mathcal{A}_2^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot)}(1^\lambda, \{\mathsf{ct}_{i,j}\}_{i\in[t],j\in[T]}, \mathsf{state})$.
5. *If $b' = b$ then output 1, and otherwise output 0.*

A.2 A Selectively-Secure t-Input Scheme from any $(t-1)$-Input Scheme

In this section we generalize the construction from Sect. 3 by presenting a construction of a selectively-secure t-input scheme assuming any fully-secure $(t-1)$-input scheme. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of t-input functionalities, where for every $\lambda \in \mathbb{N}$ the set $\mathcal{F}_\lambda$ consists of functions of the form $f : (\mathcal{X}_1)_\lambda \times \cdots \times (\mathcal{X}_t)_\lambda \to \mathcal{Z}_\lambda$. Our construction relies on the following building blocks:

1. A private-key single-input functional encryption scheme $\mathsf{FE}_1 = (\mathsf{FE}_1.\mathsf{S}, \mathsf{FE}_1.\mathsf{KG}, \mathsf{FE}_1.\mathsf{E}, \mathsf{FE}_1.\mathsf{D})$.

2. A private-key $(t-1)$-input functional encryption scheme $\mathsf{FE}^{\mathsf{sel}}_{t-1} = (\mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{S}, \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{KG}, \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{E}, \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{D})$.
3. A pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$.

Our scheme $\mathsf{FE}^{\mathsf{sel}}_{t} = (\mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{S}, \mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{KG}, \mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{E}, \mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{D})$ is defined as follows.

- **The setup algorithm.** On input the security parameter 1^λ the setup algorithm $\mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{S}$ samples $\mathsf{msk}_{\mathsf{out}} \leftarrow \mathsf{FE}_1.\mathsf{S}(1^\lambda)$,$\mathsf{msk}_{\mathsf{in}} \leftarrow \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{S}(1^\lambda)$ and outputs $\mathsf{msk} = (\mathsf{msk}_{\mathsf{out}}, \mathsf{msk}_{\mathsf{in}})$.
- **The key-generation algorithm.** On input the master secret key msk and a function $f \in \mathcal{F}_\lambda$, the key-generation algorithm $\mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{KG}$ samples a random string $z \leftarrow \{0,1\}^\lambda$ and outputs $\mathsf{sk}_f \leftarrow \mathsf{FE}_1.\mathsf{KG}(\mathsf{msk}_{\mathsf{out}}, D_{f,\perp,z,\perp})$, where $D_{f,\perp,z,\perp}$ is a single-input function that is defined in Fig. 6.
- **The encryption algorithm.** On input the master secret key msk, a message m and an index $\mathsf{i} \in [t]$, the encryption algorithm $\mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{E}$ has two cases:
  - If $(m, \mathsf{i}) = (x_1, 1)$, it samples a master secret key $\mathsf{msk}^\star \leftarrow \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{S}(1^\lambda)$, a PRF key $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, and a random string $s \in \{0,1\}^\lambda$, and then outputs a pair $(\mathsf{ct}_1, \mathsf{sk}_1)$ defined as follows:

$$\mathsf{ct}_1 \leftarrow \mathsf{FE}_1.\mathsf{E}(\mathsf{msk}_{\mathsf{out}}, (\mathsf{msk}^\star, K, 0))$$
$$\mathsf{sk}_1 \leftarrow \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x_1,\perp,0,s,\mathsf{msk}^\star,K}),$$

  where $\mathsf{AGG}_{x,\perp,0,\mathsf{msk}^\star,K}$ is a $(t-1)$-input function that is defined in Fig. 7.
  - If $(m, \mathsf{i}) = (x_i, i)$ where $i \in \{2, \dots, t\}$, it samples a random string $\tau_i \in \{0,1\}^\lambda$, and outputs

$$\mathsf{ct}_i \leftarrow \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x_i, \perp, \tau_i, \perp, \perp), i-1).$$

- **The decryption algorithm.** On input a functional key sk_f and ciphertexts $(\mathsf{ct}_1, \mathsf{sk}_1), \mathsf{ct}_2, \dots, \mathsf{ct}_t$, the decryption algorithm $\mathsf{FE}^{\mathsf{sel}}_{t}.\mathsf{D}$ computes $(\mathsf{ct}'_2, \dots, \mathsf{ct}'_t) = \mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{D}(\mathsf{sk}_1, (\mathsf{ct}_2, \dots, \mathsf{ct}_t))$, $\mathsf{sk}' = \mathsf{FE}_1.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1)$ and outputs $\mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{D}(\mathsf{sk}', (\mathsf{ct}'_2, \dots, \mathsf{ct}'_t))$.

```
$D_{f_0,f_1,z,u}((\mathsf{msk}^\star, K, w))$:
  1. If $\mathsf{msk}^\star = \perp$, output $u$ and HALT.
  2. Compute $r = \mathsf{PRF.Eval}(K, z)$.
  3. Output $\mathsf{FE}^{\mathsf{sel}}_{t-1}.\mathsf{KG}(\mathsf{msk}^\star, C_{f_w}; r)$.

$C_f((x_1, x_2), x_3, \dots, x_t)$:
  1. Output $f(x_1, \dots, x_t)$.
```

Fig. 6. The single-input function $D_{f_0,f_1,z,u}$ and the $(t-1)$-input function C_f.

```
$\mathbf{AGG}_{x_1^0,x_1^1,a,s,\mathsf{msk}^\star,K}((x_2^0,x_2^1,\tau_2,s_2,v_2),\ldots,(x_t^0,x_t^1,\tau_t,s_t,v_t))$:
  1. If $s_2 = \cdots = s_t = s$ output $(v_2,\ldots,v_t)$ and HALT.
  2. Set $x_i = x_i^a$ for all $i \in [t]$.
  3. Compute $r_i = \mathsf{PRF.Eval}(K,\tau_i)$ for $2 \leq i \leq t$.
  4. Output $(\mathsf{FE}_{t-1}^{\mathsf{sel}}.\mathsf{E}(\mathsf{msk}^\star,(x_1,x_2),1;r_2),\mathsf{FE}_{t-1}^{\mathsf{sel}}.\mathsf{E}(\mathsf{msk}^\star,x_3,2;r_3),\ldots,$
     $\mathsf{FE}_{t-1}^{\mathsf{sel}}.\mathsf{E}(\mathsf{msk}^\star,x_t,t-1;r_t))$.
```

Fig. 7. The $(t-1)$-input function $\mathsf{AGG}_{x_1^0,x_1^1,a,s,\mathsf{msk}^\star,K}$.

Theorem A.5. *Assuming that (1)* FE_1 *is fully secure, (2)* $\mathsf{FE}_{t-1}^{\mathsf{sel}}$ *is selective-message secure, and (3)* PRF *is a pseudorandom function family, then* $\mathsf{FE}_t^{\mathsf{sel}}$ *is selective-message secure.*

As in Theorem 3.1, we note that for proving that $\mathsf{FE}_t^{\mathsf{sel}}$ is selective-message secure it suffices to require selective-message security from FE_1. However, given the generic transformation for single-input schemes [4,19] (from selective security to adaptive security and from message security to full security, respectively), for simplifying the proof of Theorem A.5 we assume that FE_1 is fully secure. We refer the reader to the full version [18] for the complete proof.

A.3 From Selective to Adaptive Security for *t*-Input Schemes

In this section we generalize the construction from Sect. 4 to get a fully-secure t-input functional encryption scheme assuming any fully-secure $(t-1)$-input functional encryption scheme and any selectively-secure t-input functional encryption scheme. Our construction relies on the following building blocks:

1. A private-key single-input functional encryption scheme $\mathsf{FE}_1 = (\mathsf{FE}_1.\mathsf{S}, \mathsf{FE}_1.\mathsf{KG}, \mathsf{FE}_1.\mathsf{E}, \mathsf{FE}_1.\mathsf{D})$.
2. A private-key $(t-1)$-input functional encryption scheme $\mathsf{FE}_{t-1} = (\mathsf{FE}_{t-1}.\mathsf{S}, \mathsf{FE}_{t-1}.\mathsf{KG}, \mathsf{FE}_{t-1}.\mathsf{E}, \mathsf{FE}_{t-1}.\mathsf{D})$.
3. A private-key t-input functional encryption scheme $\mathsf{FE}_t^{\mathsf{sel}} = (\mathsf{FE}_t^{\mathsf{sel}}.\mathsf{S}, \mathsf{FE}_t^{\mathsf{sel}}.\mathsf{KG}, \mathsf{FE}_t^{\mathsf{sel}}.\mathsf{E}, \mathsf{FE}_t^{\mathsf{sel}}.\mathsf{D})$.
4. A puncturable pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval}, \mathsf{PRF.Punc})$.

The scheme $\mathsf{FE}_t = (\mathsf{FE}_t.\mathsf{S}, \mathsf{FE}_t.\mathsf{KG}, \mathsf{FE}_t.\mathsf{E}, \mathsf{FE}_t.\mathsf{D})$ is defined as follows.

- **The setup algorithm.** On input the security parameter 1^λ the setup algorithm $\mathsf{FE}_t.\mathsf{S}$ samples $\mathsf{msk}_{t-1} \leftarrow \mathsf{FE}_{t-1}.\mathsf{S}(1^\lambda)$ and $\mathsf{msk}_t \leftarrow \mathsf{FE}_t^{\mathsf{sel}}.\mathsf{S}(1^\lambda)$ and then outputs $\mathsf{msk} = (\mathsf{msk}_{t-1}, \mathsf{msk}_t)$.
- **The key-generation algorithm.** On input the master secret key msk and a function $f \in \mathcal{F}_\lambda$, the key-generation algorithm $\mathsf{FE}_t.\mathsf{KG}$ outputs $\mathsf{sk}_f \leftarrow \mathsf{FE}_t^{\mathsf{sel}}.\mathsf{KG}(\mathsf{msk}_t, D_{f,\perp,1,\underbrace{\perp,\ldots,\perp}_{t \text{ times}},\perp})$, where $D_{f,\perp,1,\underbrace{\perp,\ldots,\perp}_{t \text{ times}},\perp}$ is a t-input function that is defined in Fig. 8.

- **The encryption algorithm.** On input the master secret key msk, a message m and an index $\mathsf{i} \in [2]$, the encryption algorithm $\mathsf{FE}_{t-1}.\mathsf{E}$ has two cases:
  - If $(m, \mathsf{i}) = (x_1, 1)$, it samples $\tau_1 \leftarrow \{0,1\}^\lambda$ uniformly at random, three PRF keys $K^{\mathsf{enc}}, K^{\mathsf{key}}, K^{\mathsf{msk}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and outputs a pair $(\mathsf{ct}_1, \mathsf{sk}_1)$ defined as follows:

$$\mathsf{ct}_1 \leftarrow \mathsf{FE}^{\mathsf{sel}}_t.\mathsf{E}(\mathsf{msk}_t, (K^{\mathsf{msk}}, K^{\mathsf{key}}, \tau_1, \underbrace{0, \dots, 0}_{t-1 \text{ times}}), 1)$$

$$\mathsf{sk}_1 \leftarrow \mathsf{FE}_{t-1}.\mathsf{KG}(\mathsf{msk}_{t-1}, \mathsf{AGG}_{x_1, \perp, \underbrace{0, \dots, 0}_{t-1 \text{ times}}, \tau_1, K^{\mathsf{msk}}, K^{\mathsf{enc}}, \underbrace{\perp, \dots, \perp}_{t-1 \text{ times}}, \perp})$$

  where the single-input function $\mathsf{AGG}_{x_1, \perp, \underbrace{0, \dots, 0}_{t-1 \text{ times}}, \tau_1, K^{\mathsf{msk}}, K^{\mathsf{enc}}, \underbrace{\perp, \dots, \perp}_{t-1 \text{ times}}, \perp}$ is defined in Fig. 9.
  - If $(m, \mathsf{i}) = (x_i, i)$ and $i > 1$, it samples $\tau_i \leftarrow \{0,1\}^\lambda$ uniformly at random and outputs a pair $(\mathsf{ct}_i, \mathsf{ct}'_i)$ defined as follows:

$$\mathsf{ct}_i \leftarrow \mathsf{FE}^{\mathsf{sel}}_t.\mathsf{E}(\mathsf{msk}_t, (1, \tau_i), i)$$
$$\mathsf{ct}'_i \leftarrow \mathsf{FE}_{t-1}.\mathsf{E}(\mathsf{msk}_{t-1}, (x_i, \perp, 1, \tau_i, \underbrace{\perp, \dots, \perp}_{t-1 \text{ times}}, \perp), i-1).$$

- **The decryption algorithm.** On input a functional key sk_f and t ciphertexts $(\mathsf{ct}_1, \mathsf{sk}_1)$ and $(\mathsf{ct}_2, \mathsf{ct}'_2), \dots, (\mathsf{ct}_t, \mathsf{ct}'_t)$, the decryption algorithm $\mathsf{FE}_t.\mathsf{D}$ first computes the value $\mathsf{sk}' = \mathsf{FE}^{\mathsf{sel}}_t.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1, \dots, \mathsf{ct}_t)$, then it computes the value $\mathsf{ct}' = \mathsf{FE}_{t-1}.\mathsf{D}(\mathsf{sk}_1, \mathsf{ct}'_2, \dots, \mathsf{ct}'_t)$, and finally it outputs $\mathsf{FE}_1.\mathsf{D}(\mathsf{sk}', \mathsf{ct}')$.

```
$D_{f_0, f_1, \mathsf{c}, \tau'_1, \dots, \tau'_t, u}((K^{\mathsf{msk}}, K^{\mathsf{key}}, \tau_1, \mathsf{thr}_2, \dots, \mathsf{thr}_t), (\mathsf{c}_2, \tau_2), \dots, (\mathsf{c}_t, \tau_t))$:
 1. If $\tau'_i = \tau_i$ for all $i \in [t]$, output $u$ and HALT.
 2. Compute $r = \mathsf{PRF.Eval}(K^{\mathsf{msk}}, \tau_2 \dots \tau_t)$.
 3. Compute $r' = \mathsf{PRF.Eval}(K^{\mathsf{key}}, \tau_2 \dots \tau_t)$.
 4. Compute $\mathsf{msk}_{\tau_1, \dots, \tau_t} = \mathsf{FE}_1.\mathsf{S}(1^\lambda; r)$.
 5. For $i = 1, \dots, t$ do:
    (a) If $\mathsf{c}_i < \mathsf{thr}_i$ then set $f = f_1$ and exit loop.
    (b) If $\mathsf{c}_i > \mathsf{thr}_i$ then set $f = f_0$ and exit loop.
    (c) If $\mathsf{c}_i = \mathsf{thr}_i$ and $i < t$ continue to next iteration (with $i = i + 1$).
    (d) If $\mathsf{c}_i = \mathsf{thr}_i$ and $i = t$ set $f = f_1$.
 6. Output $\mathsf{FE}_1.\mathsf{KG}(\mathsf{msk}_{\tau_1, \dots, \tau_t}, C_f; r')$.

$C_f((x_1, \dots, x_t))$:
 1. Output $f(x_1, \dots, x_t)$.
```

Fig. 8. The t-input function $D_{f_0, f_1, \mathsf{c}, \tau'_1, \dots, \tau'_t, u}$ and the single-input function C_f.

```
$\mathbf{AGG}_{x_1^0, x_1^1, \mathsf{thr}_2, \ldots, \mathsf{thr}_t, \tau_1, K^{\mathsf{msk}}, K^{\mathsf{enc}}, \tau_{1,2}, \ldots, \tau_{1,t}, u_1}$
    $\boldsymbol{((x_2^0, x_2^1, \mathsf{c}_2, \tau_2, \tau_{2,1}, \ldots, \tau_{2,t}, u_2), \ldots,}$
    $\boldsymbol{(x_t^0, x_t^1, \mathsf{c}_t, \tau_t, \tau_{t,1}, \ldots, \tau_{t,t}, u_t)) :}$

1. If $\exists i \in [t]$ such that $\forall j \in [t] \setminus \{i\}$ it holds that $\tau_{i,j} = \tau_j$, then output $u_i$ and HALT.
2. Compute $r = \mathsf{PRF.Eval}(K^{\mathsf{msk}}, \tau_2 \ldots, \tau_t)$.
3. Compute $r' = \mathsf{PRF.Eval}(K^{\mathsf{enc}}, \tau_2 \ldots, \tau_t)$.
4. Compute $\mathsf{msk}_{\tau_1, \ldots, \tau_t} = \mathsf{FE}_1.\mathsf{S}(1^\lambda; r)$.
5. For $i = 1, \ldots, t$ do:
   (a) If $\mathsf{c}_i < \mathsf{thr}_i$ then set $x_i = x_i^1$ for all $i \in [t]$ and exit loop.
   (b) If $\mathsf{c}_i > \mathsf{thr}_i$ then set $x_i = x_i^0$ for all $i \in [t]$ and exit loop.
   (c) If $\mathsf{c}_i = \mathsf{thr}_i$ and $i < t$ continue to next iteration (with $i = i + 1$).
   (d) If $\mathsf{c}_i = \mathsf{thr}_i$ and $i = t$ set $x_i = x_i^1$ for all $i \in [t]$.
6. Output $\mathsf{FE}_1.\mathsf{E}(\mathsf{msk}_{\tau_1, \ldots, \tau_t}, (x_1 \ldots, x_t); r')$.
```

Fig. 9. The t-input function $\mathsf{AGG}_{x_1^0, x_1^1, \mathsf{thr}_2, \ldots, \mathsf{thr}_t, \tau_1, K^{\mathsf{msk}}, K^{\mathsf{enc}}, \tau'_{1,2}, \ldots, \tau'_{1,t}, u_1}$.

The following theorem captures the security of the scheme. This theorem states that under suitable assumptions on the underlying building blocks, the t-input scheme FE_t is fully private (see Definition 2.7). We refer the reader to the full version [18] for the complete proof.

Theorem A.6. *Let $t > 1$ be any fixed integer. Assuming that (1) FE_1 is fully secure, (2) FE_{t-1} is fully secure, (3) $\mathsf{FE}_t^{\mathsf{sel}}$ is selective-message secure, and (4) PRF is a puncturable pseudorandom function family, then FE_t is fully secure.*

We note that the proof of Theorem A.6 assumes that t is a fixed constant. The reason for this limitation is that the number of hybrids in the proof of security is $\lambda^{O(t)}$, where λ is the security parameter, which is polynomial for any constant t. If we assume that the underlying building blocks are sub-exponentially secure, then the proof of Theorem A.6 can be used for a super-constant number of inputs.

References

1. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: Function private functional encryption and property preserving encryption: New definitions and positive results. Cryptology ePrint Archive, Report 2013/744 (2013)
2. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013)
3. Ananth, P., Boneh, D., Garg, S., Sahai, A., Zhandry, M.: Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689 (2013)

4. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 657–677. Springer, Heidelberg (2015)
5. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015)
6. Ananth, P., Jain, A., Sahai, A.: Achieving compactness generically: Indistinguishability obfuscation from non-compact functional encryption. Cryptology ePrint Archive, Report 2015/730 (2015)
7. Asharov, G., Segev, G.: Limits on the power of indistinguishability obfuscation and functional encryption. In: Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, pp. 191–209 (2015)
8. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. J. ACM **59**(2), 6 (2012)
9. Bellare, M., O'Neill, A.: Semantically-secure functional encryption: possibility results, impossibility results and the quest for a general definition. In: Abdalla, M., Nita-Rotaru, C., Dahab, R. (eds.) CANS 2013. LNCS, vol. 8257, pp. 218–234. Springer, Heidelberg (2013)
10. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science, pp. 171–190 (2015)
11. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
12. Boneh, D., Lewi, K., Raykova, M., Sahai, A., Zhandry, M., Zimmerman, J.: Semantically secure order-revealing encryption: multi-input functional encryption without obfuscation. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 563–594. Springer, Heidelberg (2015)
13. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: hiding the function in functional encryption. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 461–478. Springer, Heidelberg (2013)
14. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
15. Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Heidelberg (2013)
16. Boyle, E., Chung, K.-M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (2014)
17. Boyle, E., Goldwasser, S., Ivan, I.: Functional signatures and pseudorandom functions. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 501–519. Springer, Heidelberg (2014)
18. Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. Cryptology ePrint Archive, Report 2015/158 (2015)
19. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (2015)
20. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)

21. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, pp. 40–49 (2013)
22. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_18
23. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4), 792–807 (1986)
24. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014)
25. Goldwasser, S., Kalai, Y., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: Reusable garbled circuits and succinct functional encryption. In: Proceedings of the 45th Annual ACM Symposium on Theory of Computing, pp. 555–564 (2013)
26. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption with bounded collusions via multi-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 162–179. Springer, Heidelberg (2012)
27. Iovino, V., Zebrowski, K.: Mergeable functional encryption. Cryptology ePrint Archive, Report 2015/103 (2015)
28. Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings of the 20th Annual ACM Conference on Computer and Communications Security, pp. 669–684 (2013)
29. Komargodski, I., Moran, T., Naor, M., Pass, R., Rosen, A., Yogev, E.: One-way functions and (im)perfect obfuscation. In: Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science, pp. 374–383 (2014)
30. Komargodski, I., Segev, G., Yogev, E.: Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 352–377. Springer, Heidelberg (2015)
31. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
32. Sahai, A., Waters, B.: Slides on functional encryption (2008). http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt
33. Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pp. 475–484 (2014)
34. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
35. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
36. Waters, B.: A punctured programming approach to adaptively secure functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9216, pp. 678–697. Springer, Heidelberg (2015)