J Cryptol (2020) 33:406–458
https://doi.org/10.1007/s00145-019-09327-x

Journal of
CRYPTOLOGY

From Minicrypt to Obfustopia via Private-Key Functional Encryption

Ilan Komargodski*
Cornell Tech, New York, USA
komargodski@cornell.edu

Gil Segev†
School of Computer Science and Engineering, Hebrew University of Jerusalem, 91904 Jerusalem, Israel
segev@cs.huji.ac.il

Communicated by Rafail Ostrovsky.

Received 9 August 2017 / Revised 21 December 2018
Online publication 12 June 2019

Abstract. Private-key functional encryption enables fine-grained access to symmetrically encrypted data. Although private-key functional encryption (supporting an unbounded number of keys and ciphertexts) seems significantly weaker than its public-key variant, its known realizations all rely on public-key functional encryption. At the same time, however, up until recently it was not known to imply any public-key primitive, demonstrating our poor understanding of this primitive. Bitansky et al. (Theory of cryptography—14th international conference, TCC 2016-B, 2016) showed that sub-exponentially secure private-key function encryption bridges from nearly exponential security in Minicrypt to slightly super-polynomial security in Cryptomania, and from sub-exponential security in Cryptomania to Obfustopia. Specifically, given any sub-exponentially secure private-key functional encryption scheme and a nearly exponentially secure one-way function, they constructed a public-key encryption scheme with slightly super-polynomial security. Assuming, in addition, a sub-exponentially secure public-key encryption scheme, they then constructed an indistinguishability obfuscator (or a public-key functional encryption scheme if the given building blocks are polynomially secure).

We show that quasi-polynomially secure private-key functional encryption bridges from sub-exponential security in Minicrypt all the way to Cryptomania. First, given any quasi-polynomially secure private-key functional encryption scheme, we construct an indistinguishability obfuscator for circuits with inputs of poly-logarithmic length.

*Supported in part by a Packard Foundation Fellowship and by an AFOSR Grant FA9550-15-1-0262. Work done while being a Ph.D. student at the Weizmann Institute of Science, supported by grants from the Israel Science Foundation (No. 950/16) and by a Levzion Fellowship.

†Supported by the European Union's 7th Framework Program (FP7) via a Marie Curie Career Integration Grant, by the European Union's Horizon 2020 Framework Program (H2020) via an ERC Grant (Grant No. 714253), 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.

Then, we observe that such an obfuscator can be used to instantiate many natural applications of indistinguishability obfuscation. Specifically, relying on sub-exponentially secure one-way functions, we show that quasi-polynomially secure private-key functional encryption implies not just public-key encryption but leads all the way to public-key functional encryption for circuits with inputs of poly-logarithmic length. Moreover, relying on sub-exponentially secure injective one-way functions, we show that quasi-polynomially secure private-key functional encryption implies a hard-on-average distribution over instances of a PPAD-complete problem. Underlying our constructions is a new transformation from single-input functional encryption to multi-input functional encryption in the private-key setting. The previously known such transformation (Brakerski et al. J Cryptol 31(2):434–520, 2018) required a sub-exponentially secure single-input scheme, and obtained a scheme supporting only a slightly super-constant number of inputs. Our transformation both relaxes the underlying assumption and supports more inputs: Given any quasi-polynomially secure single-input scheme, we obtain a scheme supporting a poly-logarithmic number of inputs.

Keywords. Private-key functional encryption, Multi-input functional encryption, PPAD hardness, Indistinguishability obfuscation.

1. Introduction

Functional encryption [20,49,52] allows tremendous flexibility when accessing encrypted data: Such encryption schemes support restricted decryption keys that allow users to learn specific functions of the encrypted data without leaking any additional information. We focus on the most general setting where the functional encryption schemes support an unbounded number of functional keys in the public-key setting, and an unbounded number of functional keys and ciphertexts in the private-key setting. In the public-key setting, it has been shown that functional encryption is essentially equivalent to indistinguishability obfuscation [4,6,22,33,54], and thus, it currently seems somewhat challenging to base its security on standard cryptographic assumptions (especially given the various attacks on obfuscation schemes and their underlying building blocks [10,25–28,40,48]—see [5, Appendix A] for a summary of these attacks).

Luckily, when examining the various applications of functional encryption (see, for example, the survey by Boneh et al. [21]), it turns out that *private-key* functional encryption suffices in many interesting scenarios.[1] However, although private-key functional encryption may seem significantly weaker than its public-key variant, constructions of private-key functional encryption schemes are currently known based only on public-key functional encryption.[2]

Minicrypt, Cryptomania, or Obfustopia? For obtaining a better understanding of private-key functional encryption, we must be able to position it correctly within the hierarchy of cryptographic primitives. Up until recently, private-key functional encryption

[1] As a concrete (yet quite general) example, consider a user who stores her data on a remote server: The user uses the master secret key both for encrypting her data, and for generating functional keys that will enable the server to offer her various useful services.

[2] This is not true in various restricted cases, for example, when the functional encryption scheme has to support an a priori bounded number of functional keys or ciphertexts [39]. However, as mentioned, we focus on schemes that support an unbounded number of functional keys and ciphertexts.

was not known to imply any cryptographic primitives other than those that are essentially equivalent to one-way functions (i.e., Minicrypt primitives [42]). Moreover, Asharov and Segev [8] proved that as long as a private-key functional encryption scheme is invoked in a black-box manner, it cannot be used as a building block to construct any public-key primitive (i.e., Cryptomania primitives [42]).[3] This initial evidence hinted that private-key functional encryption may belong to Minicrypt, and thus may be constructed based on extremely well-studied cryptographic assumptions.

Recently, Bitansky et al. [15] showed that private-key functional encryption is more powerful than suggested by the above initial evidence. First, any sub-exponentially secure private-key functional encryption scheme and any (nearly) exponentially secure one-way function can be used to construct a public-key encryption scheme.[4] Although their underlying building blocks are at least sub-exponentially secure, the resulting public-key scheme is only slightly super-polynomially secure. Second, any sub-exponentially secure private-key functional encryption scheme and any sub-exponentially secure public-key encryption scheme imply a full-fledged indistinguishability obfuscator. Lastly, in the polynomial security regime, together with a result of [38,47], any private-key functional encryption scheme and any public-key encryption scheme (and PRFs in NC^1) can be used to construct a full-fledged *public-key* functional encryption scheme.

Overall, it is known that sub-exponentially secure private-key functional encryption bridges from nearly exponential security in Minicrypt to slightly super-polynomial security in Cryptomania, and from (sub-exponential security in) Cryptomania to Obfustopia (see Fig. 1). The question we are interested in is whether private-key functional encryption can bridge, by itself, from Minicrypt to Obfustopia, without assuming additional "Cryptomaniac" assumptions.

1.1. *Our Contributions*

We show that quasi-polynomially secure private-key functional encryption bridges from sub-exponential security in Minicrypt all the way to Obfustopia. Given any *quasi-polynomially* secure private-key functional encryption scheme, we construct a (quasi-polynomially secure) indistinguishability obfuscator for circuits with inputs of poly-logarithmic length and sub-polynomial size.

Theorem 1.1. (Informal) *Assuming a quasi-polynomially secure private-key functional encryption scheme for polynomial-size circuits, there exists an indistinguishability obfuscator for the class of circuits of size* $2^{(\log \lambda)^{\epsilon}}$ *with inputs of length* $(\log \lambda)^{1+\delta}$ *bits, for some positive constants* ϵ *and* δ.

Underlying our obfuscator is a new transformation from single-input functional encryption to multi-input functional encryption in the private-key setting. The previously known such transformation of Brakerski et al. [13] required a sub-exponentially secure

[3] This holds even if the construction is allowed to generate functional keys (in a non-black-box manner) for any circuit that invokes one-way functions in a black-box manner.

[4] Bitansky et al. overcome the black-box barrier introduced by Asharov and Segev [8] by relying on the non-black-box construction of a private-key multi-input functional encryption scheme of Brakerski et al. [13].

single-input scheme, and obtained a multi-input scheme supporting only a slightly super-constant number of inputs. Our transformation both relaxes the underlying assumption and supports more inputs: Given any quasi-polynomially secure single-input scheme, we obtain a multi-input scheme supporting a poly-logarithmic number of inputs.

We demonstrate the wide applicability of our obfuscator by observing that it can be used to instantiate natural applications of (full-fledged) indistinguishability obfuscation for polynomial-size circuits. The kind of applications where our obfuscator can be used are those where the obfuscated circuit gets as input a string whose length is proportional to a different security parameter. In this case, we can assume stronger security on the primitive that uses this input and thus have a shorter input. We exemplify this observation by constructing a public-key functional encryption scheme (based on [54]), and a hard-on-average distribution of instances of a PPAD-complete problem (based on [16]).

Theorem 1.2. (Informal) *Assuming a quasi-polynomially secure private-key functional encryption scheme for polynomial-size circuits, and a sub-exponentially secure one-way function, there exists a public-key functional encryption scheme for the class of circuits of size* $2^{(\log \lambda)^{\epsilon}}$ *with inputs of length* $(\log \lambda)^{1+\delta}$ *bits, for some positive constants* ϵ *and* δ.

Theorem 1.3. (Informal) *Assuming a quasi-polynomially secure private-key functional encryption scheme for polynomial-size circuits, and a sub-exponentially secure injective one-way function, there exists a hard-on-average distribution over instances of a PPAD-complete problem.*

Compared to the work of Bitansky at el. [15], Theorem 1.2 shows that private-key functional encryption implies not just public-key encryption but leads all the way to public-key functional encryption. Furthermore, in terms of underlying assumptions, whereas Bitansky et al. assume a sub-exponentially secure private-key functional encryption scheme and a (nearly) exponentially secure one-way function, we only assume a quasi-polynomially secure private-key functional encryption scheme and a sub-exponentially secure one-way function.

In addition, recall that average-case PPAD hardness was previously shown based on compact *public-key* functional encryption (or indistinguishability obfuscation) for polynomial-size circuits and one-way permutations [37]. We show average-case PPAD hardness based on quasi-polynomially secure *private-key* functional encryption and sub-exponentially secure injective one-way function. In fact, as shown by Hubáček and Yogev [41], our result (as well as [16,37]) implies average-case hardness for CLS, a proper subclass of PPAD and PLS [31]. See Fig. 1 for an illustration of our results.

1.2. *Overview of Our Constructions*

In this section, we provide a high-level overview of our constructions. First, we recall the functionality and security requirements of multi-input functional encryption (MIFE) in the private-key setting, and explain the main ideas underlying our new construction of a multi-input scheme. Then, we describe the obfuscator we obtain from our multi-input scheme, and briefly discuss its applications to public-key functional encryption and to average-case PPAD hardness.

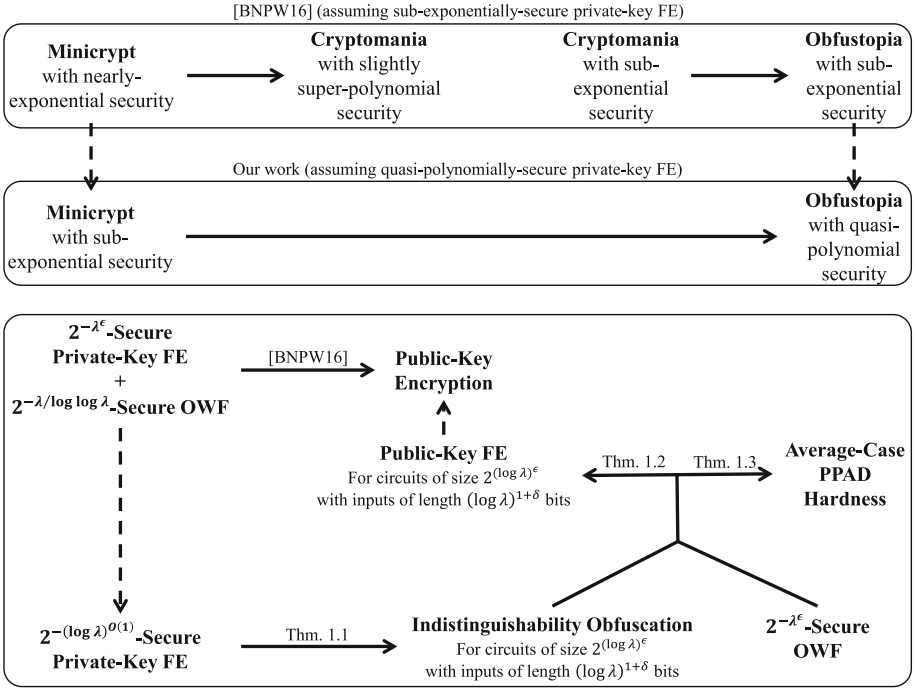

Fig. 1. An illustration of our results (dashed arrows correspond to trivial implications).

Multi-input Functional Encryption In a private-key t-input functional encryption scheme [32], the master secret key msk of the scheme is used for encrypting any message x_i to the ith coordinate, and for generating functional keys for t-input functions. A functional key sk_f corresponding to a function f enables to compute $f(x_1, \ldots, x_t)$ given $\mathsf{Enc}(x_1, 1), \ldots, \mathsf{Enc}(x_t, t)$. Building upon the previous notions of security for private-key multi-input functional encryption schemes [14,32], we consider a strengthened notion of security that combines both message privacy and function privacy (as in [1,19] for single-input schemes and as in [4,13] for multi-input schemes), to which we refer as *full security*. Specifically, we consider adversaries that are given access to "left or right" key-generation and encryption oracles.[5] 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 the function f_b. The encryption oracle receives as input triples of the form (x^0, x^1, i), and outputs an encryption of the message x^b with respect to coordinate i. We require that no efficient adversary can guess the bit b with probability noticeably higher than $1/2$, as long as for each such $t+1$ queries $(f_0, f_1), (x_1^0, x_1^1), \ldots, (x_t^0, x_t^1)$ it holds that $f_0(x_1^0, \ldots, x_t^0) = f_1(x_1^1, \ldots, x_t^1)$.

[5] In this work, we focus on selectively secure schemes, where an adversary first submits all of its encryption queries, and can then adaptively interact with the key-generation oracle (see Definition 2.7). This notion of security suffices for the applications we consider in this paper.

The BKS Approach Given any private-key single-input functional encryption scheme for all polynomial-size circuits, Brakerski et al. [13] constructed a $t(\lambda)$-input scheme for all circuits of size $s(\lambda) = 2^{(\log \lambda)^\epsilon}$, where $t(\lambda) = \delta \cdot \log \log \lambda$ for some fixed positive constants ϵ and δ, and $\lambda \in \mathbb{N}$ is the security parameter.

Their transformation is based on extending the number of inputs the scheme supports one by one. That is, for any $t \geq 1$, given a t-input scheme they construct a $(t+1)$-input scheme. Relying on the function privacy of the underlying scheme, Brakerski et al. observed that ciphertexts for one of the coordinates can be treated as a functional key for a function that has the value of the input hardwired. In terms of functionality, this idea enabled them to support $t+1$ inputs using a scheme that supports t inputs. The transformation is implemented such that every step of it incurs a *polynomial* blowup in the size of the ciphertexts and functional keys.[6] Thus, applying this transformation t times, the size of a functional key for a function of size s is roughly $(s \cdot \lambda)^{O(1)^t}$. Therefore, Brakerski et al. could only apply their transformation $t(\lambda) = \delta \cdot \log \log \lambda$ times, and this required assuming that their underlying single-input scheme is sub-exponentially secure, and that $s(\lambda) = 2^{(\log \lambda)^\epsilon}$.

Our Construction We present a new transformation that constructs a $2t$-inputs scheme directly from any t-input scheme. Our transformation shares the same polynomial efficiency loss as in [13], so applying the transformation t times makes a functional key be of size $(s \cdot \lambda)^{O(1)^t}$. But now, since each transformation doubles the number of inputs, applying the transformation t times gets us all the way to a scheme that supports $2^t = (\log \lambda)^\delta$ inputs, as required. We further observe, by a careful security analysis, that for the resulting scheme to be secure it suffices that the initial scheme is only *quasi-polynomially secure* (and the resulting scheme can be made quasi-polynomially secure as well).

Doubling the Number of Inputs via Dynamic Key Encapsulation As opposed to the approach of [13] (and the similar idea of [4]), it is much less clear how to combine the ciphertexts and functional keys of a t-input scheme to satisfy the required functionality (and security) of a $2t$-input scheme.

Our high-level idea is as follows. Given a $2t$-input function f, we will generate a functional key for a function f^* that gets t inputs each of which is composed of two inputs: $f^*(x_1 \| x_{1+t}, \ldots, x_t \| x_{2t}) = f(x_1, \ldots, x_{2t})$. We will encrypt each input such that it is possible to compute an encryption of each pair $(x_\ell, x_{\ell+t})$, and evaluate the function in two steps. First, we concatenate each such pair to get an encryption of $x_\ell \| x_{\ell+t}$. Then, given such t ciphertexts, we will apply a functional key that corresponds to f^*. By the correctness of the underlying primitives, the output must be correct. There are three main issues that we have to overcome: (1) We need to be able to generate the encryption of $x_\ell \| x_{\ell+t}$, (2) we need to make sure all of these ciphertexts are with respect to the same master secret key and that the functional key for f^* is also generated with respect to the same key, and (3) we need to prove the security of the resulting scheme. We now describe our solution.

[6] A similar strategy was also employed by Ananth and Jain [4] that showed how to use any t-input private-key scheme to get a private-key $(t+1)$-input scheme under the additional assumption that a *public-key* functional encryption scheme exists. Their construction, however, did not incur the polynomial blowup and could be applied all the way to get a scheme that supports a polynomial number of inputs.

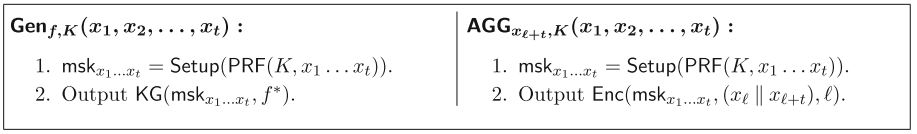

Fig. 2. The t-input functions $\mathsf{Gen}_{f,K}$ and $\mathsf{AGG}_{x_{\ell+t},K}$.

The master secret key for our scheme is a master secret key for a t-input scheme msk and a PRF key K. We split the $2t$ input coordinates into two parts: (1) the first t coordinates $1, \ldots, t$ which we call the "master coordinates" and (2) the last t coordinates $1 + t, \ldots, 2t$ which we call the "slave coordinates". Our main idea is to let each combination of the master coordinates implicitly define a master secret "encapsulation" key $\mathsf{msk}_{x_1 \ldots, x_t}$ for a t-input scheme. Details follow.

To encrypt a message x_ℓ with respect to a master coordinate $1 \leq \ell \leq t$, we encrypt x_ℓ with respect to coordinate ℓ under the key msk. To encrypt a message $x_{\ell+t}$ with respect to a slave coordinate $1 \leq \ell \leq t$, we generate a functional key for a t-input function $\mathsf{AGG}_{x_{\ell+t},K}$ under the key msk. To generate a functional key for a $2t$-input function f, we generate a functional key for a t-input function $\mathsf{Gen}_{f,K}$ under msk. Both $\mathsf{AGG}_{x_{\ell+t},K}$ and $\mathsf{Gen}_{f,K}$ first compute a pseudorandom master secret key $\mathsf{msk}_{x_1 \ldots x_t}$ using randomness generated via the PRF key K on input $x_1 \ldots x_t$. Then, $\mathsf{AGG}_{x_{\ell+t},K}$ computes an encryption of $(x_\ell \parallel x_{\ell+t})$ to coordinate ℓ under this master secret key, and $\mathsf{Gen}_{f,K}$ computes a functional key for f^* (described above) under this master secret key (see Fig. 2).

It is straightforward to verify that the above scheme indeed provides the required functionality of a $2t$-input scheme. Indeed, given t ciphertexts corresponding to the master coordinates $\mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_t}$, t ciphertexts corresponding to the slave coordinates $\mathsf{ct}_{x_{1+t}}, \ldots, \mathsf{ct}_{x_{2t}}$, and a functional key sk_f for a $2t$-input function f, we first combine $\mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_t}$ with each $\mathsf{ct}_{x_{\ell+t}}$ to get $\mathsf{ct}_{x_\ell \| x_{\ell+t}}$, which is an encryption of $x_\ell \parallel x_{\ell+t}$ under $\mathsf{msk}_{x_1 \ldots x_t}$. Then, we combine $\mathsf{ct}_{x_1}, \ldots, \mathsf{ct}_{x_t}$ with sk_f to get a functional key sk_{f^*} for f^* under the same $\mathsf{msk}_{x_1 \ldots x_t}$. Finally, we combine $\mathsf{ct}_{x_1 \| x_{1+t}}, \ldots, \mathsf{ct}_{x_t \| x_{2t}}$ with sk_{f^*} to get $f^*(x_1 \parallel x_{1+t}, \ldots, x_t \parallel x_{2t}) = f(x_1, \ldots, x_{2t})$, as required.

The security proof is done by a sequence of hybrid experiments, where we "attack" each possible sequence of master coordinates separately, namely we handle each $\mathsf{msk}_{x_1 \ldots x_t}$ separately so that it will not be explicitly needed. A typical approach for such a security proof is to embed all possible encryptions and key-generation queries under $\mathsf{msk}_{x_1 \ldots x_t}$ in the ciphertexts that are generated under msk. Handling the key-generation queries using $\mathsf{msk}_{x_1 \ldots x_t}$ is rather standard: Whenever a key-generation query is requested we compute the corresponding functional key under $\mathsf{msk}_{x_1 \ldots x_t}$ and embed it into the functional key. Handling encryption queries under $\mathsf{msk}_{x_1 \ldots x_t}$ is significantly more challenging since for every $x_1 \ldots x_t$ sequence, there are many possible ciphertexts $x_{\ell+t}$ of slave coordinates that will be paired with it to get the encryption of $x_\ell \parallel x_{\ell+t}$. It might seem as if there is not enough space to embed all these possible ciphertexts, but we observe that we can embed each ciphertext $\mathsf{ct}_{x_\ell \| x_{\ell+t}}$ in the ciphertext corresponding to $x_{\ell+t}$ (for each such $x_{\ell+t}$). This way, $\mathsf{msk}_{x_1 \ldots x_t}$ is not explicitly needed in the scheme and we can use the security of the underlying t-input scheme. In total, the number of hybrids is roughly T^t, where T is an upper bound on the running time of the adversary.

Thus, since t is roughly logarithmic in the security parameter, we have to start with a quasi-polynomially secure scheme.

From MIFE to Obfuscation Goldwasser et al. [32] observed that multi-input functional encryption is tightly related to indistinguishability obfuscation [11,33]. Specifically, a multi-input scheme that supports a polynomial number of inputs (i.e., $t(\lambda) = \mathsf{poly}(\lambda)$) readily implies an indistinguishability obfuscator (and vice-versa). We use a more fine-grained relationship (as observed by Bitansky et al. [15]) that is useful when $t(\lambda)$ is small compared to λ: A multi-input scheme that supports all circuits of size $s(\lambda)$ and $t(\lambda)$ inputs implies an indistinguishability obfuscator for all circuits of size $s(\lambda)$ that have at most $t(\lambda) \cdot \log \lambda$ input bits.

This transformation works as follows. An obfuscation of a function f of circuit size at most $s(\lambda)$ that has at most $t(\lambda) \cdot \log \lambda$ bits as input, is composed of $t(\lambda) \cdot \lambda$ ciphertexts and one functional key. We think of f as a function f^* that gets $t(\lambda)$ inputs each of which is of length $\log \lambda$ bits. The obfuscation now consists of a functional key for the circuit f^*, denoted by $\mathsf{sk}_f = \mathsf{KG}(f^*)$, and a ciphertext $\mathsf{ct}_{x,i} = \mathsf{Enc}(x, i)$ for every $(x, i) \in \{0,1\}^{\log \lambda} \times [t(\lambda)]$. To evaluate C at a point $x = (x_1 \dots x_{t(\lambda)}) \in (\{0,1\}^{\log \lambda})^{t(\lambda)}$ one has to compute and output $\mathsf{Dec}(\mathsf{sk}_f, \mathsf{ct}_{x_1,1}, \dots, \mathsf{ct}_{x_{t(\lambda)},t(\lambda)}) = f(x)$. Correctness and security of the obfuscator follow directly from the correctness and security of the multi-input scheme.

Given the relationship described above and given our multi-input scheme that supports circuits of size at most $s(\lambda) = 2^{(\log \lambda)^\epsilon}$ that have $t(\lambda) = (\log \lambda)^\delta$ inputs for some fixed positive constants ϵ and δ, we obtain Theorem 1.1.

Applications of Our Obfuscator One of the main conceptual contributions of this work is the observation that an indistinguishability obfuscator as described above (that supports circuits with a poly-logarithmic number of input bits) is in fact sufficient for many of the applications of indistinguishability obfuscation for all polynomial-size circuits. We exemplify this observation by showing how to adapt the construction of Waters [54] of a public-key functional encryption scheme and the construction of Bitansky et al. [16] of a hard-on-average distribution for PPAD, to our obfuscator. Such an adaptation is quite delicate and involves a careful choice of the additional primitives that are involved in the construction. In a very high level, since the obfuscator supports only a poly-logarithmic number of inputs, a primitive that has to be secure when applied on (part of) the input (say a one-way function), must be sub-exponentially secure. We believe that this observation may find additional applications beyond the scope of our work.

Using the Multi-input Scheme of [13]. Using the multi-input scheme of [13], one can get that sub-exponentially secure private-key functional encryption implies indistinguishability obfuscation for inputs of length slightly super-logarithmic. However, using such an obfuscator as a building block seems to inherently require to additionally assume nearly exponentially secure primitives and the resulting primitives are (at most) slightly super-polynomially secure.

Our approach, on the other hand, requires quasi-polynomially secure private-key functional encryption. In addition, our additional primitives are only sub-exponentially secure and the resulting primitives are quasi-polynomially secure.

1.3. *Additional Related Work*

Constructions of FE Schemes Private-key *single-input* functional encryption schemes that are sufficient for our applications are known to exist based on a variety of assumptions, including indistinguishability obfuscation [33,54], differing-input obfuscation [2,9], and multilinear maps [34]. Restricted functional encryption schemes that support either a bounded number of functional keys or a bounded number of ciphertexts can be based on the learning with errors (LWE) assumption (where the length of ciphertexts grows with the number of functional-key queries and with a bound on the depth of allowed functions) [36], and even based on pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with the number of functional-key queries and with an upper bound on the circuit size of the functions) [39].

In the work of Bitansky et al. [15, Proposition 1.2 & Footnote 1], it has been shown that assuming weak PRFs in NC^1, any public-key encryption scheme can be used to transform a private-key functional encryption scheme into a public-key functional encryption scheme (which can be used to get PPAD hardness [37]). This gives a better reduction than ours in terms of security loss, but requires a public-key primitive to begin with.

Constructions of MIFE Schemes There are several constructions of private-key multi-input functional encryption schemes. Mostly related to our work is the construction of Brakerski et al. [13] which we significantly improve (see Sect. 1.2 for more details). Other constructions [4,14,32] are incomparable as they either rely on stronger assumptions or could be proven secure only in an idealized generic model. Goldwasser et al. [32] constructed a multi-input scheme that supports a polynomial number of inputs assuming indistinguishability obfuscation for all polynomial-size circuits. Ananth and Jain [4] constructed a multi-input functional encryption scheme that supports a polynomial number of inputs assuming any sub-exponentially secure (single-input) *public-key* functional encryption scheme. Boneh et al. [14] constructed a multi-input scheme that supports a polynomial number of inputs based on multilinear maps, and was proven secure in the idealized generic multilinear map model.

Proof Techniques Parts of our proof rely on two useful techniques from the functional encryption literature: key encapsulation (also known as "hybrid encryption") and function privacy.

Key encapsulation 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. The usefulness of this technique in the context of functional encryption was demonstrated by Ananth et al. [3] and Brakerski et al. [13]. 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. 1.2, we use key encapsulation techniques for our *dynamic key-generation* technique, a crucial ingredient in our constructions and proofs of security.

The security guarantees of functional encryption typically focus on *message privacy* that ensures that a ciphertext does not reveal any unnecessary information on the

plaintext. In various cases, however, it is also useful to consider *function privacy* [1,17–19,51], asking that a functional key sk_f does not reveal any unnecessary information on the function f. Brakerski and Segev [19] (and the follow-up of Ananth and Jain [4]) showed that any private-key (multi-input) functional encryption scheme can be generically transformed into one that satisfies both message privacy and function privacy. Function privacy was found useful as a building block in the construction of several functional encryption schemes [3,13,46]. In particular, 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 [53]).

Follow-Up Work In a recent work, Kitagawa et al. [44] showed that indistinguishability obfuscation for *all* circuits can be constructed from sub-exponentially secure private-key functional encryption without any further assumptions. Their technique is quite different from ours. Roughly speaking, they replace the public-key functional encryption scheme in the construction of indistinguishability obfuscation of Bitansky and Vaikuntanathan [22] with a primitive called *puncturable private-key functional encryption* and show how to generically construct it from any private-key functional encryption scheme.

A question that is still left open in this line of work is whether a public-key functional encryption scheme can be generically constructed from a private-key one with only a polynomial loss in security.

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 our construction of a private-key multi-input functional encryption scheme based on any single-input scheme. In Sect. 4, we present our construction of an indistinguishability obfuscator for circuits with inputs of poly-logarithmic length, and its applications to public-key functional encryption and average-case PPAD hardness.

2. Preliminaries

In this section, we present the notation and basic definitions that are used in this work. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. For 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 $\{1, \ldots, n\}$.

Throughout the paper, we denote by λ the security parameter. A function $\mathsf{neg} : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for every constant $c > 0$ there exists an integer N_c such that $\mathsf{neg}(\lambda) < \lambda^{-c}$ for all $\lambda > N_c$. Two sequences of random variables $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *computationally indistinguishable* if for any probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\mathsf{neg}(\cdot)$ such that $\left|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]\right| \leq \mathsf{neg}(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

2.1. *One-Way Functions and Pseudorandom Generators*

We rely on the standard (parameterized) notions of one-way functions and pseudorandom generators.

Definition 2.1. (*One-way function*) An efficiently computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is (t,μ)-$\mathsf{one\text{-}way}$ if for every probabilistic algorithm $\mathcal{A}$ that runs in time $t = t(\lambda)$ it holds that

$$\mathsf{Adv}^{\mathsf{OWF}}_{f,\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \Pr_{x\leftarrow\{0,1\}^\lambda}[\mathcal{A}(1^\lambda, f(x)) \in f^{-1}(f(x))] \leq \mu(\lambda),$$

for all sufficiently large $\lambda \in \mathbb{N}$, where the probability is taken over the choice of $x \in \{0,1\}^\lambda$ and over the internal randomness of $\mathcal{A}$.

Whenever $t = t(\lambda)$ is a super-polynomial function and $\mu = \mu(\lambda)$ is a negligible function, we will often omit t and μ and simply call the function *one-way*. In case $t(\lambda) = 1/\mu(\lambda) = 2^{\lambda^\epsilon}$, for some constant $0 < \epsilon < 1$, we will say that f is sub-exponentially one-way.

Definition 2.2. (*Pseudorandom generator*) Let $\ell(\cdot)$ be a function. An efficiently computable function $\mathsf{PRG}\colon \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{2\ell(\lambda)}$ is a (t,μ)-$\mathsf{secure\ pseudorandom\ generator}$ if for every probabilistic algorithm $\mathcal{A}$ that runs in time $t = t(\lambda)$ it holds that

$$\mathsf{Adv}^{\mathsf{PRG}}_{f,\mathcal{A}} \stackrel{\mathsf{def}}{=} \left|\Pr_{x\leftarrow\{0,1\}^{\ell(\lambda)}}[\mathcal{A}(1^\lambda, \mathsf{PRG}(x)) = 1] - \Pr_{r\leftarrow\{0,1\}^{2\ell(\lambda)}}[\mathcal{A}(1^\lambda, r) = 1]\right| \leq \mu(\lambda)$$

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

Whenever $t = t(\lambda)$ is a super-polynomial function and $\mu = \mu(\lambda)$ is a negligible function, we will often omit t and μ and simply call the function a *pseudorandom generator*. In case $t(\lambda) = 1/\mu(\lambda) = 2^{\lambda^\epsilon}$, for some constant $0 < \epsilon < 1$, we will say that PRG is sub-exponentially secure.

2.2. *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 2.3. (*Pseudorandomness*) A function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ is (t, μ)-*secure pseudorandom* if for every probabilistic algorithm $\mathcal{A}$ that runs in time $t(\lambda)$, it holds that

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

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 [12,23,45,53]. 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 2.4. (*Puncturable PRF*) A (t, μ)-secure pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ is *puncturable* if there exists a probabilistic polynomial-time algorithm $\mathsf{PRF.Punc}$ such that 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 algorithm that runs in time at most $t(\lambda)$ 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 $\mathcal{A}$ it holds that

$$\mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathcal{A}_2(K_S, \mathsf{PRF.Eval}_K(x), \mathsf{state}) = 1\right] - \Pr\left[\mathcal{A}_2(K_S, y, \mathsf{state}) = 1\right]\right| \leq \mu(\lambda)$$

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 2.4 it holds that $S = \{x\}$ for some $x \in \mathcal{X}_\lambda$). As observed by [12,23,45,53], the GGM construction [35] of PRFs from any one-way function can be easily altered to yield such a puncturable pseudorandom function family.

2.3. *Private-Key Multi-input Functional Encryption*

In this section, we define the functionality and security of private-key *t-input* functional encryption. 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 $\mathsf{Enc}(\mathsf{msk}, m, \ell)$ takes as input a master secret key msk, a message m, and an index $\ell \in [t]$, where $m \in (\mathcal{X}_\ell)_\lambda$, and outputs a ciphertext ct_ℓ. 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 2.5. (*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 Setup, Enc and KG.

In terms of security, we rely on the private-key variant of the standard indistinguishability-based notion that considers both message privacy and function privacy [1,13,19]. Intuitively, we say that a t-input scheme is secure if for any two t-tuples of messages $(x_1^0, \ldots, x_t^0)$ and $(x_1^1, \ldots, x_t^1)$ that are encrypted with respect to indices $\ell = 1$ through $\ell = t$, and for every pair of functions (f_0, f_1), the triplets $(\mathsf{sk}_{f_0}, \mathsf{Enc}(\mathsf{msk}, x_1^0, 1), \ldots, \mathsf{Enc}(\mathsf{msk}, x_t^0, t))$ and $(\mathsf{sk}_{f_1}, \mathsf{Enc}(\mathsf{msk}, x_1^1, 1), \ldots, \mathsf{Enc}(\mathsf{msk}, x_t^1, t))$ are computationally indistinguishable as long as $f_0(x_1^0, \ldots, x_t^0) = f_1(x_1^1, \ldots, x_t^1)$ (note that this captures 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. We formalize our notions of security using left or right key-generation and encryption oracles. Specifically, for each $b \in \{0, 1\}$ and $\ell \in \{1, \ldots, t\}$ we let the left or right key-generation and encryption oracles be $\mathsf{KG}_b(\mathsf{msk}, f_0, f_1) \stackrel{\mathsf{def}}{=} \mathsf{KG}(\mathsf{msk}, f_b)$ and $\mathsf{Enc}_b(\mathsf{msk}, (m_0, m_1), \ell) \stackrel{\mathsf{def}}{=} \mathsf{Enc}(\mathsf{msk}, m_b, \ell)$. Before formalizing our notions of security, we define the notion of a *valid t-input adversary*. Then, we define *selective security*.

Definition 2.6. (*Valid adversary*) A probabilistic polynomial-time algorithm $\mathcal{A}$ is called *valid* 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 $((x_i^0, x_i^1), i) \in \mathcal{X}_i \times \mathcal{X}_i \times [t]$ 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 2.7. (*Selective security*) Let $t = t(\lambda), T = T(\lambda), Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda), Q_{\mathsf{enc}} = Q_{\mathsf{enc}}(\lambda)$ and $\mu = \mu(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. 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 $(T, Q_{\mathsf{key}}, Q_{\mathsf{enc}}, \mu)$-*selectively secure* if for any valid adversary $\mathcal{A}$ that on input 1^λ runs in time $T(\lambda)$ and issues at most $Q_{\mathsf{key}}(\lambda)$ key-generation queries and at most $Q_{\mathsf{enc}}(\lambda)$ encryption queries for each index $i \in [t]$, it holds that

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

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 $\vec{x}_i = ((x^0_{i,1}, x^1_{i,1}), \ldots, (x^0_{i,Q_{\mathsf{enc}}}, x^1_{i,Q_{\mathsf{enc}}}))$ 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 $j \in [Q_{\mathsf{enc}}]$.
4. $b' \leftarrow \mathcal{A}_2^{\mathsf{KG}_b(\mathsf{msk},\cdot,\cdot)}\left(1^\lambda, \{\mathsf{ct}_{i,j}\}_{i\in[t],j\in[T]}, \mathsf{state}\right)$.
5. If $b' = b$ then output 1, and otherwise output 0.

Known Constructions for $t = 1$ 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. [3] gave a generic transformation from selective 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 [3,19], given any selectively secure 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 [33,54], differing-input obfuscation [2,9], and multilinear maps [34]. In addition, schemes that are fully secure for a bounded number of key-generation queries Q_{key} can be based on the learning with errors (LWE) assumption (where the length of ciphertexts grows with Q_{key} and with a bound on the depth of allowed functions) [36], and even based on pseudorandom generators computable by small-depth circuits (where the length of ciphertexts grows with Q_{key} and with an upper bound on the circuit size of the functions) [39].

Known Constructions for $t > 1$ Private-key multi-input functional encryption schemes are much less understood than single-input ones. Goldwasser et al. [32] gave the first construction of a selectively secure multi-input functional encryption scheme for a polynomial number of inputs relying on indistinguishability obfuscation and one-way functions [11,33,43]. Following the work of Goldwasser et al., a *fully secure* private-key multi-input functional encryption scheme for a polynomial number of inputs based was constructed based on multilinear maps [14]. Later, Ananth, Jain, and Sahai, and Bitasnky and Vaikuntanathan [4,6,22] showed a selectively secure multi-input functional encryption scheme for a polynomial number of inputs based on any sub-exponentially secure single-input *public-key* functional encryption scheme. Brakerski et al. [13] showed that a *fully secure* single-input *private-key* scheme implies a *fully secure* multi-input scheme for any *constant* number of inputs. Furthermore, Brakerski et al. observed that their construction can be used to get a fully secure t-input scheme for $t = O(\log \log \lambda)$ inputs, where λ is the security parameter, if the underlying single-input scheme is sub-exponentially secure.

2.4. *Public-Key Functional Encryption*

In this section, we define the functionality and security of public-key (single-input) functional encryption. Let $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of finite sets, and let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be an ensemble of finite function families. For each $\lambda \in \mathbb{N}$, each function $f \in \mathcal{F}_\lambda$ takes as input a string, $x \in \mathcal{X}_\lambda$, and outputs a value $f(x) \in \mathcal{Z}_\lambda$.

A public-key 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 and a master public key mpk. The encryption algorithm $\mathsf{Enc}(\mathsf{mpk}, m)$ takes as input a master public key mpk and a message $m \in \mathcal{X}_\lambda$, and outputs a ciphertext ct. 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 2.8. (*Correctness*) A public-key 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 $x \in \mathcal{X}_\lambda$, it holds that

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

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

In terms of security, we rely on the public-key variant of the existing indistinguishability-based notions for message privacy.[7] Intuitively, we say that a scheme is secure if the

[7] We note that the notion of *function privacy* is very different from the one in the private-key setting, and in particular, natural definitions already imply obfuscation.

encryption of any pair of messages $\mathsf{Enc}(\mathsf{mpk}, m_0)$ and $\mathsf{Enc}(\mathsf{mpk}, m_1)$ cannot be distinguished as long as for any function f for which a functional key is queries, it holds that $f(m_0) = f(m_1)$. 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. Before formalizing our notions of security, we define the notion of a *valid adversary*. Then, we define *selective* security.[8]

Definition 2.9. (*Valid adversary*) A probabilistic polynomial-time algorithm $\mathcal{A}$ is called *valid* if for all public-key functional encryption schemes $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$, for all $\lambda \in \mathbb{N}$ and $b \in \{0, 1\}$, and for all $f \in \mathcal{F}_\lambda$ and $((x^0, x^1) \in (\mathcal{X})^2$ with which $\mathcal{A}$ queries the left or right encryption oracle, it holds that $f(x^0) = f(x^1)$.

Definition 2.10. (*Selective security*) Let $t = t(\lambda)$, $T = T(\lambda)$, $Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda)$ and $\mu = \mu(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. A public-key functional encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ over a message space $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a function space $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ is $(T, Q_{\mathsf{key}}, \mu)$-*selectively secure* if for any valid adversary $\mathcal{A}$ that on input 1^λ runs in time $T(\lambda)$ and issues at most $Q_{\mathsf{key}}(\lambda)$ key-generation queries, it holds that

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

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

1. $\left(x^0, x^1, \mathsf{state}\right) \leftarrow \mathcal{A}_1\left(1^\lambda\right)$.
2. $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda)$, $b \leftarrow \{0, 1\}$.
3. $b' \leftarrow \mathcal{A}_2^{\mathsf{KG}(\mathsf{msk},\cdot)}\left(1^\lambda, \mathsf{Enc}(\mathsf{mpk}, x^b), \mathsf{state}\right)$.
4. If $b' = b$ then output 1, and otherwise output 0.

2.5. *Indistinguishability Obfuscation*

We consider the standard notion of indistinguishability obfuscation [11,33]. We say that two circuits, C_0 and C_1, are *functionally equivalent*, and denote it by $C_0 \equiv C_1$, if for every x it holds that $C_0(x) = C_1(x)$.

Definition 2.11. (*Indistinguishability obfuscation*) Let $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a class of polynomial-size circuits operating on inputs of length n. An efficient algorithm $i\mathcal{O}$ is called a (t, μ)-indistinguishability obfuscator for the class $\mathcal{C}$ if it takes as input a security parameter λ and a circuit in $\mathcal{C}$ and outputs a new circuit so that following properties are satisfied:

1. **Functionality** For any input length $n \in \mathbb{N}$, any $\lambda \in \mathbb{N}$, and any $C \in \mathcal{C}_n$ it holds that

[8] We focus on selective security and do not define full security since there is a generic transformation [3].

$$\Pr\left[C \equiv i\mathcal{O}(1^\lambda, C)\right] = 1,$$

where the probability is taken over the internal randomness of $i\mathcal{O}$.

2. **Indistinguishability** For any probabilistic adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that runs in time $t = t(\lambda)$, it holds that

$$\mathsf{Adv}^{i\mathcal{O}}_{i\mathcal{O},\mathcal{C},\mathcal{A}} \stackrel{\text{def}}{=} \left|\Pr\left[\mathsf{Exp}^{i\mathcal{O}}_{i\mathcal{O},\mathcal{C},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mu(\lambda),$$

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

(a) $(C_0, C_1, \mathsf{state}) \leftarrow \mathcal{A}_1(1^\lambda)$ such that $C_0, C_1 \in \mathcal{C}$ and $C_0 \equiv C_1$.
(b) $\widehat{C} \leftarrow i\mathcal{O}(C_b)$, $b \leftarrow \{0, 1\}$.
(c) $b' \leftarrow \mathcal{A}_2\left(1^\lambda, \widehat{C}, \mathsf{state}\right)$.
(d) If $b' = b$ then output 1, and otherwise output 0.

3. Private-Key MIFE for a Poly-Logarithmic Number of Inputs

In this section, we present our construction of a private-key multi-input functional encryption scheme. The main technical tool underlying our approach is a transformation from a t-input scheme to a $2t$-input scheme which is described in Sect. 3.1. Then, in Sects. 3.2 and 3.3, we show that by iteratively applying the aforementioned transformation $O(\log\log\lambda)$ times, and by carefully controlling the security loss and the efficiency loss by adjusting the security parameter appropriately, we obtain a t-input scheme, where $t = (\log\lambda)^\delta$ for some constant $0 < \delta < 1$ (recall that $\lambda \in \mathbb{N}$ denotes the security parameter).

3.1. *From* t *Inputs to* $2t$ *Inputs*

Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of $2t$-input functionalities, where for every $\lambda \in \mathbb{N}$ the set $\mathcal{F}_\lambda$ consists of functions of the form $f : (\mathcal{X}_1)_\lambda \times \cdots \times (\mathcal{X}_{2t})_\lambda \to \mathcal{Z}_\lambda$. Our construction relies on the following building blocks:

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

For a $2t$-input function $f : (\mathcal{X}_1)_\lambda \times \cdots \times (\mathcal{X}_{2t})_\lambda \to \mathcal{Z}_\lambda$, denote by C_f the t-input function, where each input $i \in [t]$ is a pair of inputs that come from $(\mathcal{X}_i)_\lambda \times (\mathcal{X}_{t+i})_\lambda$, and the output is $\mathcal{Z}_\lambda$. The function is defined as:

$$C_f\left((x_1, x_{t+1}), \ldots, (x_t, x_{2t})\right) = f(x_1, \ldots, x_{2t}).$$

3.1.1. *Overview of Our Construction*

Our approach is to use the underlying t-input scheme to dynamically generate a master secret key *per* t inputs $x_1, \ldots, x_t$. Toward this goal, we associate with each x_ℓ, where $1 \leq$

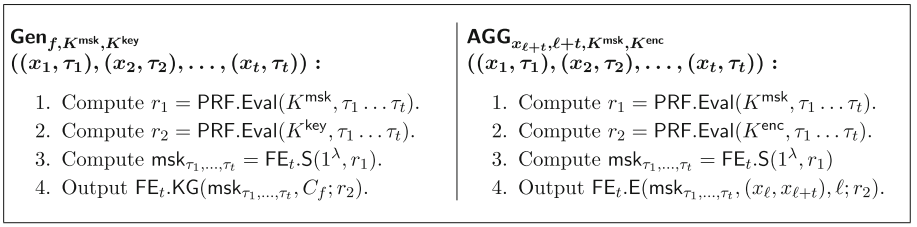

Fig. 3. The t-input functions $\mathsf{Gen}_{f,K^{\mathsf{msk}},K^{\mathsf{key}}}$ and $\mathsf{AGG}_{x_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}}}$.

$\ell \le t$, a random tag τ_ℓ and we are going to derive from $\tau_1\ldots\tau_t$ (using a PRF) randomness for this master secret key that we denote $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$. That is, using the ciphertext of $x_1,\ldots,x_t$, we will derive $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ and use it to generate a functional key for the t-input function C_f. We are left with generating ciphertexts of $(x_1,x_{t+1}),\ldots,(x_t,x_{2t})$ under the key $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$. To do that, upon an encryption request for $x_{\ell+t}$ for $1 \le \ell \le t$, we generate a key for a function that has $x_{\ell+t}$ hardwired, gets as input $(x_1,\tau_1),\ldots,(x_t,\tau_t)$, computes the dynamic master secret key $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ (using the same PRF key and input $\tau_1\ldots\tau_t$), and outputs an encryption of $(x_\ell,x_{\ell+t})$ under the new master secret key. Once we have the functional key and all ciphertexts under the dynamic master secret key, we can compute $C_f((x_1,x_{t+1}),\ldots,(x_t,x_{2t})) = f(x_1,\ldots,x_{2t})$, as needed.

We proceed with a slightly more precise description of the scheme that also explains how we derandomize the derivation of randomness for the generation of each dynamic master secret key using a PRF. The setup procedure outputs a PRF key K^{msk} and a key $\mathsf{msk}_{\mathsf{in}}$ for a t-input scheme FE_t. To generate a key for a function f, we sample a PRF key K^{key} and output $\mathsf{sk}_f \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\mathsf{Gen}_{f,K^{\mathsf{msk}},K^{\mathsf{key}}})$, where $\mathsf{Gen}_{f,K^{\mathsf{msk}},K^{\mathsf{key}}}$ is the t-input function that is defined in the left side of Fig. 3. To encrypt an input x relative to an index $\ell \in [2t]$, we have two cases depending on whether $1 \le \ell \le t$ or $t < \ell \le 2t$. In the former, we sample a random tag τ and encrypt (x,τ) using $\mathsf{msk}_{\mathsf{in}}$ relative to index ℓ. In the latter, we sample a PRF key K^{enc} and generate a functional key for the function $\mathsf{AGG}_{x,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}}}$ using $\mathsf{msk}_{\mathsf{in}}$, where $\mathsf{AGG}_{x,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}}}$ is the t-input function defined in the right side of Fig. 3. Finally, to decrypt a functional key sk_f and ciphertexts $\mathsf{ct}_1,\ldots,\mathsf{ct}_t,\mathsf{sk}_{t+1},\ldots,\mathsf{sk}_{2t}$, we computes $\mathsf{ct}'_i = \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_i,\mathsf{ct}_1,\ldots,\mathsf{ct}_t)$ for each $t < i < 2t$ and $\mathsf{sk}' = \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_f,\mathsf{ct}_1,\ldots,\mathsf{ct}_t)$, and finally output $\mathsf{FE}_t.\mathsf{D}(\mathsf{sk}',\mathsf{ct}'_{t+1},\ldots,\mathsf{ct}'_{2t})$.

This completes the full description of the scheme in terms of functionality; however, to carry out the security proof we need to make several modifications. Our proof is by a sequence of hybrids, where we "attack" each possible master secret key $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ separately, so that it will not be explicitly needed. In the proof, we maintain a counter c_ℓ per $\ell \in [t]$ and increase it by 1 every time a new ciphertext is issued with index ℓ. We logically think about all the encryption queries made with indices $1,\ldots,t$ as a $Q_{\mathsf{enc}} \times t$ matrix, where Q_{enc} bounds the number of ciphertexts generated per each coordinate. We denote the current sequence that we are attacking by $\mathsf{thr}_1,\ldots,\mathsf{thr}_t \in [Q_{\mathsf{enc}}]$ and we embed it into each ciphertext corresponding to index $\ell = 1$. That is, thr_ℓ is the index of the ciphertext we are attacking out of all the ciphertexts that were generated with $\ell \in [t]$. This splits all $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ to three sets: (1) ones that we already handled ("above"

$\mathsf{thr}_1, \ldots, \mathsf{thr}_t$), (2) the one that we are currently handling ("equal" to $\mathsf{thr}_1, \ldots, \mathsf{thr}_t$), and (3) the ones we are yet to handle ("below" $\mathsf{thr}_1, \ldots, \mathsf{thr}_t$).

We use a "double encryption" methodology to handle inputs that are "above" the threshold differently than how we handle inputs that are "below". This technique says that instead of encrypting an input once, we are going to encrypt it twice and use only one of them in an honest execution. In the proof, we use the slots interchangeably, depending on whether the given input is "above" or "below" the threshold. We use the same "double encryption" trick to directly argue about function privacy by hardwiring the function in two slots, and using the extra slot in the proof of security.

Once we marked and identified the $\tau_1, \ldots, \tau_t$ that we want to attack, we need to handle key-generation and encryption queries so that we can get "get rid" of $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$. To handle a key-generation query, we pre-compute the corresponding functional key under $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ and embed it into the same functional key—we call this value w. Handling encryption queries under $\mathsf{msk}_{x_1\ldots x_t}$ is done by embedding each ciphertext $\mathsf{ct}_{x_\ell \| x_{\ell+t}}$ in the ciphertext corresponding to $x_{\ell+t}$ (for each such $x_{\ell+t}$). After these changes $\mathsf{msk}_{\tau_1\ldots\tau_t}$ is not explicitly needed in the scheme and we can use the security of the underlying t-input scheme.

There are overall about Q_{enc}^t different options for $(\mathsf{thr}_1, \ldots, \mathsf{thr}_t)$ and for each of them the number of hybrids needed to get rid of $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ is constant. So, for t which is roughly logarithmic, our security loss is quasi-polynomial.

3.1.2. *The Construction*

Our scheme $\mathsf{FE}_{2t} = (\mathsf{FE}_{2t}.\mathsf{S}, \mathsf{FE}_{2t}.\mathsf{KG}, \mathsf{FE}_{2t}.\mathsf{E}, \mathsf{FE}_{2t}.\mathsf{D})$ is defined as follows.

- **The setup algorithm** On input the security parameter 1^λ the setup algorithm $\mathsf{FE}_{2t}.\mathsf{S}$ samples a master secret key for a t-input scheme $\mathsf{msk}_{\mathsf{in}} \leftarrow \mathsf{FE}_t.\mathsf{S}(1^\lambda)$, and a PRF key $K^{\mathsf{msk}} \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^\lambda)$, and outputs $\mathsf{msk} = (\mathsf{msk}_{\mathsf{in}}, K^{\mathsf{msk}})$.
- **The key-generation algorithm** On input the master secret key msk and a function $f \in \mathcal{F}_\lambda$, the key-generation algorithm $\mathsf{FE}_{2t}.\mathsf{KG}$ samples a PRF key $K^{\mathsf{key}} \leftarrow \mathsf{PRF}.\mathsf{Gen}(1^\lambda)$ and outputs $\mathsf{sk}_f \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f,\perp,K^{\mathsf{msk}},K^{\mathsf{key}},\perp})$, where $\mathsf{Gen}_{f,\perp,K^{\mathsf{msk}},K^{\mathsf{key}},\perp}$ is the t-input function that is defined in Fig. 4.

$\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$

$((x_1^0, x_1^1, \tau_1, \mathsf{c}_1, \mathsf{thr}_1, \ldots, \mathsf{thr}_t), (x_2^0, x_2^1, \tau_2, \mathsf{c}_2), \ldots, (x_t^0, x_t^1, \tau_t, \mathsf{c}_t))$:

1. For $i = 1, \ldots, t$ 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 $i = t$, then output w and HALT.
2. Compute $r_1 = \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{msk}}, \tau_1 \ldots \tau_t)$.
3. Compute $r_2 = \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{key}}, \tau_1 \ldots \tau_t)$.
4. Compute $\mathsf{msk}_{\tau_1,\ldots,\tau_t} = \mathsf{FE}_t.\mathsf{S}(1^\lambda, r_1)$.
5. Output $\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_1,\ldots,\tau_t}, C_f; r_2)$.

Fig. 4. The t-input function $\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$.

$\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$

$((x^0_1, x^1_1, \tau_1, \mathsf{c}_1, \mathsf{thr}_1, \ldots, \mathsf{thr}_t), (x^0_2, x^1_2, \tau_2, \mathsf{c}_2), \ldots, (x^0_t, x^1_t, \tau_t, \mathsf{c}_t))$:

1. For $i = 1, \ldots, t$ do:
   (a) If $\mathsf{c}_i < \mathsf{thr}_i$, then set $x_i = x^1_i$ for all $i \in [t]$ and exit loop.
   (b) If $\mathsf{c}_i > \mathsf{thr}_i$, then set $x_i = x^0_i$ 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 $i = t$, output v and HALT.
2. Compute $r_1 = \mathsf{PRF.Eval}(K^{\mathsf{msk}}, \tau_1 \ldots \tau_t)$.
3. Compute $r_2 = \mathsf{PRF.Eval}(K^{\mathsf{enc}}, \tau_1 \ldots \tau_t)$.
4. Compute $\mathsf{msk}_{\tau_1,\ldots,\tau_t} = \mathsf{FE}_t.\mathsf{S}(1^\lambda, r_1)$
5. Output $\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_1,\ldots,\tau_t}, (x_\ell, x_{\ell+t}), \ell; r_2)$.

Fig. 5. The t-input function $\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$.

- **The encryption algorithm** On input the master secret key msk, a message x and an index $\ell \in [2t]$, the encryption algorithm $\mathsf{FE}_{2t}.\mathsf{E}$ distinguished between the following three cases:

  – If $\ell = 1$, it samples a random string $\tau \in \{0,1\}^\lambda$, and then outputs ct_ℓ defined as follows.

  $$\mathsf{ct}_\ell \leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x, \bot, \tau, 1, \underbrace{1, \ldots, 1, 0}_{t \text{ slots}}), \ell).$$

  – If $1 < \ell \leq t$, it samples a random string $\tau \in \{0,1\}^\lambda$, and then outputs ct_ℓ defined as follows.

  $$\mathsf{ct}_\ell \leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x, \bot, \tau, 1), \ell).$$

  – If $t < \ell \leq 2t$, it samples a PRF key $K^{\mathsf{enc}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and outputs sk_ℓ defined as follows.

  $$\mathsf{sk}_\ell \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x,\bot,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}},\bot}),$$

  where $\mathsf{AGG}_{x,\bot,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}},\bot}$ is the t-input function that is defined in Fig. 5.

- **The decryption algorithm** On input a functional key sk_f and ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_t, \mathsf{sk}_{t+1}, \ldots, \mathsf{sk}_{2t}$, the decryption algorithm $\mathsf{FE}_t.\mathsf{D}$ computes

  $$\forall i \in \{t+1, \ldots, 2t\}: \mathsf{ct}'_i = \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_i, \mathsf{ct}_1, \ldots, \mathsf{ct}_t),$$
  $$\mathsf{sk}' = \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_t),$$

  and outputs $\mathsf{FE}_t.\mathsf{D}(\mathsf{sk}', \mathsf{ct}'_{t+1}, \ldots, \mathsf{ct}'_{2t})$.

Correctness For any $\lambda \in \mathbb{N}$, $f \in \mathcal{F}_\lambda$ and $(x_1, \ldots, x_{2t}) \in (\mathcal{X}_1)_\lambda \times \cdots \times (\mathcal{X}_{2t})_\lambda$, let sk_f denote a functional key for f and let $\mathsf{ct}_1, \ldots, \mathsf{ct}_t, \mathsf{sk}_{t+1}, \ldots, \mathsf{sk}_{2t}$ denote encryptions of $x_1, \ldots, x_{2t}$. Then, for every $i \in \{1, \ldots, t\}$, it holds that

$$\begin{aligned}\mathsf{ct}'_{i+t} &= \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_{i+t}, \mathsf{ct}_1, \ldots, \mathsf{ct}_t)\\ &= \mathsf{AGG}_{x_{i+t},\perp,i+t,K^{\mathsf{msk}},K^{\mathsf{enc}}_{i+t},\perp}((x_1, \perp, \tau_1, 1, 1, \ldots, 1, 0),\\ &\quad (x_2, \perp, \tau_2, 1), \ldots, (x_t, \perp, \tau_t, 1))\\ &= \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_1,\ldots,\tau_t}, (x_i, x_{i+t}), i; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{enc}}_{i+t}, \tau_1 \ldots \tau_t))\end{aligned}$$

and

$$\begin{aligned}\mathsf{sk}' &= \mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_f, \mathsf{ct}_1, \ldots, \mathsf{ct}_t)\\ &= \mathsf{Gen}_{f,\perp,K^{\mathsf{msk}},K^{\mathsf{key}}_f,\perp}((x_1, \perp, \tau_1, 1, 1, \ldots, 1, 0), (x_2, \perp, \tau_2, 1), \ldots, (x_t, \perp, \tau_t, 1))\\ &= \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_1,\ldots,\tau_t}, C_f; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{key}}_f, \tau_1 \ldots \tau_t))\end{aligned}$$

where $\mathsf{msk}_{\tau_1,\ldots,\tau_t} = \mathsf{FE}_t.\mathsf{S}(1^\lambda, \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{msk}}, \tau_1 \ldots \tau_t))$. Therefore,

$$\mathsf{FE}_t.\mathsf{D}(\mathsf{sk}', \mathsf{ct}'_{t+1}, \ldots, \mathsf{ct}'_{2t}) = C_f((x_1, x_{t+1}), \ldots, (x_t, x_{2t})) = f(x_1, \ldots, x_{2t}).$$

Security The following theorem captures the security our transformation. The proof of this theorem is given in Sect. 3.1.3.

Theorem 3.1. *Let $t = t(\lambda)$, $T = T(\lambda)$, $Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda)$, $Q_{\mathsf{enc}} = Q_{\mathsf{enc}}(\lambda)$ and $\mu = \mu(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$, and assume that FE_t is a $(T, Q_{\mathsf{key}}, Q_{\mathsf{enc}}, \mu)$-selectively secure t-input functional encryption scheme and that PRF is a (T, μ)-secure puncturable pseudorandom function family. Then, FE_{2t} is $(T', Q'_{\mathsf{key}}, Q'_{\mathsf{enc}}, \mu')$-selectively secure, where*

- $T'(\lambda) = T(\lambda) - Q_{\mathsf{key}}(\lambda) \cdot \mathsf{poly}(\lambda)$, *for some fixed polynomial* $\mathsf{poly}(\cdot)$.
- $Q'_{\mathsf{key}}(\lambda) = Q_{\mathsf{key}}(\lambda) - t(\lambda) \cdot Q_{\mathsf{enc}}(\lambda)$.
- $Q'_{\mathsf{enc}}(\lambda) = Q_{\mathsf{enc}}(\lambda)$.
- $\mu'(\lambda) = 8t(\lambda) \cdot (Q_{\mathsf{enc}}(\lambda))^{t(\lambda)+1} \cdot Q_{\mathsf{key}}(\lambda) \cdot \mu(\lambda)$.

3.1.3. *Proof of Theorem 3.1 (Proof of Security)*

Let $t = t(\lambda)$ and let $\mathcal{A}$ be a valid $2t$-input adversary that runs in time $T' = T'(\lambda)$, and issues at most $Q'_{\mathsf{enc}} = Q'_{\mathsf{enc}}(\lambda)$ encryption queries with respect to each index $i \in [2t]$, and at most $Q'_{\mathsf{key}} = Q'_{\mathsf{key}}(\lambda)$ key-generation queries. We present a sequence of experiments and upper bound $\mathcal{A}$'s advantage in distinguishing each two consecutive experiments. The first experiment is the experiment $\mathsf{Exp}^{\mathsf{selFE}_{2t}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}}(\lambda)$ (see Definition 2.7), and the last experiment is completely independent of the bit b. This enables us to prove that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{selFE}_{2t}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}}(\lambda) &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{selFE}_{2t}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|\\ &\leq 8t(\lambda) \cdot (Q'_{\mathsf{enc}}(\lambda))^{t(\lambda)+1} \cdot Q'_{\mathsf{key}}(\lambda) \cdot \mu(\lambda)\end{aligned}$$

for all sufficiently large $\lambda \in \mathbb{N}$. In what follows, we first describe the notation used throughout the proof, and then describe the experiments.

Notation We denote the ith ciphertext with respect to each index $1 \leq \ell \leq t$ by $\mathsf{ct}_{\ell,i}$ and the ith ciphertext with respect to each index $t < \ell \leq 2t$ by $\mathsf{sk}_{\ell,i}$. We denote the ith encryption query corresponding to each index $\ell \in [2t]$ by $(x^0_{\ell,i}, x^1_{\ell,i})$. For the ith ciphertext with respect to each index $1 \leq \ell \leq t$, we denote it associated random string by $\tau_{\ell,i}$. For the ith ciphertext with respect to each index $t+1 \leq \ell \leq 2t$, we denote its associated PRF key by $K^{\mathsf{enc}}_{\ell,i}$. Finally, we denote by (f^0_i, f^1_i) the ith the pair of functions with which the adversary queries the key-generation oracle and by K^{key}_i its associated PRF key.

For ease of following the proof, we give an outline of the sequence of experiments. By $\sim$ we denote computational indistinguishability and by $\equiv$ we denote equivalence. The core of the proof is a sequence of $\approx (Q'_{\mathsf{enc}})^t$ hybrids called $\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda)$ for every $k_1, \ldots, k_t \in [Q'_{\mathsf{enc}}] \cup \{0\}$. The values $k_1, \ldots, k_t$ represent the current $\mathsf{msk}_{\tau_1,\ldots,\tau_t}$ that we are attacking, by identifying τ_ℓ with the $k_\ell{}^{\text{th}}$ encryption done with index $\ell \in [t]$; see Sect. 3.1.1. These experiments satisfy the following invariant:

$$\mathcal{H}^{(2,k_1\ldots,k_{i-1},k_i,Q'_{\mathsf{enc}},0,\ldots,0)}(\lambda) \equiv \mathcal{H}^{(2,k_1\ldots,k_{i-1},k_i+1,0,\ldots,0)}(\lambda). \tag{3.1}$$

We show that

$$\mathcal{H}^{(2,k_1,\ldots,k_{t-1},k_t)}(\lambda) \sim \mathcal{H}^{(2,k_1,\ldots,k_{t-1},k_t+1)}(\lambda) \tag{3.2}$$

using the following sequence of hybrids

$$\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda) \sim \mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda) \sim \ldots \sim \mathcal{H}^{(7,k_1,\ldots,k_t)}(\lambda) \equiv \mathcal{H}^{(2,k_1,\ldots,k_{t-1},k_t+1)}(\lambda).$$

Using the sequences from (3.1) and (3.2) roughly $(Q'_{\mathsf{enc}})^t$ times, we get the final sequence of hybrids:

$$\mathcal{H}^{(0)}(\lambda) \sim \mathcal{H}^{(1)}(\lambda) \equiv \mathcal{H}^{(2,1,\ldots,1,0)}(\lambda) \sim \ldots \sim \mathcal{H}^{(2,Q'_{\mathsf{enc}},0,\ldots,0)}(\lambda) \sim \mathcal{H}^{(8)}(\lambda),$$

where $\mathcal{H}^{(0)}(\lambda)$ is the original experiment corresponding to $b \leftarrow \{0,1\}$ chosen uniformly at random and $\mathcal{H}^{(8)}(\lambda)$ is an experiment that is independent of b.

In the description of the experiments below, we use boxes to highlight the differences between an experiment and the previous one.

Experiment $\mathcal{H}^{(0)}(\lambda)$ This is the original experiment corresponding to $b \leftarrow \{0,1\}$ chosen uniformly at random, namely $\mathsf{Exp}^{\mathsf{selFE}_{2t}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}}(\lambda)$. Recall that in this experiment the ciphertexts and the functional keys are generated as follows.

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}, 2 \leq \ell \leq t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, \bot, \tau_{\ell,i}, 1, \underbrace{1, \ldots, 1, 0}_{t \text{ slots}}), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, \bot, \tau_{\ell,i}, 1), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, \bot, \ell+t, K^{\mathsf{msk}}, K^{\mathsf{enc}}_{\ell+t,i}, \bot}).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f^b_i, \bot, K^{\mathsf{msk}}, K^{\mathsf{key}}_i, \bot}).$$

Experiment $\mathcal{H}^{(1)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(0)}(\lambda)$ by modifying the ciphertexts as follows. Given inputs $(x^0_{\ell,i}, x^1_{\ell,i})$, instead of setting the field x_1 to be $\bot$ we set it to be $x^1_{\ell,i}$. In addition, we add a counter in every ciphertext. The scheme has the following form:

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}, 2 \leq \ell \leq t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, \boxed{x^1_{\ell,i}}, \tau_{\ell,i}, \boxed{i}, \underbrace{1, \ldots, 1, 0}_{t \text{ slots}}), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, \boxed{x^1_{\ell,i}}, \tau_{\ell,i}, \boxed{i}), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, \boxed{x^1_{\ell+t,i}}, \ell+t, K^{\mathsf{msk}}, K^{\mathsf{enc}}_{\ell+t,i}, \bot}).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f^b_i, \boxed{f^1_i}, K^{\mathsf{msk}}, K^{\mathsf{key}}_i, \bot}).$$

Note that all ciphertexts are generated so that $\mathsf{thr}_1, \ldots, \mathsf{thr}_{t-1} = 1$ and $\mathsf{thr}_t = 0$, while $\mathsf{c}_1, \ldots, \mathsf{c}_t \geq 1$ (after we added the counter). Thus, the circuit $\mathsf{AGG}_{x^b_{\ell+t}, x^1_{\ell+t}, \ell+t, K^{\mathsf{msk}}, K^{\mathsf{enc}}_{\ell+t}, \bot}$ always sets $x_i = x^b_i$ and ignores the second input x^1_i (see Fig. 5). Similarly, the circuit $\mathsf{Gen}_{f^0, f^1, K^{\mathsf{msk}}, K^{\mathsf{key}}, w}$ always sets $f_i = f^b_i$ and ignores the second input f^1_i (see Fig. 4). Thus, the security of the underlying scheme FE_t guarantees that the adversary $\mathcal{A}$ has only a negligible advantage in distinguishing experiments $\mathcal{H}^{(0)}$ and $\mathcal{H}^{(1)}$. Specifically, let $\mathcal{F}'$ denote the family of functions $\mathsf{AGG}_{x^0_{\ell+t}, x^1_{\ell+t}, \ell+t, K^{\mathsf{msk}}, K^{\mathsf{enc}}, v}$ (as defined in Fig. 5) and $\mathsf{Gen}_{f^0, f^1, K^{\mathsf{msk}}, K^{\mathsf{key}}, w}$ (as defined in Fig. 4). In Sect. 3.1.4, we prove the following claim:

Claim 3.2. *There exists a valid t-input adversary $\mathcal{B}^{(0)\to(1)}$ that runs in time $T'(\lambda) + (t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)) \cdot \mathsf{poly}(\lambda)$ and issues at most $Q'_{\mathsf{enc}}(\lambda)$ encryption queries*

with respect to each index $i \in [t]$ and at most $t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)$ key-generation queries, such that

$$\left|\Pr\left[\mathcal{H}^{(0)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(1)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(0)\to(1)}}(\lambda).$$

Experiment $\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(1)}(\lambda)$ by modifying the ciphertexts as follows. In each ciphertext corresponding to index $\ell = 1$ we embed the tuple $(k_1, \ldots, k_t)$ instead of the tuple $(1, \ldots, 1, 0)$.

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}$, $2 \leq \ell \leq t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i, \boxed{k_1, \ldots, k_t}), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, x^1_{\ell+t,i}, \ell+t, K^{\mathsf{msk}}, K^{\mathsf{enc}}_{\ell+t,i}, \perp}).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f^b_i, f^1_i, K^{\mathsf{msk}}, K^{\mathsf{key}}_i, \perp}).$$

Notice that $\mathcal{H}^{(2,1,\ldots,1,0)} = \mathcal{H}^{(1)}$.

Experiment $\mathcal{H}^{(3,k_1,k_2,\ldots,k_t)}(\lambda)$. This experiment is obtained from the experiment $\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda)$ by modifying the ciphertexts and the functional keys as follows. First, we compute $\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}$ as

$$\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{FE}_t.\mathsf{S}(1^\lambda; \mathsf{PRF.Eval}(K^{\mathsf{msk}}, \tau_{1,k_1} \ldots \tau_{t,k_t})).$$

Then, for every $\ell \in [t]$ and $i \in [T]$, we compute

$$\begin{aligned}
\gamma_{\ell+t,i} &= \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, (x^b_{\ell,k_\ell}, x^b_{\ell+t,i}), \ell; \mathsf{PRF.Eval}(K^{\mathsf{enc}}_{\ell+t,i}, \tau_{1,k_1} \ldots \tau_{t,k_t})),\\
\delta_i &= \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, C_{f^b_i}; \mathsf{PRF.Eval}(K^{\mathsf{key}}_i, \tau_{1,k_1} \ldots \tau_{t,k_t})).
\end{aligned}$$

Each $\gamma_{\ell+t,i}$ is embedded into $\mathsf{sk}_{\ell+t,i}$, and each δ_i is embedded into sk_{f_i}. Moreover, instead of using K^{msk}, K^{key}_i, and $K^{\mathsf{enc}}_{\ell+t,i}$, we use $K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$, $K^{\mathsf{key}}_i|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$, and $K^{\mathsf{enc}}_{\ell+t,i}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$, which are the keys K^{msk}, K^{key}, and K^{enc}_i all punctured at the same point $\{\tau_{1,k_1} \ldots \tau_{t,k_t}\}$. The scheme has the following form:

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}, 2 \le \ell \le t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i, k_1, \ldots, k_t), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, x^1_{\ell+t,i}, \ell+t, \boxed{K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}}, \boxed{K^{\mathsf{enc}}_{\ell+t,i}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}}, \boxed{\gamma_{\ell+t,i}}}),\\
&\quad \mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{FE}_t.\mathsf{S}(1^\lambda; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{msk}}, \tau_{1,k_1}\ldots\tau_{t,k_t})),\\
&\quad \gamma_{\ell+t,i} = \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, (x^b_{\ell,k_\ell}, x^b_{\ell+t,i}), \ell; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{enc}}_{\ell+t,i}, \tau_{1,k_1}\ldots\tau_{t,k_t})).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\begin{aligned}
\mathsf{sk}_{f_i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f^b_i, f^1_i, \boxed{K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}}, \boxed{K^{\mathsf{key}}_i|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}}, \boxed{\delta_i}}),\\
&\quad \delta_i = \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, C_{f^b_i}; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{key}}_i, \tau_{1,k_1}\ldots\tau_{t,k_t})).
\end{aligned}$$

Let $i^*_1, i^*_2 \ldots, i^*_{2t}$ be a combination of index of inputs and let i^* be an index of a function key. If $(\tau_{1,i^*_1}, \ldots, \tau_{t,i^*_t}) \neq (\tau_{1,k_1}, \ldots, \tau_{t,k_t})$, then by the definition of the circuits $\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ and $\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$, the functionalities do not change. Thus, let us assume that $(\tau_{1,i^*_1}, \ldots, \tau_{t,i^*_t}) = (\tau_{1,k_1}, \ldots, \tau_{t,k_t})$. In this case, by the definition of the experiment, the outputs of $\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ and each $\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$ are just δ_{i^*} and $\gamma_{\ell+t,i^*_{\ell+t}}$, respectively, which are defined exactly as the original outputs. So, in any case functionality is preserved. Thus, the security of the scheme FE_t guarantees that the adversary $\mathcal{A}$ has only a negligible advantage in distinguishing experiments $\mathcal{H}^{(2,k_1,\ldots,k_t)}$ and $\mathcal{H}^{(3,k_1,\ldots,k_t)}$. Specifically, let $\mathcal{F}'$ denote the family of functions $\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$ (as defined in Fig. 5) and $\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ (as defined in Fig. 4). In Sect. 3.1.4, we prove the following claim:

Claim 3.3. *There exists a valid t-input adversary $\mathcal{B}^{(2,k_1,\ldots,k_t)\to(3,k_1,\ldots,k_t)}$ that runs in time $T'(\lambda)+(t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot\mathsf{poly}(\lambda)$ and issues at most $Q'_{\mathsf{enc}}(\lambda)$ encryption queries with respect to each index $i \in [t]$ and at most $t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)$ key-generation queries, such that*

$$\begin{aligned}
&\left|\Pr\left[\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda)=1\right] - \Pr\left[\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda)=1\right]\right|\\
&\quad \le \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(2,k_1,\ldots,k_t)\to(3,k_1,\ldots,k_t)}}(\lambda).
\end{aligned}$$

Experiment $\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda)$ by modifying the ciphertexts and functional keys as follows. Instead of sampling the randomness for $\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}$, $\gamma_{\ell+t,i}$, and δ_i using a PRF, we sample them uniformly at random.

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}, 2 \le \ell \le t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i, k_1, \ldots, k_t), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, x^1_{\ell+t,i}, \ell+t, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K^{\mathsf{enc}}_{\ell+t,i}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \gamma_{\ell+t,i}}),\\
&\boxed{\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}} \leftarrow \mathsf{FE}_t.\mathsf{S}(1^\lambda),\\
&\boxed{\gamma_{\ell+t,i}} \leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, (x^b_{\ell,k_\ell}, x^b_{\ell+t,i}), \ell).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\begin{aligned}
\mathsf{sk}_{f_i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f^b_i, f^1_i, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K^{\mathsf{key}}_i|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \delta_i}),\\
&\boxed{\delta_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, C_{f^b_i}).
\end{aligned}$$

The pseudorandomness of the PRF keys K^{msk}, K^{key}_i, and $K^{\mathsf{enc}}_{\ell+t,i}$ at their respective punctured points enables us to bound $\mathcal{A}$'s advantage in distinguishing experiments $\mathcal{H}^{(3,k_1,\ldots,k_t)}$ and $\mathcal{H}^{(4,k_1,\ldots,k_t)}$. In total, there are $1 + t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)$ PRF keys, and in Sect. 3.1.4 we prove the following claim:

Claim 3.4. *There exists an algorithm $\mathcal{B}^{(3,k_1,\ldots,k_t)\to(4,k_1,\ldots,k_t)}$ that runs in time $T'(\lambda) + (t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)) \cdot \mathsf{poly}(\lambda)$ and issues $t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)$ queries such that*

$$\begin{aligned}
&\left|\Pr\left[\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda) = 1\right]\right|\\
&\qquad \le (1 + t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)) \cdot \mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF},\mathcal{B}^{(3,k_1,\ldots,k_t)\to(4,k_1,\ldots,k_t)}}(\lambda).
\end{aligned}$$

Experiment $\mathcal{H}^{(5,k_1,\ldots,k_t)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda)$ by modifying the ciphertexts and functional keys as follows. Instead of having $(x^b_{\ell,k_\ell}, \ldots, x^b_{\ell+t,i})$ encrypted in $\gamma_{\ell+t,i}$, we encrypt the value $(x^1_{\ell,k_\ell}, \ldots, x^1_{\ell+t,i})$. Similarly, instead of generating a key for $C_{f^b_i}$ in δ_i, we generate a key for $C_{f^1_i}$.

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}, 2 \le \ell \le t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i, k_1, \ldots, k_t), 1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i), \ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, x^1_{\ell+t,i}, \ell+t, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K^{\mathsf{enc}}_{\ell+t,i}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \gamma_{\ell+t,i}}),\\
&\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} \leftarrow \mathsf{FE}_t.\mathsf{S}(1^\lambda),\\
&\gamma_{\ell+t,i} \leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, \boxed{(x^1_{\ell,k_\ell}, x^1_{\ell+t,i})}, \ell).
\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\begin{aligned}\mathsf{sk}_{f_i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f_i^b, f_i^1, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K_i^{\mathsf{key}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \delta_i}),\\ \delta_i &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, \boxed{C_{f_i^1}}).\end{aligned}$$

We observe that this change only affects combinations of ciphertexts which contain $\mathsf{ct}_{1,k_1}, \ldots, \mathsf{ct}_{t,k_t}$. Every such combination with every $\gamma_{t+1,i_{t+1}}, \ldots, \gamma_{2t,i_{2t}}$ and every δ_j results with $f_j^1(x^1_{1,k_1}, \ldots, x^1_{t,k_t}, x^1_{t+1,i_{t+1}}, \ldots, x^1_{2t,k_{2t}})$ which must be equal to $f_j^b(x^b_{1,k_1}, \ldots, x^b_{t,k_t}, x^b_{t+1,i_{t+1}}, \ldots, x^b_{2t,k_{2t}})$ since the adversary is *valid* (see Definition 2.6). Thus, the security of the underlying FE_t scheme guarantees that the adversary $\mathcal{A}$ has only a negligible advantage in distinguishing experiments $\mathcal{H}^{(4,k_1,\ldots,k_t)}$ and $\mathcal{H}^{(5,k_1,\ldots,k_t)}$. Specifically, let $\mathcal{F}'$ denote the family of functions C_f as defined in Fig. 4. In Sect. 3.1.4, we prove the following claim:

Claim 3.5. *There exists a valid t-input adversary $\mathcal{B}^{(4,k_1,\ldots,k_t)\to(5,k_1,\ldots,k_t)}$ that runs in time $T'(\lambda) + (t(\lambda) \cdot Q'_{\mathsf{enc}}(\lambda) + Q'_{\mathsf{key}}(\lambda)) \cdot \mathsf{poly}(\lambda)$ and issues at most $Q'_{\mathsf{enc}}(\lambda)$ and $Q'_{\mathsf{key}}(\lambda)$ encryption and key-generation queries, respectively, such that*

$$\begin{aligned}&\left|\Pr\left[\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(5,k_1,\ldots,k_t)}(\lambda) = 1\right]\right|\\ &\quad\le \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(4,k_1,\ldots,k_t)\to(5,k_1,\ldots,k_t)}}(\lambda).\end{aligned}$$

Experiment $\mathcal{H}^{(6,k_1,\ldots,k_t)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(5,k_1,\ldots,k_t)}(\lambda)$ by modifying the ciphertexts and functional keys as follows. Instead of sampling the randomness for $\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}$, $\gamma_{\ell+t,i}$, and δ_i uniformly at random, we sample them using PRFs (as in hybrid $\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda)$).

- Ciphertexts ($i = 1, \ldots, Q'_{\mathsf{enc}}$, $2 \le \ell \le t$):

$$\begin{aligned}\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i, k_1, \ldots, k_t), 1),\\ \mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}}, (x^b_{\ell,i}, x^1_{\ell,i}, \tau_{\ell,i}, i), \ell),\\ \mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{AGG}_{x^b_{\ell+t,i}, x^1_{\ell+t,i}, \ell+t, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K^{\mathsf{enc}}_{\ell+t,i}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \gamma_{\ell+t,i}}),\\ &\boxed{\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}} = \mathsf{FE}_t.\mathsf{S}(1^\lambda; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{msk}}, \tau_{1,k_1} \ldots \tau_{t,k_t})),\\ &\boxed{\gamma_{\ell+t,i}} = \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, (x^1_{\ell,k_\ell}, x^1_{\ell+t,i}), \ell; \mathsf{PRF}.\mathsf{Eval}(K^{\mathsf{enc}}_{\ell+t,i}, \tau_{1,k_1} \ldots \tau_{t,k_t})).\end{aligned}$$

- Functional keys ($i = 1, \ldots, Q'_{\mathsf{key}}$):

$$\begin{aligned}\mathsf{sk}_{f_i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}}, \mathsf{Gen}_{f_i^b, f_i^1, K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, K_i^{\mathsf{key}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}, \delta_i}),\\ &\boxed{\delta_i} = \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, C_{f_i^1}; \mathsf{PRF}.\mathsf{Eval}(K_i^{\mathsf{key}}, \tau_{1,k_1} \ldots \tau_{t,k_t})).\end{aligned}$$

The pseudorandomness of the PRF keys K^{msk}, K_i^{key}, and $K_{\ell+t,i}^{\mathsf{enc}}$ at their respective punctured points enables us to bound $\mathcal{A}$'s advantage in distinguishing experiments $\mathcal{H}^{(5,k_1,\ldots,k_t)}$ and $\mathcal{H}^{(6,k_1,\ldots,k_t)}$. In total, there are $1+t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda)$ PRF keys, and the proof of the following claim is analogous to the proof of Claim 3.4.

Claim 3.6. *There exists an algorithm $\mathcal{B}^{(5,k_1,\ldots,k_t)\to(6,k_1,\ldots,k_t)}$ that runs in time $T'(\lambda)+(t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot\mathsf{poly}(\lambda)$ and issues $t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda)$ queries such that*

$$\left|\Pr\left[\mathcal{H}^{(5,k_1,\ldots,k_t)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(6,k_1,\ldots,k_t)}(\lambda)=1\right]\right| \le (1+t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot \mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF},\mathcal{B}^{(5,k_1,\ldots,k_t)\to(6,k_1,\ldots,k_t)}}(\lambda).$$

Experiment $\mathcal{H}^{(7,k_1,\ldots,k_t)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(6,k_1,\ldots,k_t)}(\lambda)$ by modifying the ciphertexts and functional keys as follows. The PRF keys $K^{\mathsf{msk}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$, $K_i^{\mathsf{key}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$, and $K_{\ell+t,i}^{\mathsf{enc}}|_{\{\tau_{1,k_1}\ldots\tau_{t,k_t}\}}$ are "unpunctured" (namely, we use the original keys rather than the punctured ones) and we replace each $\gamma_{\ell+t,i}$ and δ_i with $\perp$. Finally, we replace k_t in $\mathsf{ct}_{1,i}$ with k_t+1.

- Ciphertexts ($i=1,\ldots,Q'_{\mathsf{enc}}$, $2\le\ell\le t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},(x^b_{\ell,i},x^1_{\ell,i},\tau_{\ell,i},i,k_1,\ldots,k_t),1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},(x^b_{\ell,i},x^1_{\ell,i},\tau_{\ell,i},i),\ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\mathsf{AGG}_{x^b_{\ell+t,i},x^1_{\ell+t,i},\ell+t,\boxed{K^{\mathsf{msk}}},\boxed{K^{\mathsf{enc}}_{\ell+t,i}},\boxed{\perp}}).
\end{aligned}$$

- Functional keys ($i=1,\ldots,Q'_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\mathsf{Gen}_{f_i^b,f_i^1,\boxed{K^{\mathsf{msk}}},\boxed{K_i^{\mathsf{key}}},\boxed{\perp}}).$$

Observe that since thr_t in $\mathsf{ct}_{1,i}$ is now k_t+1, then for computing on the challenge (f_j^0,f_j^1) on inputs $(x^0_{1,k_1},x^1_{1,k_1}),\ldots,(x^0_{t,k_t},x^1_{t,k_t}),(x^0_{t+1,i_1},x^1_{t+1,i_1}),\ldots,(x^0_{2t,i_t},x^1_{2t,i_t})$, when combining the ciphertexts and key, we get a key for f_j^1 which is combined with the output of AGG. The latter is triggering the output $f_j^1(x^1_{1,k_1},\ldots,x^1_{t,k_t},x^1_{t+1,i_1},\ldots,x^1_{2t,i_t})$, as required. Thus, the security of the scheme FE_t guarantees that the adversary $\mathcal{A}$ has only a negligible advantage in distinguishing experiments $\mathcal{H}^{(6,k_1,\ldots,k_t)}$ and $\mathcal{H}^{(7,k_1,\ldots,k_t)}$. Specifically, let $\mathcal{F}'$ denote the family of functions $\mathsf{AGG}_{x^0,x^1,\tau,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$ and $\mathsf{Gen}_{f^0,f^1,,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ as defined in Figs. 5 and 4, respectively. The proof of the following claim is analogous to the proof of Claim 3.3.

Claim 3.7. *There exists a valid t-input adversary $\mathcal{B}^{(6,k_1,\ldots,k_t)\to(7,k_1,\ldots,k_t)}$ that runs in time $T'(\lambda)+(t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot\mathsf{poly}(\lambda)$ and issues at most $Q'_{\mathsf{enc}}(\lambda)$ encryption queries with respect to each index $i\in[t]$ and at most $t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda)$ key-generation queries, such that*

$$\left|\Pr\left[\mathcal{H}^{(6,k_1,\ldots,k_t)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(7,k_1,\ldots,k_t)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}^{(6,k_1,\ldots,k_t)\to(7,k_1,\ldots,k_t)}}(\lambda).$$

Next, we observe that $\mathcal{H}^{(7,k_1,\ldots,k_t)}(\lambda) = \mathcal{H}^{(2,k_1\ldots,k_{t-1},k_t+1)}(\lambda)$ and that $\mathcal{H}^{(2,k_1,\ldots,k_{t-1},Q'_{\mathsf{enc}})}(\lambda) = \mathcal{H}^{(2,k_1,\ldots,k_{t-1}+1,0)}(\lambda)$. More generally, it holds that $\mathcal{H}^{(2,k_1\ldots,k_i,Q'_{\mathsf{enc}},0,\ldots,0)}(\lambda) = \mathcal{H}^{(2,k_1\ldots,k_i+1,0,\ldots,0)}(\lambda)$.

Experiment $\mathcal{H}^{(8)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)}(\lambda)$ by modifying the ciphertexts and functional keys as follows. In sk_{f_i} we replace f_i^b with $\perp$. Moreover, in $\mathsf{sk}_{\ell+t,i}$ and all $\mathsf{ct}_{\ell,i}$ we replace x_i^b with $\perp$. Notice that this experiment is completely independent of the bit b, and therefore $\Pr[\mathcal{H}^{(8)}(\lambda)=1]=1/2$.

- Ciphertexts ($i=1,\ldots,Q'_{\mathsf{enc}}$, $2\leq\ell\leq t$):

$$\begin{aligned}
\mathsf{ct}_{1,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},(\boxed{\perp},x^1_{\ell,i},\tau_{\ell,i},i,T+1,0,\ldots,0),1),\\
\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},(\boxed{\perp},x^1_{\ell,i},\tau_{\ell,i},i),\ell),\\
\mathsf{sk}_{\ell+t,i} &\leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\mathsf{AGG}_{\boxed{\perp},x^1_{\ell+t,i},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}}_{\ell+t,i},\perp}).
\end{aligned}$$

- Functional keys ($i=1,\ldots,Q'_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\mathsf{Gen}_{\boxed{\perp},f_i^1,K^{\mathsf{msk}},K_i^{\mathsf{key}},\perp}).$$

We observe that since $\mathsf{thr}_1 = Q'_{\mathsf{enc}}+1$, it is always the case that the functions $\mathsf{AGG}_{x^0,x^1,\tau,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$ and $\mathsf{Gen}_{f^0,f^1,,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ use x^1 and f^1 as their inputs and ignore their first input. Thus, the security of the underlying schemes FE_t enables us to bound $\mathcal{A}$'s advantage in distinguishing between the experiments $\mathcal{H}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)}$ and $\mathcal{H}^{(8)}$. Specifically, let $\mathcal{F}'$ denote the family of functions $\mathsf{AGG}_{x^0_{\ell+t},x^1_{\ell+t},\ell+t,K^{\mathsf{msk}},K^{\mathsf{enc}},v}$ (as defined in Fig. 5) and $\mathsf{Gen}_{f^0,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},w}$ (as defined in Fig. 4). The proof of the following claim is similar to the proof of Claim 3.2.

Claim 3.8. *There exists a valid t-input adversary $\mathcal{B}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)\to(8)}$ that runs in time $T'(\lambda)+(t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot\mathsf{poly}(\lambda)$ and issues at most $Q'_{\mathsf{enc}}(\lambda)$ encryption queries with respect to each index $i\in[t]$ and at most $t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda)$ key-generation queries, such that*

$$\left|\Pr\left[\mathcal{H}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(8)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)\to(8)}}(\lambda).$$

Putting together Claims 3.2–3.8 with the assumptions that FE_t is a $(T,Q'_{\mathsf{key}},Q'_{\mathsf{enc}},\mu)$-selectively secure t-input functional encryption scheme and that PRF is a (T,μ)-secure puncturable pseudorandom function family, and with the facts that $\mathcal{H}^{(0)}(\lambda)=$

$\mathsf{Exp}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F},\mathcal{A}}(\lambda)$, $\mathcal{H}^{(1)}(\lambda) = \mathcal{H}^{(2,1,\ldots,1,0)}(\lambda)$, and $\Pr\left[\mathcal{H}^{(8)}(\lambda)=1\right] = 1/2$, we observe that

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{selFE_{2t}}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}} &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{selFE_{2t}}}_{\mathsf{FE}_{2t},\mathcal{F},\mathcal{A}}(\lambda)=1\right]-\frac{1}{2}\right| \\
&= \left|\Pr\left[\mathcal{H}^{(0)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(8)}(\lambda)=1\right]\right| \\
&\le \left|\Pr\left[\mathcal{H}^{(0)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(1)}(\lambda)=1\right]\right| \\
&\quad + \sum_{k_1=1}^{Q'_{\mathsf{enc}}}\sum_{k_2=0}^{Q'_{\mathsf{enc}}}\cdots\sum_{k_t=0}^{Q'_{\mathsf{enc}}}\sum_{i=2}^{6}\left|\Pr\left[\mathcal{H}^{(i,k_1,\ldots,k_t)}(\lambda)=1\right]\right. \\
&\quad \left.-\Pr\left[\mathcal{H}^{(i+1,k_1,\ldots,k_t)}(\lambda)=1\right]\right| \\
&\quad + \left|\Pr\left[\mathcal{H}^{(2,Q'_{\mathsf{enc}}+1,0,\ldots,0)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(8)}(\lambda)=1\right]\right| \\
&\le (2+(4+t\cdot Q'_{\mathsf{enc}}+Q'_{\mathsf{key}})\cdot(Q'_{\mathsf{enc}})^t)\cdot\mu \\
&\le 8t\cdot(Q'_{\mathsf{enc}})^{t+1}\cdot Q'_{\mathsf{key}}\cdot\mu. \qquad (3.3)
\end{aligned}
$$

3.1.4. *Proofs of Claims 3.2–3.5*

Proof of Claim 3.2. The adversary $\mathcal{B}^{(0)\to(1)} = \mathcal{B}$ given input 1^λ is defined as follows. First, $\mathcal{B}$ samples $K^{\mathsf{msk}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $b \leftarrow \{0,1\}$ and emulates the execution of $\mathcal{A}_1$ on input 1^λ by simulating the encryptions as follows: When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $\ell=1$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$, queries the encryption oracle $\mathsf{FE}_t.\mathsf{E}_\sigma(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot,\cdot)$ with the triple $((x^b,\perp,\tau,1,1,\ldots,1,0),(x^b,x^1,\tau,i,1,\ldots,1,0),1)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $1<\ell\le t$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$, queries the encryption oracle $\mathsf{FE}_t.\mathsf{E}_\sigma(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot,\cdot)$ with the triple $((x^b,\perp,\tau,1),(x^b,x^1,\tau,i),\ell)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $t<\ell\le 2t$, $\mathcal{B}$ samples $K^{\mathsf{enc}}\leftarrow\mathsf{PRF.Gen}(1^\lambda)$, queries the key-generation oracle $\mathsf{FE}_t.\mathsf{KG}_\sigma(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair $(\mathsf{AGG}_{x^b,\perp,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}},\perp},\mathsf{AGG}_{x^b,x^1,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}},\perp})$ and returns the output to $\mathcal{A}_1$. We do the above with all input triples until $\mathcal{A}_1$ outputs state and halts.

Then, we emulate the execution of $\mathcal{A}_2$ on input 1^λ, state and all the ciphertexts that were already generated by simulating the key-generation oracle as follows: When $\mathcal{A}_2$ requests a functional key for $(f^0,f^1)\in\mathcal{F}\times\mathcal{F}$, $\mathcal{B}$ samples a random $K^{\mathsf{key}}\leftarrow\mathsf{PRF.Gen}(1^\lambda)$, queries the key-generation oracle $\mathsf{FE}_t.\mathsf{KG}_\sigma(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair of circuits $(\mathsf{Gen}_{f^b,\perp,K^{\mathsf{msk}},K^{\mathsf{key}},\perp},\mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},\perp})$ and returns the output to $\mathcal{A}_2$. We do the above until $\mathcal{A}_2$ outputs b' and halts. Finally, $\mathcal{B}$ outputs 1 if $b'=b$ and otherwise it outputs 0. The adversary $\mathcal{B}$ is *valid* since by definition $\mathsf{Gen}_{f^b,\perp,K^{\mathsf{msk}},K^{\mathsf{key}},\perp}$ on input of the form $((x_1^b,\perp,\tau_1,1,1,\ldots,1,0),(x_2^b,\perp,\tau_2,1),\ldots,(x_t^b,\perp,\tau_t,1))$ is always equal to $\mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},\perp}$ on any input of the form $((x_1^b,x_1^1,\tau_1,i_1,1,\ldots,1,0),(x_2^b,x_2^1,\tau_2,i_2),\ldots,(x_t^b,x_t^1,\tau_t,i_t))$.

Note that when $\sigma = 0$ then $\mathcal{A}$'s view is identical to its view in the experiment $\mathcal{H}^{(0)}$, and when $\sigma = 1$ then $\mathcal{A}$'s view is identical to its view in the modified experiment $\mathcal{H}^{(1)}$ described above. Therefore,

$$\left|\Pr\left[\mathcal{H}^{(0)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(1)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(0)\to(1)}}(\lambda).$$

□

Proof of Claim 3.3. The adversary $\mathcal{B}^{(2,k_1,\ldots,k_t)\to(3,k_1,\ldots,k_t)} = \mathcal{B}$ given input 1^λ is defined as follows. First, $\mathcal{B}$ samples $K^{\mathsf{msk}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, $\tau_{1,k_1}, \ldots, \tau_{t,k_t} \leftarrow \{0,1\}^\lambda$, $b \leftarrow \{0,1\}$, computed the punctured PRF key $K^{\mathsf{msk}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{PRF.Punc}(K^{\mathsf{msk}}, \tau_{1,k_1}, \ldots, \tau_{t,k_t})$, and emulates the execution of $\mathcal{A}_1$ on input 1^λ by simulating the encryptions as follows: When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0, x^1) \in \mathcal{X}_\lambda$ with respect to index $\ell = 1$, $\mathcal{B}$ samples $\tau \in \{0,1\}^\lambda$ (if $i = k_1$, we use τ_{1,k_1} that was sampled in the beginning), queries the encryption oracle $\mathsf{FE}_t.\mathsf{E}_\sigma(\mathsf{msk}_{\mathsf{in}}, \cdot, \cdot, \cdot)$ with the triple $((x^b, x^1, \tau, i, k_1, \ldots, k_t), (x^b, x^1, \tau, i, k_1, \ldots, k_t), 1)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0, x^1) \in \mathcal{X}_\lambda$ with respect to index $1 < \ell \leq t$, $\mathcal{B}$ samples $\tau \in \{0,1\}^\lambda$ (if $i = k_\ell$, we use τ_{ℓ,k_ℓ} that was sampled in the beginning), queries the encryption oracle $\mathsf{FE}_t.\mathsf{E}_\sigma(\mathsf{msk}_{\mathsf{in}}, \cdot, \cdot, \cdot)$ with the triple $((x^b, x^1, \tau, i), (x^b, x^1, \tau, i), \ell)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0, x^1) \in \mathcal{X}_\lambda$ with respect to index $t \leq \ell + t \leq 2t$, $\mathcal{B}$ samples $K^{\mathsf{enc}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, computes a punctured key $K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{PRF.Gen}(K^{\mathsf{enc}}, \tau_{1,k_1}, \ldots, \tau_{t,k_t})$, queries the key-generation oracle $\mathsf{FE}_t.\mathsf{KG}_\sigma(\mathsf{msk}_{\mathsf{in}}, \cdot, \cdot)$ with the pair $(\mathsf{AGG}_{x^b,x^1,\ell,K^{\mathsf{msk}},K^{\mathsf{enc}},\perp}, \mathsf{AGG}_{x^b,x^1,\ell,K^{\mathsf{msk}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},\gamma_{\ell,t}})$, where $\gamma = \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}, (x^b_{\ell,k_\ell}, x^b); \mathsf{PRF.Eval}(K^{\mathsf{enc}}, \tau_{1,k_1} \ldots \tau_{t,k_t}))$, and returns the output to $\mathcal{A}_1$. We do the above with all input triples until $\mathcal{A}_1$ outputs state and halts.

Then, we emulate the execution of $\mathcal{A}_2$ on input 1^λ, state and all the ciphertexts that were already generated by simulating the key-generation oracle as follows: When $\mathcal{A}_2$ requests a functional key for $(f^0, f^1) \in \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ samples a random $K^{\mathsf{key}} \leftarrow \mathsf{PRF.Gen}(1^\lambda)$, computes $K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}} = \mathsf{PRF.Punc}(K^{\mathsf{key}}, \tau_{1,k_1} \ldots \tau_{t,k_t})$, queries the key-generation oracle $\mathsf{FE}_t.\mathsf{KG}_\sigma(\mathsf{msk}_{\mathsf{in}}, \cdot, \cdot)$ with the pair of circuits $(\mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}},K^{\mathsf{key}},\perp}, \mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},\delta})$, where $\delta = \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1}\ldots\tau_{t,k_t}}, C_{f^b}; \mathsf{PRF.Eval}(K^{\mathsf{key}}, \tau_{1,k_1} \ldots \tau_{t,k_t}))$, and returns the output to $\mathcal{A}_2$. We do the above until $\mathcal{A}_2$ outputs b' and halts. Finally, $\mathcal{B}$ outputs 1 if $b' = b$ and otherwise it outputs 0.

Note that when $\sigma = 0$ then $\mathcal{A}$'s view is identical to its view in the experiment $\mathcal{H}^{(2,k_1,\ldots,k_t)}$, and when $\sigma = 1$ then $\mathcal{A}$'s view is identical to its view in the modified experiment $\mathcal{H}^{(3,k_1,\ldots,k_t)}$ described above. Therefore,

$$\left|\Pr\left[\mathcal{H}^{(2,k_1,\ldots,k_t)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(2,k_1,\ldots,k_t)\to(3,k_1,\ldots,k_t)}}(\lambda).$$

□

Proof of Claim 3.4. The proof of this claim proceeds by $(1+t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))$ hybrid experiments, where in each we replace only one PRF evaluation with sampling a string uniformly at random. Since all indistinguishability proofs of the experiments are very similar, we provide the proof for one and omit the missing details. In what follows, we prove that the experiment $\mathcal{H}^{(3,k_1,\ldots,k_t)}$ is indistinguishable from an experiment $\mathcal{H}'^{(3,k_1,\ldots,k_t)}$ where the randomness for $\mathsf{msk}_{\tau_{1,k_1}\ldots\tau_{1,k_1}}$ is chosen uniformly at random (rather than using K^{msk}).

The adversary $\mathcal{B}$ given input 1^λ is defined as follows. First, $\mathcal{B}$ samples $\mathsf{msk}_{\mathsf{in}} \leftarrow \mathsf{FE}_t.\mathsf{S}(1^\lambda)$, $\tau_{1,k_1},\ldots,\tau_{t,k_t} \leftarrow \{0,1\}^\lambda$ and $b \leftarrow \{0,1\}$. Now, $\mathcal{A}$ is given $R(\tau_{1,k_1}\ldots\tau_{t,k_t})$ and a punctured PRF key $K^{\mathsf{msk}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}}$ and its goal is to guess if $R(\tau_{1,k_1}\ldots\tau_{t,k_t})$ is uniformly random or the output of a PRF. First, $\mathcal{B}$ computes $\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{FE}_t.\mathsf{S}(1^\lambda; R(\tau_{1,k_1}\ldots\tau_{t,k_t}))$. Then, $\mathcal{B}$ emulates the execution of $\mathcal{A}$ on input 1^λ by simulating the encryption oracle as follows: When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $\ell=1$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$ (if $i=k_1$, we use τ_{1,k_1} that was sampled in the beginning), computes $\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair $((x^b,x^1,\tau,i,k_1,\ldots,k_t),1)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $1<\ell\le t$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$ (if $i=k_\ell$, we use τ_{ℓ,k_ℓ} that was sampled in the beginning), computes the encryption $\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair $((x^b,x^1,\tau,i),\ell)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $t\le\ell+t\le 2t$, $\mathcal{B}$ samples $K^{\mathsf{enc}}\leftarrow\mathsf{PRF.Gen}(1^\lambda)$, computes a punctured key $K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}} = \mathsf{PRF.Gen}(K^{\mathsf{enc}},\tau_{1,k_1},\ldots,\tau_{t,k_t})$, computes the functional key $\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\cdot)$ with the function $\mathsf{AGG}_{x^b,x^1,\ell,K^{\mathsf{msk}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},\gamma_{\ell,t}}$, where $\gamma=\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},(x^b_{\ell,k_\ell},x^b);\mathsf{PRF.Eval}(K^{\mathsf{enc}},\tau_{1,k_1}\ldots\tau_{t,k_t}))$, and returns the output to $\mathcal{A}_1$. We do the above with all input triples until $\mathcal{A}_1$ outputs state and halts.

Then, we emulate the execution of $\mathcal{A}_2$ on input 1^λ, state and all the ciphertexts that were already generated by simulating the key-generation oracle as follows: When $\mathcal{A}_2$ submits a key-generation query $(f^0,f^1)\in\mathcal{F}\times\mathcal{F}$, $\mathcal{B}$ samples a $K^{\mathsf{key}}\leftarrow\mathsf{PRF.Gen}(1^\lambda)$, computes $K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}} = \mathsf{PRF.Punc}(K^{\mathsf{key}},\tau_{1,k_1}\ldots\tau_{t,k_t})$, computes a functional for $\mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},\delta}$ using the algorithm $\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\cdot)$, where $\delta=\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\tau_{1,k_1}\ldots\tau_{t,k_t}},C_{f^b};\mathsf{PRF.Eval}(K^{\mathsf{key}},\tau_{1,k_1}\ldots\tau_{t,k_t}))$, and returns the output to $\mathcal{A}_2$. We do the above until $\mathcal{A}_2$ outputs b' and halts. Finally, $\mathcal{B}$ outputs 1 if $b'=b$ and otherwise it outputs 0.

Note that when the value $R(\tau_{1,k_1}\ldots\tau_{t,k_t})$ is sampled using K^{msk}, then $\mathcal{A}$'s view is identical to its view in the experiment $\mathcal{H}^{(3,k_1,\ldots,k_t)}$, and when $R(\tau_{1,k_1}\ldots\tau_{t,k_t})$ is sampled uniformly at random, then $\mathcal{A}$'s view is identical to its view in the modified experiment $\mathcal{H}'^{(3,k_1,\ldots,k_t)}$ described above. Therefore,

$$\left|\Pr\left[\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda)=1\right]-\Pr\left[\mathcal{H}'^{(3,k_1,\ldots,k_t)}(\lambda)=1\right]\right|\le\mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF},\mathcal{B}}(\lambda).$$

An analogous argument shows that every two consecutive experiments are indistinguishable (with the same advantage). Thus,

$$\left|\Pr\left[\mathcal{H}^{(3,k_1,\ldots,k_t)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda)=1\right]\right| \leq (1+t(\lambda)\cdot Q'_{\mathsf{enc}}(\lambda)+Q'_{\mathsf{key}}(\lambda))\cdot \mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF},\mathcal{B}^{(3,k_1,\ldots,k_t)\to(4,k_1,\ldots,k_t)}}(\lambda).$$

□

Proof of Claim 3.5. The adversary $\mathcal{B}^{(4,k_1,\ldots,k_t)\to(5,k_1,\ldots,k_t)}=\mathcal{B}$ given input 1^λ is defined as follows. First, $\mathcal{B}$ samples $\mathsf{msk}_{\mathsf{in}}\leftarrow\mathsf{FE}_t.\mathsf{S}(1^\lambda)$, $K^{\mathsf{msk}}\leftarrow\mathsf{PRF}.\mathsf{Gen}(1^\lambda)$, $\tau_{1,k_1},\ldots,\tau_{t,k_t}\leftarrow\{0,1\}^\lambda, b\leftarrow\{0,1\}$, computed the punctured PRF key $K^{\mathsf{msk}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}=\mathsf{PRF}.\mathsf{Punc}(K^{\mathsf{msk}},\tau_{1,k_1},\ldots,\tau_{t,k_t})$, and emulates the execution of $\mathcal{A}_1$ on input 1^λ by simulating the encryptions as follows: When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $\ell=1$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$ (if $i=k_1$, we use τ_{1,k_1} that was sampled in the beginning), computes the encryption $\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair $((x^b,x^1,\tau,i,k_1,\ldots,k_t),1)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $1<\ell\leq t$, $\mathcal{B}$ samples $\tau\in\{0,1\}^\lambda$ (if $i=k_\ell$, we use τ_{ℓ,k_ℓ} that was sampled in the beginning), computes the encryption $\mathsf{FE}_t.\mathsf{E}(\mathsf{msk}_{\mathsf{in}},\cdot,\cdot)$ with the pair $((x^b,x^1,\tau,i),\ell)$ and returns the output to $\mathcal{A}_1$. When $\mathcal{A}_1$ requests the ith encryption of the pair $(x^0,x^1)\in\mathcal{X}_\lambda$ with respect to index $t\leq\ell+t\leq 2t$, $\mathcal{B}$ samples $K^{\mathsf{enc}}\leftarrow\mathsf{PRF}.\mathsf{Gen}(1^\lambda)$, computes a punctured key $K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}}=\mathsf{PRF}.\mathsf{Gen}(K^{\mathsf{enc}},\tau_{1,k_1},\ldots,\tau_{t,k_t})$, computes the functional key $\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\cdot)$ with the function $\mathsf{AGG}_{x^b,x^1,\ell,K^{\mathsf{msk}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},K^{\mathsf{enc}}|_{\tau_{1,k_1},\ldots,\tau_{t,k_t}},\gamma_{\ell,t}}$, where γ is the output of the output of the encryption oracle $\mathsf{FE}_t.\mathsf{E}_\sigma(\mathsf{msk}_{\tau_{1,k_1}\ldots\tau_{t,k_t}},\cdot,\cdot,\cdot)$ on input the triple $((x^b_{\ell,k_\ell},x^b),(x^1_{\ell,k_\ell},x^1),\ell)$, and returns the output to $\mathcal{A}_1$. We do the above with all input triples until $\mathcal{A}_1$ outputs state and halts.

Then, we emulate the execution of $\mathcal{A}_2$ on input 1^λ, state and all the ciphertexts that were already generated by simulating the key-generation oracle as follows: When $\mathcal{A}_2$ submits a key-generation query $(f^0,f^1)\in\mathcal{F}\times\mathcal{F}$, $\mathcal{B}$ samples a random $K^{\mathsf{key}}\leftarrow\mathsf{PRF}.\mathsf{Gen}(1^\lambda)$, computes $K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}}=\mathsf{PRF}.\mathsf{Punc}(K^{\mathsf{key}},\tau_{1,k_1}\ldots\tau_{t,k_t})$, computes a functional key using $\mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}_{\mathsf{in}},\cdot)$ for the t-input circuit $\mathsf{Gen}_{f^b,f^1,K^{\mathsf{msk}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},K^{\mathsf{key}}|_{\tau_{1,k_1}\ldots\tau_{t,k_t}},\delta}$), where δ is the value returned by the key-generation oracle $\mathsf{FE}_t.\mathsf{KG}_\sigma(\mathsf{msk}_{\tau_{1,k_1}\ldots\tau_{t,k_t}},\cdot,\cdot)$ on input the pair of functions (C_{f^b},C_{f^1}), and returns the output to $\mathcal{A}_2$. We do the above until $\mathcal{A}_2$ outputs b' and halts. Finally, $\mathcal{B}$ outputs 1 if $b'=b$ and otherwise it outputs 0.

Note that when $\sigma=0$ then $\mathcal{A}$'s view is identical to its view in the experiment $\mathcal{H}^{(4,k_1,\ldots,k_t)}$, and when $\sigma=1$ then $\mathcal{A}$'s view is identical to its view in the modified experiment $\mathcal{H}^{(5,k_1,\ldots,k_t)}$ described above. Therefore,

$$\left|\Pr\left[\mathcal{H}^{(4,k_1,\ldots,k_t)}(\lambda)=1\right]-\Pr\left[\mathcal{H}^{(5,k_1,\ldots,k_t)}(\lambda)=1\right]\right| \leq \mathsf{Adv}^{\mathsf{selFE_t}}_{\mathsf{FE}_t,\mathcal{F}',\mathcal{B}'^{(4,k_1,\ldots,k_t)\to(5,k_1,\ldots,k_t)}}(\lambda).$$

□

3.2. *Efficiency Analysis*

In this section, we analyze the overhead incurred by our transformation by analyzing the running time of each procedure given an instantiation of a given single-input scheme. Specifically, for a message space $\mathcal{X}_1 \times \cdots \times \mathcal{X}_{2t}$ and a function space $\mathcal{F}$ that consists of $2t$-input functions, we instantiate our scheme (by applying our transformation $\log t$ times) and analyze the size of a master secret key, the size of a functional key, the size of a ciphertext and the time it takes to evaluate a functional key with $2t$ ciphertexts. We assume that the initial single-input scheme FE_1 is efficient in the following sense. Setup with security parameter 1^λ runs in time $\mathsf{poly}(\lambda)$, key generation for a function of size s takes time $\mathsf{poly}(s, \lambda)$, encryption of a message of size m takes time $\mathsf{poly}(m, \lambda)$, and decryption also runs in fixed polynomial time.

Let $\lambda \in \mathbb{N}$ be a security parameter with which we instantiate the $2t$-input scheme, let us assume that $\mathcal{F}$ consists of functions of size at most $s = s(\lambda)$ and that each $\mathcal{X}_i$ consists of messages of size at most $m = m(\lambda)$. Assuming that $\log t \leq \mathsf{poly}(\lambda)$ (to simplify notation), we show that there exists a fixed constant $\mathsf{c} \in \mathbb{N}$ such that:

- the setup procedure takes time λ^{c},
- the key-generation procedure takes time $(s \cdot \lambda)^{t^{\log \mathsf{c}}}$,
- the encryption procedure takes time $(m \cdot \lambda)^{t^{\log \mathsf{c}}}$, and
- the decryption procedure takes time $t^{\log t} \cdot \lambda^{\mathsf{c}}$.

For a circuit A that receives inputs of lengths $x_1 \ldots, x_m$, we denote by $\mathsf{Time}(A, x_1, \ldots, x_m)$ the size of the circuit when applied to inputs of length $\sum_{i=1}^{m} x_i$. For a function family $\mathcal{F}$, we denote by $\mathsf{Size}(\mathcal{F})$ the maximal size of the circuit that implements a function from $\mathcal{F}$.

The Setup Procedure The setup procedure of FE_{2t} is composed of sampling a key for a scheme FE_t and generating a PRF key. Iterating this, we see that a master secret key in our final scheme consists of a single master secret key for a single-input scheme and $\log t$ additional PRF keys. Namely,

$$\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{S}, 1^\lambda) = \mathsf{Time}(\mathsf{FE}_t.\mathsf{S}, 1^\lambda) + p_1(\lambda),$$

where p_1 is a fixed polynomial that depends on the key-generation time of the PRF, and thus

$$\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda) = \mathsf{Time}(\mathsf{FE}_1.\mathsf{S}, \lambda) + \log t \cdot p_1(\lambda).$$

The Key-Generation Procedure The key-generation procedure of FE_{2t} depends on the complexity of the key-generation procedure of the FE_t scheme. Let $\mathcal{F}^{2t}$ be the function family that is supported by the scheme FE_{2t}.

$$\begin{aligned}&\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{KG}, \lambda, \mathsf{Size}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda), \mathsf{Size}(\mathcal{F}^{2t})) = \\ &\quad \mathsf{Time}(\mathsf{FE}_t.\mathsf{KG}, \lambda, 2\mathsf{Size}(\mathcal{F}^{2t}), \\ &\quad \mathsf{Time}(\mathsf{FE}_t.\mathsf{S}, \lambda), \mathsf{Time}(\mathsf{FE}_t.\mathsf{KG}, \mathsf{Size}(\mathcal{F}^{2t})), p_2(\lambda))) + p_3(\lambda),\end{aligned}$$

where p_2 subsumes the size of the embedded PRF keys and the complexity of the simple operations that are done in Gen, and p_3 subsumes the running time of the generation of the PRF key K^{key}.

The dominant part in the above equation is that the time it takes to generate a key with respect to FE_{2t} for a function whose size is $\mathsf{Size}(\mathcal{F}^{2t})$ depends on the circuit size of key-generation in the scheme FE_t for a function whose size is $\mathsf{Time}(\mathsf{FE}_t.\mathsf{KG}, \mathsf{Size}(\mathcal{F}^{2t}))$ (namely, it is a function that outputs a functional key for a function whose size is $\mathsf{Size}(\mathcal{F}^{2t})$). Thus, applying this equation recursively, we get that for large enough $c \in \mathbb{N}$ (that depends on the exponents of p_2 and p_3), it holds that

$$\begin{aligned}&\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{KG}, \lambda, \mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda), \mathsf{Size}(\mathcal{F}^{2t})) \leq (\mathsf{Size}(\mathcal{F}^{2t}) \cdot \lambda)^{c^{\log t}} \\ &= (\mathsf{Size}(\mathcal{F}^{2t}) \cdot \lambda)^{t^{\log c}}.\end{aligned}$$

The Encryption Procedure The encryption procedure of FE_{2t} depends on the complexity of encryption and key-generation of the FE_t scheme. Let m be the length of a message to encrypt. For $\ell \leq t$, the complexity is at most

$$\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{E}, \lambda, \mathsf{Size}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda), m) \leq \mathsf{Time}(\mathsf{FE}_t.\mathsf{E}, \lambda, 2m, (t+2)\lambda).$$

For $t+1 \leq \ell \leq 2t$, the complexity of encryption is

$$\begin{aligned}&\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{E}, \lambda, \mathsf{Size}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda), m) \leq \\ &\quad \mathsf{Time}(\mathsf{FE}_t.\mathsf{KG}, \lambda, \mathsf{Time}(\mathsf{FE}_t.\mathsf{S}, \lambda), \mathsf{Time}(\mathsf{FE}_t.\mathsf{E}, 2m), p_4(\lambda)),\end{aligned}$$

where p_4 subsumes the running time of the key-generation procedure of the PRF and the various other simple operations made by AGG.

The dominant part is that an encryption of a message with respect to the scheme FE_{2t} requires generating a key with respect to the scheme FE_t for a function whose size is $\mathsf{Time}(\mathsf{FE}_t.\mathsf{E}, 2m)$. Thus, similarly to the analysis of the key-generation procedure, we get that for some fixed $c \in \mathbb{N}$ (that depends on the exponents of p_4 and the time it takes to encrypt a message with respect to FE_1), we get that

$$\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{E}, \lambda, \mathsf{Size}(\mathsf{FE}_{2t}.\mathsf{S}, \lambda), m) \leq (m \cdot \lambda)^{t^{\log c}}.$$

The Decryption Procedure Decryption in the scheme FE_{2t} requires $t+2$ decryption operations with respect to the scheme FE_t. Let $\mathsf{ct}(t)$ and $\mathsf{sk}(t)$ be the length of a ciphertext and a key in the scheme FE_t, respectively. We get that

$$\begin{aligned}\mathsf{Time}(\mathsf{FE}_{2t}.\mathsf{D}, \mathsf{sk}(t), 2t \cdot \mathsf{ct}(t)) &= (t+2) \cdot \mathsf{Time}(\mathsf{FE}_t.\mathsf{D}, \mathsf{sk}(t), t \cdot \mathsf{ct}(t)) \\ &\leq (t+2)^{\log t} \cdot p_5(\lambda),\end{aligned}$$

where p_5 is a polynomial that subsumes the complexity of decryption in FE_1.

3.3. *Iteratively Applying Our Transformation*

In this section, we show that by iteratively applying our transformation $O(\log\log\lambda)$ times we obtain a t-input scheme, where $t = (\log\lambda)^{\delta}$ for some constant $0 < \delta < 1$. We prove the following two theorems:

Lemma 3.9. *Let $T = T(\lambda)$, $Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda)$, $Q_{\mathsf{enc}} = Q_{\mathsf{enc}}(\lambda)$ and $\mu = \mu(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$ and let $\epsilon \in (0,1)$. Assume any $\left(T, Q_{\mathsf{key}}, Q_{\mathsf{enc}}, \mu\right)$-selectively secure single-input private-key functional encryption scheme with the following properties:*

1. *it supports circuits and messages of size $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$ and*
2. *the size of a ciphertext and a functional key is bounded by $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$,*

then for some constant $\delta \in (0,1)$, there exists a $\left(T', Q'_{\mathsf{key}}, Q'_{\mathsf{enc}}, \mu'\right)$-selectively secure $(\log\lambda)^{\delta}$-input private-key functional encryption scheme with the following properties:

1. *it supports circuits and messages of size $\mathsf{poly}(2^{(\log\lambda)^{\epsilon}})$,*
2. *$T'(\lambda) \geq T(\lambda) - (\log\log\lambda)\cdot p(\lambda)$,*
3. *$Q'_{\mathsf{key}}(\lambda) \geq Q_{\mathsf{key}}(\lambda) - (2\log\lambda)\cdot Q_{\mathsf{enc}}(\lambda)$,*
4. *$Q'_{\mathsf{enc}}(\lambda) = Q_{\mathsf{enc}}(\lambda)$, and*
5. *$\mu'(\lambda) \leq 2^{(3\log\log\lambda)^2}\cdot (Q_{\mathsf{enc}}(\lambda))^{2(\log\lambda)^{\delta}+2}\cdot (Q_{\mathsf{key}}(\lambda))^{\log\log\lambda}\cdot\mu(\lambda)$.*

Proof. Let FE_1 be a $\left(T, Q_{\mathsf{key}}, Q_{\mathsf{enc}}, \mu\right)$-selectively secure single-input scheme with the properties from the statement.

Let us analyze the complexity of the t-input scheme where $t(\lambda) = (\log\lambda)^{\delta}$, where $\delta > 0$ is some fixed constant that we fix later. In terms of complexity, using the properties of the single-input scheme and our efficiency analysis from Sect. 3.2, we have that *setup* takes a polynomial time in λ, *key-generation* for a function of size s takes time at most $(s\cdot\lambda)^{t^{\log c}}$ and *encryption* of a message of length m takes time $(m\cdot\lambda)^{t^{\log c}}$ for some large enough constant $c > 1$ (recall that c is an upper bound on the exponents of the running time of key generation and encryption procedures of the underlying single-input scheme). Plugging in $t = (\log\lambda)^{\delta}$, where $\delta = 2\epsilon/(3\log c)$, and $s, m \leq 2^{c'\cdot(\log\lambda)^{\epsilon}}$ for any $c' \in \mathbb{N}$, we get that key-generation and encryption take time at most

$$\begin{aligned}\left(2^{c'\cdot(\log\lambda)^{\epsilon}}\cdot\lambda\right)^{t^{\log c}} &= \left(2^{c'\cdot(\log\lambda)^{\epsilon}}\cdot\lambda\right)^{(\log\lambda)^{2\epsilon/3}}\\ &= 2^{c'\cdot(\log\lambda)^{5\epsilon/3}}\cdot 2^{(\log\lambda)^{5\epsilon/3}} = 2^{(c'+1)\cdot(\log\lambda)^{5\epsilon/3}}.\end{aligned}$$

For large enough λ, decryption of such a key-message pair takes time at most $\mathsf{poly}(2^{(\log\lambda)^{5\epsilon/3}})\cdot(t+2)^{\log t} \leq 2^{(\log\lambda)^{2\epsilon}}$.

In terms of security, by Theorem 3.1, we have that if FE_t is $(T^{(t)}, Q^{(t)}_{\mathsf{key}}, Q^{(t)}_{\mathsf{enc}}, \mu^{(t)})$-selectively secure and PRF is a $(T^{(t)}, \mu^{(t)})$-secure puncturable pseudorandom function family, then FE_{2t} is $(T^{(2t)}, Q^{(2t)}_{\mathsf{key}}, Q^{(2t)}_{\mathsf{enc}}, \mu^{(2t)})$-selectively secure, where

1. $T^{(2t)}(\lambda) = T^{(t)}(\lambda) - p(\lambda)$,
2. $Q^{(2t)}_{\mathsf{key}}(\lambda) = Q^{(t)}_{\mathsf{key}}(\lambda) - t(\lambda)\cdot Q^{(t)}_{\mathsf{enc}}$,

3. $Q_{\mathsf{enc}}^{(2t)}(\lambda) = Q_{\mathsf{enc}}^{(t)}(\lambda)$, and
4. $\mu^{(2t)}(\lambda) = 8t(\lambda) \cdot (Q_{\mathsf{enc}}(\lambda))^{t(\lambda)+1} \cdot Q_{\mathsf{key}}(\lambda) \cdot \mu^{(t)}(\lambda)$.

Iterating these recursive equations, using the fact that $Q_{\mathsf{key}}^{(2t)} \leq Q_{\mathsf{key}}^{(t)}$, and plugging in our initial scheme parameters, we get that

$$\begin{aligned}
Q'_{\mathsf{enc}}(\lambda) &= Q_{\mathsf{enc}}^{(1)}(\lambda) = Q_{\mathsf{enc}}(\lambda), \\
Q'_{\mathsf{key}}(\lambda) &= Q_{\mathsf{key}}^{(t)}(\lambda) - t(\lambda) \cdot Q_{\mathsf{enc}}(\lambda) \\
&\geq Q_{\mathsf{key}}(\lambda) - 2t(\lambda) \cdot Q_{\mathsf{enc}}(\lambda) \\
&\geq Q_{\mathsf{key}}(\lambda) - (2\log(\lambda)) \cdot Q_{\mathsf{enc}}(\lambda), \\
T'(\lambda) &\geq T(\lambda) - \log t(\lambda) \cdot p(\lambda) \\
&\geq T(\lambda) - (\log\log\lambda) \cdot p(\lambda), \\
\mu'(\lambda) &\leq (8t(\lambda))^{\log t(\lambda)} \cdot (Q_{\mathsf{enc}}(\lambda))^{2t(\lambda)+2} \cdot (Q_{\mathsf{key}}(\lambda))^{\log t(\lambda)} \cdot \mu(\lambda) \\
&\leq 2^{(3\log t(\lambda))^2} \cdot (Q_{\mathsf{enc}}(\lambda))^{2t(\lambda)+2} \cdot (Q_{\mathsf{key}}(\lambda))^{\log t(\lambda)} \cdot \mu(\lambda) \\
&\leq 2^{(3\log\log\lambda)^2} \cdot (Q_{\mathsf{enc}}(\lambda))^{2(\log\lambda)^\delta+2} \cdot (Q_{\mathsf{key}}(\lambda))^{\log\log\lambda} \cdot \mu(\lambda),
\end{aligned}$$

as needed. □

Claim 3.10. *Let $\lambda \in \mathbb{N}$ be a security parameter and fix any constant $\epsilon \in (0,1)$. Assuming any $(2^{2\cdot(\log\lambda)^{1/\epsilon}}, 2^{2\cdot(\log\lambda)^{1/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme supporting polynomial-size circuits, there exists a $(2^{2\cdot(\log\lambda)^2}, 2^{2\cdot(\log\lambda)^2}, 2^{(\log\lambda)^2}, 2^{-(\log\lambda)^3})$-selectively secure single-input private-key functional encryption scheme with the following properties*

1. *it supports circuits and messages of size $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$ and*
2. *the size of a ciphertext and a functional key is bounded by $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$.*

Proof. We instantiate the given scheme with security parameter $\tilde{\lambda} = 2^{(\log\lambda)^{2\epsilon}}$. The resulting scheme is $\left(2^{2\cdot(\log\lambda)^2}, 2^{2\cdot(\log\lambda)^2}, 2^{(\log\lambda)^2}, 2^{-(\log\lambda)^3}\right)$-selectively secure and for a circuit (resp., message) of size $\tilde{\lambda}$, the size of a functional key (resp., ciphertext) is bounded by $\mathsf{poly}(\tilde{\lambda})$. □

Combining Claim 3.10 and Theorem 3.9, we get the following theorem.

Theorem 3.11. *Let $\lambda \in \mathbb{N}$ be a security parameter and fix any constant $\epsilon \in (0,1)$. Assuming any $(2^{2\cdot(\log\lambda)^{1/\epsilon}}, 2^{1\cdot(\log\lambda)^{2/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme supporting polynomial-size circuits, then for some $\delta \in (0,1)$, there exists a $(2^{(\log\lambda)^2}, 2^{(\log\lambda)^2}, 2^{(\log\lambda)^2}, 2^{-(\log\lambda)^2})$-selectively secure $(\log\lambda)^\delta$-input private-key functional encryption scheme supporting circuits of size $2^{(\log\lambda)^\epsilon}$.*

Proof. Assuming any $\left(2^{2\cdot(\log\lambda)^{1/\epsilon}}, 2^{2\cdot(\log\lambda)^{1/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}}\right)$-selectively secure single-input private-key functional encryption scheme supporting polynomial-size

circuits. By Claim 3.10, it implies a $\left(2^{2\cdot(\log\lambda)^2}, 2^{2\cdot(\log\lambda)^2}, 2^{(\log\lambda)^2}, 2^{-(\log\lambda)^3}\right)$-selectively secure single-input private-key functional encryption scheme with the following properties:

1. it supports circuits and messages of size $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$ and
2. the size of a ciphertext and a functional key is bounded by $\mathsf{poly}(2^{(\log\lambda)^{2\epsilon}})$.

Using Theorem 3.9, we get that for some constant $\delta \in (0,1)$, there exists a $\left(T', Q'_{\mathsf{key}}, Q'_{\mathsf{enc}}, \mu'\right)$-selectively secure $(\log\lambda)^\delta$-input private-key functional encryption scheme with the following properties:

1. it supports circuits and messages of size at most $\mathsf{poly}(2^{(\log\lambda)^{\epsilon/2}})$,
2. $T'(\lambda) \geq 2^{2\cdot(\log\lambda)^2} - (\log\log\lambda)\cdot p(\lambda) \geq 2^{(\log\lambda)^2}$,
3. $Q'_{\mathsf{key}}(\lambda) \geq 2^{2\cdot(\log\lambda)^2} - (2\log\lambda)\cdot 2^{(\log\lambda)^2} \geq 2^{(\log\lambda)^2}$,
4. $Q'_{\mathsf{enc}}(\lambda) = 2^{(\log\lambda)^2}$, and
5. $\mu'(\lambda) \leq 2^{(3\log\log\lambda)^2}\cdot(2^{(\log\lambda)^2})^{2(\log\lambda)^\delta+2}\cdot(2^{(\log\lambda)^2})^{\log\log\lambda}\cdot 2^{-(\log\lambda)^3} \leq 2^{-(\log\lambda)^2}$.

□

4. Applications of Our Construction

In this section, we present our construction of an indistinguishability obfuscator for circuits with inputs of poly-logarithmic length, and its applications to public-key functional encryption and average-case PPAD hardness.

4.1. *Obfuscation for Circuits with Poly-logarithmic Input Length*

We show that any selectively secure t-input private-key functional encryption scheme that supports circuits of size s can be used to construct an indistinguishability obfuscator that supports circuits of size s that have at most $t\cdot\log\lambda$ inputs, where $\lambda\in\mathbb{N}$ is the security parameter. This is similar to the proof of Goldwasser et al. [32] that showed that private-key multi-input functional encryption for a polynomial number of inputs implies indistinguishability obfuscation (and a follow-up refinement of Bitansky et al. [15]).

We consider the following restricted class of circuits:

Definition 4.1. Let $\lambda\in\mathbb{N}$ and let $s(\cdot)$ and $t'(\cdot)$ be functions. Let $\mathcal{C}_\lambda^{s,t'}$ denote the class of all circuits of size at most $s(\lambda)$ that get as input $t'(\lambda)$ bits.

Lemma 4.2. *Let $t = t(\lambda)$, $s = s(\lambda)$, $T = T(\lambda)$, $Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda)$, $Q_{\mathsf{enc}} = Q_{\mathsf{enc}}(\lambda)$ and $\mu = \mu(\lambda)$ be functions of the security parameter $\lambda\in\mathbb{N}$, and assume a $(T, Q_{\mathsf{key}}, Q_{\mathsf{enc}}, \mu)$-selectively secure t-input private-key functional encryption scheme for functions of size at most s, where $Q_{\mathsf{key}}(\lambda)\geq 1$ and $Q_{\mathsf{enc}}(\lambda)\geq\lambda$. Then, there exists a $(T(\lambda)-\lambda\cdot t(\lambda)\cdot p(\lambda), \mu(\lambda))$-secure indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda^{s,t'}$, where $p(\cdot)$ is some fixed polynomial and $t'(\lambda) = t(\lambda)\cdot\log\lambda$.*

Proof. Let FE_t be a t-input scheme as in the statement of the lemma. We construct an obfuscator for circuits of size at most $s(\lambda)$ that receive $t(\lambda) \cdot \log \lambda$ bits as input. On input a circuit $C \in \mathcal{C}_\lambda^{s,t'}$, the obfuscator works as follows:

1. Sample a master secret key $\mathsf{msk} \leftarrow \mathsf{FE}_t.\mathsf{S}(1^\lambda)$.
2. Compute $\mathsf{ct}_{i,j} = \mathsf{FE}_t.\mathsf{E}(\mathsf{msk}, i, j)$ for every $i \in \{0,1\}^{\log \lambda}$ and $j \in [t(\lambda)]$.
3. Compute $\mathsf{sk}_C = \mathsf{FE}_t.\mathsf{KG}(\mathsf{msk}, C)$
4. Output $\widehat{C} = \{\mathsf{sk}_C\} \cup \{\mathsf{ct}_{i,j}\}_{i \in \{0,1\}^{\log \lambda}, j \in [t(\lambda)]}$.

Evaluation of an obfuscated circuit $\widehat{C}$ on an input $x \in (\{0,1\}^{\log \lambda})^t$, where we view x as $x = x_1 \ldots x_t$ and $x_i \in \{0,1\}^{\log \lambda}$, is done by outputting the result of a single execution of the decryption procedure of the t-input scheme $\mathsf{FE}_t.\mathsf{D}(\mathsf{sk}_C, \mathsf{ct}_{x_1,1}, \ldots, \mathsf{ct}_{x_t,t})$. Notice that the description size of the obfuscated circuit is upper bounded by some fixed polynomial in λ.

For security, notice that a single functional key is generated and it is for a circuit of size at most $s(\lambda)$. Moreover, the number of ciphertexts is bounded by λ ciphertexts per coordinate. Thus, following [32], one can show that an adversary that can break the security of the above obfuscator can be used to break the security of the FE_t scheme with the same success probability (it can even break FE_t that satisfies a weaker security notion in which the functional keys are also fixed ahead of time, before seeing any ciphertext). □

Applying Lemma 4.2 with the t-input scheme from Theorem 3.11, we obtain the following corollary.

Corollary 4.3. *Let $\lambda \in \mathbb{N}$ be a security parameter and fix any constant $\epsilon \in (0,1)$. Assume a $(2^{2(\log \lambda)^{1/\epsilon}}, 2^{2(\log \lambda)^{1/\epsilon}}, 2^{(\log \lambda)^{1/\epsilon}}, 2^{-(\log \lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme for all functions of polynomial size. Then, for some constant $\delta \in (0,1)$, there exists a $(2^{(\log \lambda)^2}, 2^{-(\log \lambda)^2})$-secure indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda^{2^{O((\log \lambda)^\epsilon)}, (\log \lambda)^{1+\delta}}$.*

4.2. *Public-Key Functional Encryption*

In this section, we present a construction of a public-key functional encryption scheme based on our multi-input private-key scheme.

Theorem 4.4. *Let $\lambda \in \mathbb{N}$ be a security parameter and fix any $\epsilon \in (0,1)$. There exists a constant $\delta > 0$ for which the following holds. Assume a $(2^{2(\log \lambda)^{1/\epsilon}}, 2^{2(\log \lambda)^{1/\epsilon}}, 2^{(\log \lambda)^{1/\epsilon}}, 2^{-(\log \lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme for all functions of polynomial size, and that $(2^{2\lambda^{\epsilon'}}, 2^{-2\lambda^{\epsilon'}})$-secure one-way functions exist for $\epsilon' > 1/(1+\delta)$. Then, for some constant $\zeta > 1$, there exists a $(2^{(\log \lambda)^\zeta}, 2^{(\log \lambda)^\zeta}, 2^{-(\log \lambda)^\zeta})$-selectively secure public-key encryption scheme for the circuit class $\mathcal{C}_\lambda^{2^{O((\log \lambda)^\epsilon)}, (\log \lambda)^{1+\delta}}$.*

Our construction is essentially the construction of Waters [54], who showed how to construct a public-key functional encryption scheme for the set of all polynomial-size

circuits assuming indistinguishability obfuscation for all polynomial-size circuits. Here, we make a more careful analysis of his scheme and show that for a specific range of parameters, it suffices to use the obfuscator we have obtained in Corollary 4.3.

Waters' construction builds upon his notion of puncturable deterministic encryption which we review in Sect. 4.2.1. In Sect. 4.2.2, we present the construction and analyze its security. In Sect. 4.2.3, we prove Theorem 4.4.

4.2.1. *Puncturable Deterministic Encryption*

Here, we review the functionality and security of puncturable deterministic encryption (PDE) as put forward by Waters [54]. Let $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be a message space and $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$ be a key space. A PDE scheme consists of four probabilistic polynomial time algorithms $\mathsf{PDE} = (\mathsf{PDE.S}, \mathsf{PDE.E}, \mathsf{PDE.D}, \mathsf{PDE.P})$. The setup procedure $\mathsf{PDE.S}$ gets as input a security parameter (in unary representation) and generates a key $K \in \mathcal{K}_\lambda$. The encryption procedure $\mathsf{PDE.E}$ is a *deterministic* procedure that takes as input a key $K \in \mathcal{K}_\lambda$ and a message $m \in \mathcal{M}_\lambda$, and output a ciphertext ct. The decryption procedure $\mathsf{PDE.D}$ takes as input a key $K \in \mathcal{K}_\lambda$ and a ciphertext ct and outputs either a message $m \in \mathcal{M}_\lambda$ or $\perp$. The puncturing procedure $\mathsf{PDE.P}$ takes as input a key $K \in \mathcal{K}_\lambda$ as well as a pair of messages $x^0, x^1 \in \mathcal{M}_\lambda$, and output a "punctured" key $K|_{\{x^0,x^1\}}$.

Definition 4.5. (*Correctness*) A PDE scheme $\mathsf{PDE} = (\mathsf{PDE.S}, \mathsf{PDE.E}, \mathsf{PDE.D}, \mathsf{PDE.P})$ is ρ-*correct* if for all $\lambda \in \mathbb{N}$, all $x^0, x^1 \in \mathcal{M}_\lambda$, all $m \in \mathcal{M}_\lambda$, it holds that

$$\Pr\left[\mathsf{PDE.D}(K, \mathsf{PDE.E}(K, m)) \neq m\right] \leq \rho(\lambda),$$

and for all $m \in \mathcal{M}_\lambda \setminus \{x^0, x^1\}$

$$\Pr\left[\mathsf{PDE.D}(K|_{\{x^0,x^1\}}, \mathsf{PDE.E}(K, m)) \neq m\right] \leq \rho(\lambda),$$

where $K \leftarrow \mathsf{PDE.S}(1^\lambda)$, $K|_{\{x^0,x^1\}} \leftarrow \mathsf{PDE.P}(K, x^0, x^1)$, and the probabilities are taken over the internal randomness of $\mathsf{PDE.S}$ and of $\mathsf{PDE.P}$.

Definition 4.6. (*PDE security*) A PDE scheme $\mathsf{PDE} = (\mathsf{PDE.S}, \mathsf{PDE.E}, \mathsf{PDE.D}, \mathsf{PDE.P})$ over a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ is (t, μ)-*secure* if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that runs in time $t = t(\lambda)$ it holds that

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

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

1. $x^0, x^1 \leftarrow \mathcal{A}_1(1^\lambda)$.
2. $K \leftarrow \mathsf{PDE.S}(1^\lambda)$, $K|_{\{x^0,x^1\}} \leftarrow \mathsf{PDE.P}(K, x^0, x^1)$.
3. $\mathsf{ct}_0 = \mathsf{PDE.E}(K, x^0)$, $\mathsf{ct}_1 = \mathsf{PDE.E}(K, x^1)$, $b \leftarrow \{0, 1\}$.
4. $b' \leftarrow \mathcal{A}_2(K|_{\{x^0,x^1\}}, \mathsf{ct}_b, \mathsf{ct}_{1-b}))$.
5. If $b' = b$ then output 1, and otherwise output 0.

We say that PDE is *secure* if it is (t, μ)-secure for some $t = t(\lambda)$ that is super-polynomial and $\mu = \mu(\lambda)$ that is negligible. We say that PDE is sub-exponentially secure if $t(\lambda) = 1/\mu(\lambda) = 2^{\lambda^\epsilon}$ for some constant $0 < \epsilon < 1$.

Waters [54] presented an elegant construction of a PDE scheme assuming any puncturable PRF family (which, in turn, is known to exist based on any one-way function [12,23,45,53]). Here, we state a parameterized version of his result.

Lemma 4.7. *Assume the existence of a* $(T_{\mathsf{OWF}}, \mu_{\mathsf{OWF}})$*-secure one-way function* $f : \{0,1\}^\lambda \to \{0,1\}^\lambda$*. Then, there exists a* $2^{-\lambda}$*-correct* $(T_{\mathsf{OWF}} - p_1(\lambda), \mu_{\mathsf{OWF}} - p_2(\lambda))$*-secure PDE scheme, where* $p_1(\cdot)$ *and* $p_2(\cdot)$ *are two fixed polynomials, with the following properties:*

1. $\mathcal{M}_\lambda = \{0,1\}^\lambda$ *and* $\mathcal{K}_\lambda = \{0,1\}^{2\lambda}$.
2. *For every* $m \in \{0,1\}^\lambda$ *and* $K \in \{0,1\}^{2\lambda}$ *it holds that* $\mathsf{PDE.E}(K, m) \in \{0,1\}^{3\lambda+\lambda} = \{0,1\}^{4\lambda}$.
3. PDE.D *gets as input elements in* $\{0,1\}^{2\lambda} \times \{0,1\}^{4\lambda}$ *and runs in fixed polynomial time in* λ.
4. PDE.P *gets as input elements in* $\{0,1\}^{2\lambda} \times \{0,1\}^\lambda \times \{0,1\}^\lambda$ *and outputs an element which is of fixed polynomial size in* λ.

4.2.2. *The Construction*

In this section, we present our construction of a public-key functional encryption scheme for the class $\mathcal{C}_\lambda^{s,t}$ of circuits, where $s(\lambda) = 2^{O((\log \lambda)^\epsilon)}$ and $t(\lambda) = (\log \lambda)^{1+\delta}$ for some fixed constants $\epsilon, \delta > 0$. Our construction relies on the following building blocks:

1. An indistinguishability obfuscator $i\mathcal{O}$ for the class of circuits $\mathcal{C}_\lambda^{s,t}$.
2. A puncturable deterministic encryption PDE = (PDE.S, PDE.E, PDE.D, PDE.P) for the message space $\mathcal{M}_\lambda = \{0,1\}^{t(\lambda)/6}$.
3. A puncturable pseudorandom function family PRF = (PRF.Gen, PRF.Eval, PRF.Punc) with key space $\{0,1\}^{t(\lambda)}$, domain $\{0,1\}^{t(\lambda)/3}$, and range $\{0,1\}^{t(\lambda)/3}$.
4. A length-doubling pseudorandom generation $\mathsf{PRG}\colon \{0,1\}^{t(\lambda)/6} \to \{0,1\}^{t(\lambda)/3}$.

Our scheme pkFE = (pkFE.S, pkFE.KG, pkFE.E, pkFE.D) is defined as follows.

- **The setup algorithm** On input the security parameter 1^λ the setup algorithm pkFE.S samples a PRF key $K \leftarrow \mathsf{PRF.Gen}(1^\lambda)$ and computes $\widehat{C_K} = i\mathcal{O}(C_K)$, where the circuit C_K that is defined in Fig. 6. The output of the procedure is $\mathsf{msk} = K$ and $\mathsf{mpk} = \widehat{C_K}$.

$C_K(r)$:
1. Compute $z = \mathsf{PRG}(r)$.
2. Compute $k = \mathsf{PRF.Eval}(K, z)$.
3. Output (z, k).

$P_{f,K,z^*,c_0,c_1,y,k'}(z, c)$:
1. If $z^* \neq \perp$ and $z = z^*$ then
   (a) If $c \notin \{c_0, c_1\}$ output $f(\mathsf{PDE.D}(k', c))$.
   (b) If $c \in \{c_0, c_1\}$ output y.
2. Else, compute $k = \mathsf{PRF.Eval}(K, z)$.
3. Output $f(\mathsf{PDE.D}(k, c))$.

Fig. 6. The functions C_K and $P_{f,K,z^*,c_0,c_1,y,k'}$.

- **The key-generation algorithm** On input the master secret key msk and a function $f \in \mathcal{F}_\lambda$, the key-generation algorithm $\mathsf{pkFE.KG}$ outputs $\mathsf{sk}_f = i\mathcal{O}(P_{f,K,\perp,\perp,\perp,\perp,\perp})$, where the circuit $P_{f,K,z^*,c_0,c_1,y,k'}$ is defined in Fig. 6.
- **The encryption algorithm** On input the master public key $\mathsf{mpk} = \widehat{C_K}$ and a message x, the encryption algorithm $\mathsf{pkFE.E}$ chooses a random $r \in \{0,1\}^{t(\lambda)/6}$ and runs the obfuscated program $\widehat{C_K}$ on r to get (z,k). It then computes $c = \mathsf{PDE.E}(k,x)$ and outputs $\mathsf{ct} = (z,c)$.
- **The decryption algorithm** On input a functional key $\mathsf{sk}_f = \widehat{P_{f,K}}$ and a ciphertext $\mathsf{ct} = (t,c)$ the decryption procedure $\mathsf{pkFE.D}$ runs the obfuscated program $\widehat{P_{f,K}}$ on input (z,c) and outputs the response.

Correctness and Security We argue that the assumed indistinguishability obfuscator can be used in the scheme above. The input size of C_K is $t(\lambda)/6 < t(\lambda)$ and its size is bounded by $\mathsf{poly}(t(\lambda)) < s(\lambda)$. To analyze the size of $P_{f,K,z^*,c_0,c_1,y,k'}$ we have to analyze the parameters of the underlying PDE scheme. By Lemma 4.7, its key space is $\{0,1\}^{t(\lambda)/3}$ and its ciphertext space is $\{0,1\}^{2t(\lambda)/3}$. Thus, the input size of $P_{f,K,z^*,c_0,c_1,y,k'}$ is $t(\lambda)/3+2t(\lambda)/3 = t(\lambda)$ and for a function $f \in \mathcal{C}_\lambda^{s,t}$ of size $2^{c\cdot(\log\lambda)^\epsilon}$ its size is bounded by $\mathsf{poly}(s(\lambda),t(\lambda)) = \mathsf{poly}(2^{c\cdot(\log\lambda)^\epsilon},(\log\lambda)^{1+\delta}) \leq s(\lambda)$. Thus, the obfuscator can be used. Now, the fact that the scheme is correct follows directly from the correctness of the underlying indistinguishability obfuscator and the puncturable deterministic scheme (see [54] for more details).

The following theorem, which is proved in Sect. 4.2.4, captures the security of the scheme (note that given the generic transformation of Ananth et al. [3] it suffices to prove selective security, as any such scheme can be transformed into an adaptively secure one).

Lemma 4.8. *Let* $T_{i\mathcal{O}} = T_{i\mathcal{O}}(\lambda)$, $\mu_{i\mathcal{O}} = \mu_{i\mathcal{O}}(\lambda)$, $T_{\mathsf{PDE}} = T_{\mathsf{PDE}}(\lambda)$, $\mu_{\mathsf{PDE}} = \mu_{\mathsf{PDE}}(\lambda)$, $T_{\mathsf{PRF}} = T_{\mathsf{PRF}}(\lambda)$, $\mu_{\mathsf{PRF}} = \mu_{\mathsf{PRF}}(\lambda)$, $T_{\mathsf{PRG}} = T_{\mathsf{PRG}}(\lambda)$, *and* $\mu_{\mathsf{PRG}} = \mu_{\mathsf{PRG}}(\lambda)$ *be functions of the security parameter* $\lambda \in \mathbb{N}$. *If* $i\mathcal{O}$ *is a* $(T_{i\mathcal{O}},\mu_{i\mathcal{O}})$*-secure indistinguishability obfuscator,* PDE *is a* η_{PDE}*-correct* $(T_{\mathsf{PDE}},\mu_{\mathsf{PDE}})$*-deterministic encryption scheme,* PRF *is a* $(T_{\mathsf{PRF}},\mu_{\mathsf{PRF}})$*-secure puncturable pseudorandom function, and* PRG *is a* $(T_{\mathsf{PRG}},\mu_{\mathsf{PRG}})$*-secure pseudorandom generator, then* pkFE *is a* $(T_{\mathsf{pkFE}},Q_{\mathsf{key}},\mu_{\mathsf{pkFE}})$*-selectively secure public-key functional encryption scheme, where:*

1. $T_{\mathsf{pkFE}}(\lambda) = \min\{T_{i\mathcal{O}}(\lambda), T_{\mathsf{PDE}}(t(\lambda)/6), T_{\mathsf{PRF}}(t(\lambda)), T_{\mathsf{PRG}}(t(\lambda)/6)\} - p(\lambda)$ *for some fixed polynomial* $p(\lambda)$.
2. $Q_{\mathsf{key}}(\lambda) = T_{\mathsf{pkFE}}(\lambda)$.
3. $\mu_{\mathsf{pkFE}}(\lambda) = \mu_{\mathsf{PRG}}(t(\lambda)/6) + \mu_{i\mathcal{O}}(\lambda) + 1/2^{t(\lambda)/6} + Q_{\mathsf{key}}(\lambda)\cdot(\mu_{i\mathcal{O}}(\lambda) + \eta_{\mathsf{PDE}}(t(\lambda)/6)) + \mu_{\mathsf{PRF}}(t(\lambda)) + \mu_{\mathsf{PDE}}(t(\lambda)/6)$.

4.2.3. *Proof of Theorem 4.4*

Assume a $(2^{2(\log\lambda)^{1/\epsilon}}, 2^{2(\log\lambda)^{1/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme for all functions of polynomial size. By Corollary 4.3, for every $\epsilon > 0$, there exists a $\delta > 0$ such that there exists a $(2^{(\log\lambda)^2}, 2^{-(\log\lambda)^2})$-secure indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda^{2^{O((\log\lambda)^\epsilon)},(\log\lambda)^{1+\delta}}$.

Assume a $(2^{2\lambda^{\epsilon'}}, 2^{-2\lambda^{\epsilon'}})$-secure one-way function for some constant $1/(1+\delta) < \epsilon' < 1$. Thus, the following primitives exist:

1. a $(2^{\lambda^{\epsilon'}}, 2^{-\lambda^{\epsilon'}})$-secure PDE scheme (by Lemma 4.7),
2. a $(2^{\lambda^{\epsilon'}}, 2^{-\lambda^{\epsilon'}})$-secure puncturable pseudorandom function family, and
3. a $(2^{\lambda^{\epsilon'}}, 2^{-\lambda^{\epsilon'}})$-secure pseudorandom generator.

Recall that $t(\lambda) = (\log \lambda)^{1+\delta}$ and let $\epsilon'' = (1+\delta)\cdot\epsilon'$. Notice that $\epsilon'' > 1$ by the choice of ϵ'. Plugging these primitives in Lemma 4.8, we get that there exist constants $1 < \zeta', \zeta'' < \epsilon''$ for which the public-key functional encryption scheme is $(T_{\mathsf{pkFE}}, Q_{\mathsf{key}}, \mu_{\mathsf{pkFE}})$-secure where

1. $T_{\mathsf{pkFE}}(\lambda) \geq \min\{2^{2(\log\lambda)^2}, 2^{((\log\lambda)^{(1+\delta)}/6)^{\epsilon'}}\} - p(\lambda) = 2^{((\log\lambda)^{\epsilon''}/6^{\epsilon'}} - p(\lambda) \geq 2^{(\log\lambda)^{\zeta'}}$.
2. $Q_{\mathsf{key}}(\lambda) = T_{\mathsf{pkFE}}(\lambda)$.
3. $\mu_{\mathsf{pkFE}}(\lambda) \leq 2^{((\log\lambda)^{\epsilon''}/6^{\epsilon'}} + 2^{-(\log\lambda)^2} + 2^{-(\log\lambda)^{1+\delta}/6} + Q_{\mathsf{key}}(\lambda)(2^{-(\log\lambda)^2} + 2^{((\log\lambda)^{\epsilon''}/6^{\epsilon'}}) + 2^{((\log\lambda)^{\epsilon''}/6^{\epsilon'}} + 2^{((\log\lambda)^{\epsilon''}/6^{\epsilon'}} \leq 2^{-(\log\lambda)^{\zeta''}}$.

Finally, we set $\zeta = \min\{\zeta', \zeta''\}$ and obtain the result.

4.2.4. *Proof of Lemma 4.8*

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a valid adversary that runs in time $T = T(\lambda)$ and issues at most $Q_{\mathsf{key}} = Q_{\mathsf{key}}(\lambda)$ key-generation queries. Following [54], we present a sequence of experiments and upper bound $\mathcal{A}$'s advantage in distinguishing each two consecutive experiments. The first experiment is the experiment $\mathsf{Exp}^{\mathsf{sel\text{-}pkFE}}_{\mathsf{pkFE},\mathcal{F},\mathcal{A}}$ (see Definition 2.10), and the last experiment is completely independent of the bit b. This enables us to prove that

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

for all sufficiently large $\lambda \in \mathbb{N}$. We denote by (x^0, x^1) the challenge ciphertext and by f_i the ith function with which the adversary queries the key-generation oracle with.

To ease notation and without loss of generality, instead of running $\mathsf{PDE.S}(1^\lambda)$ and $\mathsf{PRF.Gen}(1^\lambda)$ to generate PDE and PRF keys, respectively, we generate both of them by simply sampling a uniformly random string from $\{0,1\}^\lambda$. (This was done also in [54].)

Experiment $\mathcal{H}^{(0)}(\lambda)$ This is the original experiment $\mathsf{Exp}^{\mathsf{sel\text{-}pkFE}}_{\mathsf{pkFE},\mathcal{F},\mathcal{A}}$ corresponding to $b \leftarrow \{0,1\}$ chosen uniformly at random. Recall that in this experiment the ciphertexts and the functional keys are generated as follows.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_K)\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda) \quad (\text{This is } \mathsf{msk})\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ z^* &= \mathsf{PRG}(r^*),\\ k^* &= \mathsf{PRF.Eval}(K, z^*),\\ c^* &= \mathsf{PDE.E}(k^*, x^b)\end{aligned}$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow i\mathcal{O}(P_{f_i,K,\perp,\perp,\perp,\perp,\perp}).$$

Experiment $\mathcal{H}^{(1)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(1)}(\lambda)$ by modifying the challenge ciphertexts to sample z^* uniformly at random rather than using a PRG.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_K)\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda).\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ \boxed{z^* &\leftarrow \{0,1\}^{t(\lambda)/3}},\\ k^* &= \mathsf{PRF.Eval}(K, z^*),\\ c^* &= \mathsf{PDE.E}(k^*, x^b)\end{aligned}$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow i\mathcal{O}(P_{f_i,K,\perp,\perp,\perp,\perp,\perp}).$$

By the pseudorandomness of PRG, we have the following claim.

Claim 4.9. *There exists an adversary $\mathcal{B}^{(0)\to(1)}$ that runs in time $T(\lambda) \cdot \mathsf{poly}(\lambda)$, such that*

$$\left|\Pr\left[\mathcal{H}^{(0)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(1)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{PRG},\mathcal{B}^{(0)\to(1)}}(t(\lambda)/6).$$

Experiment $\mathcal{H}^{(2)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(1)}(\lambda)$ by modifying the master public key as follows. Instead of obfuscating the circuit C_K, we obfus-

cate a circuit $C_{K|_{\{z^*\}}}$ in which instead of embedding the PRF key K, we the punctured PRF key $K|_{\{z^*\}}$ at the point z^*.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_{\boxed{K|_{\{z^*\}}}})\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda).\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ z^* &\leftarrow \{0,1\}^{t(\lambda)/3},\\ k^* &= \mathsf{PRF.Eval}(K, z^*),\\ c^* &= \mathsf{PDE.E}(k^*, x^b)\end{aligned}$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\mathsf{sk}_{f_i} \leftarrow i\mathcal{O}(P_{f_i,K,\perp,\perp,\perp,\perp,\perp}).$$

For any adversary, with very high probability, it is impossible to distinguish $i\mathcal{O}(C_K)$ from $i\mathcal{O}(C_{K|_{\{z^*\}}})$ since C_K is functionally equivalent to $C_{K|_{\{z^*\}}}$. Indeed, for any input to C_K, we apply the PRF on an input which is the output of PRG. Since $z^* \notin \mathsf{Im}(\mathsf{PRG})$ with probability $1 - 2^{t(\lambda)/6}$, for every input to $C_{K|_{\{z^*\}}}$, the PRF is never evaluated at the point z^*. Thus, from the security of $i\mathcal{O}$, we have the following claim.

Claim 4.10. *There exists an adversary $\mathcal{B}^{(1)\to(2)}$ that runs in time $T(\lambda) \cdot \mathsf{poly}(\lambda)$, such that*

$$\left|\Pr\left[\mathcal{H}^{(1)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(2)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{i\mathcal{O}}_{i\mathcal{O},\mathcal{B}^{(1)\to(2)}}(\lambda) + 1/2^{t(\lambda)/6}.$$

Experiment $\mathcal{H}^{(3)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(2)}(\lambda)$ by modifying the functional keys as follows. In each such key for a function f_i, we replace in the circuit $P_{f_i,K,\perp,\perp,\perp,\perp,\perp}$ the PRF key K with the punctured key $K|_{\{z^*\}}$ and also embed the values $k' = \mathsf{PDE.P}(k^*, x^0, x^1)$, $c'_0 = \mathsf{PDE.E}(k^*, x^0)$, $c'_1 = \mathsf{PDE.E}(k^*, x^1)$, $c_0 = c'_{b'}$, $c_1 = c'_{1-b'}$ for $b' \leftarrow \{0,1\}$, and $y = f_i(x^0) = f_i(x^1)$.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_{K|_{\{z^*\}}})\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda).\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ z^* &\leftarrow \{0,1\}^{t(\lambda)/3},\\ k^* &= \mathsf{PRF.Eval}(K, z^*),\\ c^* &= \mathsf{PDE.E}(k^*, x^b)\end{aligned}$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\begin{aligned}\mathsf{sk}_{f_i} &\leftarrow i\mathcal{O}(P_{f_i, \boxed{K|_{\{z^*\}}}, \boxed{z^*}, \boxed{c_0}, \boxed{c_1}, \boxed{y}, \boxed{k'}})\\ k' &= \mathsf{PDE.P}(k^*, x^0, x^1),\\ c'_0 &= \mathsf{PDE.E}(k^*, x^0),\\ c'_1 &= \mathsf{PDE.E}(k^*, x^1),\\ c_0 &= c'_{b'}, c_1 = c'_{1-b'} \text{ for } b' \leftarrow \{0,1\},\\ y &= f_i(x^0) = f_i(x^1).\end{aligned}$$

The only difference between $i\mathcal{O}(P_{f_i,K,\perp,\perp,\perp,\perp,\perp})$ and $i\mathcal{O}(P_{f_i,K|_{\{z^*\}},z^*,c_0,c_1,y,k'})$ is that the output of the circuit is hardwired for two inputs. Thus, if the PDE is correct, then the two circuits are equivalent. Using the security of $i\mathcal{O}$ and a standard hybrid argument (over the sequence of functional keys), we have the following claim.

Claim 4.11. *There exists an adversary $\mathcal{B}^{(2)\to(3)}$ that runs in time $T(\lambda) \cdot \mathsf{poly}(\lambda)$, such that*

$$\left|\Pr\left[\mathcal{H}^{(2)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(3)}(\lambda) = 1\right]\right| \leq Q_{\mathsf{key}} \cdot (\mathsf{Adv}^{i\mathcal{O}}_{i\mathcal{O},\mathcal{B}^{(2)\to(3)}}(\lambda) + \eta_{\mathsf{PDE}}(t(\lambda)/6)).$$

Experiment $\mathcal{H}^{(4)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(3)}(\lambda)$ by modifying the challenge ciphertext as follows. Instead of computing k^* using a PRF with key K, we sample it uniformly at random.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_{K|_{\{z^*\}}})\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda).\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ z^* &\leftarrow \{0,1\}^{t(\lambda)/3},\\ &\boxed{k^* \leftarrow \{0,1\}^{t(\lambda)/3}},\\ c^* &= \mathsf{PDE.E}(k^*, x^b)\end{aligned}$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\begin{aligned}\mathsf{sk}_{f_i} &\leftarrow i\mathcal{O}(P_{f_i, K|_{\{z^*\}}, z^*, c_0, c_1, y, k'})\\ k' &= \mathsf{PDE.P}(k^*, x^0, x^1),\\ c_0' &= \mathsf{PDE.E}(k^*, x^0),\\ c_1' &= \mathsf{PDE.E}(k^*, x^1),\\ c_0 &= c_{b'}', c_1 = c_{1-b'}' \text{ for } b' \leftarrow \{0,1\},\\ y &= f_i(x^0) = f_i(x^1).\end{aligned}$$

We observe that the key K does not exist in the scheme anymore and is replaced with a punctured key $K|_{\{z^*\}}$. Thus, by the pseudorandomness at a punctured point property of the PRF, we have the following claim.

Claim 4.12. *There exists an adversary $\mathcal{B}^{(3)\to(4)}$ that runs in time $T(\lambda) \cdot \mathsf{poly}(\lambda)$, such that*

$$\left|\Pr\left[\mathcal{H}^{(3)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(4)}(\lambda) = 1\right]\right| \leq \mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF}, \mathcal{B}^{(3)\to(4)}}(t(\lambda)).$$

Experiment $\mathcal{H}^{(5)}(\lambda)$ This experiment is obtained from the experiment $\mathcal{H}^{(4)}(\lambda)$ by modifying the challenge ciphertext as follows. Instead of encrypting x^b, we encrypt x^0.

Notice that this experiment is completely independent of the bit b, and therefore $\Pr[\mathcal{H}^{(5)}(\lambda) = 1] = 1/2$.

1. Public parameters:

$$\begin{aligned}\mathsf{mpk} &\leftarrow i\mathcal{O}(C_{K|_{\{z^*\}}})\\ K &\leftarrow \mathsf{PRF.Gen}(1^\lambda).\end{aligned}$$

2. Challenge ciphertext:

$$\begin{aligned}\mathsf{ct}^* &= (z^*, c^*)\\ r^* &\leftarrow \{0,1\}^{t(\lambda)/6},\\ z^* &\leftarrow \{0,1\}^{t(\lambda)/3},\end{aligned}$$

$$k^* \leftarrow \{0,1\}^{t(\lambda)/3},$$
$$c^* = \mathsf{PDE.E}(k^*, \boxed{x^0})$$

3. Functional keys ($i = 1, \ldots, Q_{\mathsf{key}}$):

$$\begin{aligned}\mathsf{sk}_{f_i} \leftarrow i\mathcal{O}(P_{f_i, K|_{\{z^*\}}, z^*, c_0, c_1, y, k'})\\ k' = \mathsf{PDE.P}(k^*, x^0, x^1),\\ c'_0 = \mathsf{PDE.E}(k^*, x^0),\\ c'_1 = \mathsf{PDE.E}(k^*, x^1),\\ c_0 = c'_{b'}, c_1 = c'_{1-b'} \text{ for } b' \leftarrow \{0,1\},\\ y = f_i(x^0) = f_i(x^1).\end{aligned}$$

We observe that the security of the PDE gives the following claim.

Claim 4.13. *There exists an adversary $\mathcal{B}^{(4)\to(5)}$ that runs in time $T(\lambda) \cdot \mathsf{poly}(\lambda)$, such that*

$$\left|\Pr\left[\mathcal{H}^{(3)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(4)}(\lambda) = 1\right]\right| \le \mathsf{Adv}^{\mathsf{PDE}}_{\mathsf{PDE}, \mathcal{B}^{(4)\to(5)}}(t(\lambda)/6).$$

Putting together Claims 4.9–4.13, we get that

$$\begin{aligned}\mathsf{Adv}^{\mathsf{sel\text{-}pkFE}}_{\Pi,\mathcal{F},\mathcal{A}} &\stackrel{\mathsf{def}}{=} \left|\Pr\left[\mathsf{Exp}^{\mathsf{sel\text{-}pkFE}}_{\Pi,\mathcal{F},\mathcal{A}}(\lambda) = 1\right] - \frac{1}{2}\right|\\ &= \left|\Pr\left[\mathcal{H}^{(0)}(\lambda) = 1\right] - \Pr\left[\mathcal{H}^{(5)}(\lambda) = 1\right]\right|\\ &\le \mathsf{Adv}^{\mathsf{PRG}}_{\mathsf{PRG}, \mathcal{B}^{(0)\to(1)}}(t(\lambda)/6) + \mathsf{Adv}^{i\mathcal{O}}_{i\mathcal{O}, \mathcal{B}^{(1)\to(2)}}(\lambda) + 1/2^{t(\lambda)/6}\\ &\quad + Q_{\mathsf{key}} \cdot (\mathsf{Adv}^{i\mathcal{O}}_{i\mathcal{O}, \mathcal{B}^{(2)\to(3)}}(\lambda) + \eta_{\mathsf{PDE}}(t(\lambda)/6)) + \mathsf{Adv}^{\mathsf{puPRF}}_{\mathsf{PRF}, \mathcal{B}^{(3)\to(4)}}(t(\lambda))\\ &\quad + \mathsf{Adv}^{\mathsf{PDE}}_{\mathsf{PDE}, \mathcal{B}^{(4)\to(5)}}(t(\lambda)/6)\\ &\le \mu_{\mathsf{PRG}}(t(\lambda)/6) + \mu_{i\mathcal{O}}(\lambda) + 1/2^{t(\lambda)/6}\\ &\quad + Q_{\mathsf{key}}(\lambda)(\mu_{i\mathcal{O}}(\lambda) + \eta_{\mathsf{PDE}}(t(\lambda)/6)) + \mu_{\mathsf{PRF}}(t(\lambda)) + \mu_{\mathsf{PDE}}(t(\lambda)/6).\end{aligned}$$

4.3. *Average-Case PPAD Hardness*

We present a construction of a hard-on-average distribution of sink-of-verifiable-line (SVL) instances assuming any quasi-polynomially secure private-key (single-input) functional encryption scheme and sub-exponentially secure one-way function. Following the work of Abbot et al. [7] and Bitansky et al. [16], this shows that the complexity class PPAD [24,29,30,50] contains complete problems that are hard on average (we refer the reader to [16] for more details). In what follows, we first recall the SVL problem, and then state and prove our hardness result.

Hardwired: $K \in \{0,1\}^{t(\lambda)}$
Input: $i \in [T(\lambda)]$, $\sigma \in \{0,1\}^{t(\lambda)/2}$

$\mathsf{VS}_K(i,\sigma)$:

1. If $\sigma \neq \mathsf{PRF.Eval}(K,i)$, return $\perp$ and HALT.
2. If $i = T$, return SOLVED and HALT.
3. Return $(i+1, \mathsf{PRF.Eval}(K, i+1))$.

Padding: The circuit is padded to size $s(\lambda)$, where s is some fixed polynomial.

Fig. 7. The functions VS_K.

Definition 4.14. (*Sink-of-verifiable-line*) An SVL instance $(\mathsf{S}, \mathsf{V}, x_s, T)$ consists of a source $x_s \in \{0,1\}^\lambda$, a target index $T \in [2^\lambda]$, and a pair of circuits $\mathsf{S}\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$ and $\mathsf{V}\colon \{0,1\}^\lambda \times [T] \to \{0,1\}$, such that for $(x,i) \in \{0,1\}^\lambda \times [T]$, it holds that $\mathsf{V}(x,i) = 1$ if and only if $x = x_i = \mathsf{S}^{i-1}(x_s)$, where $x_1 = x_s$. A string $w \in \{0,1\}^\lambda$ is a valid witness if and only if $\mathsf{V}(w,T) = 1$.

Theorem 4.15. *Let $\lambda \in \mathbb{N}$ be a security parameter and fix any constant $\epsilon \in (0,1)$. Assume a $(2^{2(\log\lambda)^{1/\epsilon}}, 2^{2(\log\lambda)^{1/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme for all functions of polynomial size, and that $(2^{\lambda^{2\epsilon'}}, 2^{-\lambda^{2\epsilon'}})$-secure injective one-way functions exist for some large enough constant $\epsilon' \in (0,1)$. Then, there exists a distribution with an associated efficient sampling procedure that generates instances of sink-of-verifiable-line which are hard to solve for any polynomial-time algorithm.*

We present the construction of a hard-on-average SVL distribution. This distribution is efficiently samplable, and we later show that under appropriate cryptographic assumptions, it is hard to solve it in polynomial time. The construction relies on the following building blocks:

1. An indistinguishability obfuscator $i\mathcal{O}$ for the class of circuits $\mathcal{C}_\lambda^{s,t}$, where $s(\lambda) = 2^{O(\log\lambda)^\epsilon}$ and $t(\lambda) = (\log\lambda)^{1+\delta}$ for some fixed constants $\epsilon, \delta > 0$.
2. A puncturable pseudorandom function family $\mathsf{PRF} = (\mathsf{PRF.Gen}, \mathsf{PRF.Eval})$ with key space $\{0,1\}^{t(\lambda)}$, domain $[T(\lambda)]$ and range $\{0,1\}^{t(\lambda)/2}$, where $T(\lambda) = 2^{(\log\lambda)^{1+\delta/3}}$.
3. An injective one-way function $\mathsf{OWF}\colon \{0,1\}^{t(\lambda)/2} \to \{0,1\}^{\ell(t(\lambda))}$, where $\ell(\lambda)$ is any fixed polynomial.[9]

The construction consists of an obfuscation of a (padded) circuit VS that given a valid signature on an index i produces a signature on the next index $i+1$, where signatures will be implemented by the puncturable PRF. The circuit is formally described in Fig. 7.

More precisely, an instance in the distribution is composed of $\widehat{\mathsf{VS}_K} \leftarrow i\mathcal{O}(\mathsf{VS}_K)$, an obfuscation of the circuit VS_K, where $K \leftarrow \mathsf{PRF.Gen}(1^{t(\lambda)})$ and a signature $\sigma_1 =$

[9] The injective one-way function can be relaxed to be a family of one-way functions such that a random element is an injective function with high probability. Furthermore, this primitive will not be used in the construction, but rather only in the proof of security.

$\mathsf{PRF}_K(1)$. This induces an SVL instance $(\mathsf{S}, \mathsf{V}, x_s, T)$ where the successor circuit S computes $\widehat{\mathsf{VS}_K}$, the verification circuit V uses $\widehat{\mathsf{VS}_K}$ to test inputs along the chain from the source input $x_s = (1, \sigma_1)$ to the target input (T, σ_T).

We observe that the circuit VS_K has input of length $t(\lambda)/2 + t(\lambda)/2 = t(\lambda)$ and the size of VS_K is bounded by some fixed polynomial in $t(\lambda)$ which is smaller than $s(\lambda)$.

For security, we rely on sub-exponentially secure injective one-way functions and the $i\mathcal{O}$ we obtained in Corollary 4.3. We use a parameterized version of the main result of [16].

Theorem 4.16. ([16]) *Let $T_{i\mathcal{O}} = T_{i\mathcal{O}}(\lambda)$, $\mu_{i\mathcal{O}} = \mu_{i\mathcal{O}}(\lambda)$, $T_{\mathsf{PRF}} = T_{\mathsf{PRF}}(\lambda)$, $\mu_{\mathsf{PRF}} = \mu_{\mathsf{PRF}}(\lambda)$, and $\mu_{\mathsf{OWF}} = \mu_{\mathsf{OWF}}(\lambda)$ be functions of the security parameter $\lambda \in \mathbb{N}$. Assume that $i\mathcal{O}$ is a $(T_{i\mathcal{O}}, \mu_{i\mathcal{O}})$-secure indistinguishability obfuscator for the class $\mathcal{C}_\lambda^{s,t}$, PRF be a $(T_{\mathsf{PRF}}, \mu_{\mathsf{PRF}})$-secure puncturable pseudorandom function, OWF be a $(T_{\mathsf{OWF}}, \mu_{\mathsf{OWF}})$-secure family of injective one-way functions, then for any distinguisher D that runs in time at most $\min\{T_{i\mathcal{O}}(\lambda), T_{\mathsf{PRF}}(t(\lambda)), T_{\mathsf{OWF}}(t(\lambda)/2)\}$, the probability of solving the SVL problem is at most*

$$\mu_{\mathsf{OWF}}(\log T(\lambda)) + T(\lambda) \cdot (\mu_{i\mathcal{O}}(\lambda) + \mu_{\mathsf{PRF}}(t(\lambda)) + \mu_{\mathsf{OWF}}(t(\lambda)/2)).$$

We are now ready to prove Theorem 4.15.

Proof of Theorem 4.15. Assume a $(2^{2(\log\lambda)^{1/\epsilon}}, 2^{2(\log\lambda)^{1/\epsilon}}, 2^{(\log\lambda)^{1/\epsilon}}, 2^{-(\log\lambda)^{1.5/\epsilon}})$-selectively secure single-input private-key functional encryption scheme for all functions of polynomial size. By Corollary 4.3, for every $\epsilon > 0$, there exists a $\delta > 0$ such that there exists a $(2^{(\log\lambda)^2}, 2^{-(\log\lambda)^2})$-secure indistinguishability obfuscator for the circuit class $\mathcal{C}_\lambda^{2^{O((\log\lambda)^\epsilon)}, (\log\lambda)^{1+\delta}}$.

Assume a $(2^{2\lambda^{\epsilon'}}, 2^{-2\lambda^{\epsilon'}})$-secure (injective) one-way function for some large enough constant ϵ' such that $(1+\delta/2)/(1+\delta) \leq \epsilon' < 1$. Thus, there exists a $(2^{\lambda^{\epsilon'}}, 2^{-\lambda^{\epsilon'}})$-secure puncturable pseudorandom function family.

Recall that $T(\lambda) = 2^{(\log\lambda)^{1+\delta/3}}$ and $t(\lambda) = (\log\lambda)^{1+\delta}$. Plugging these primitives in Lemma 4.8 we get that for some (small enough) constants $\zeta', \zeta'' \in (0,1)$, every adversary that runs in time

$$\min\{T_{i\mathcal{O}}(\lambda), T_{\mathsf{PRF}}(t(\lambda)), T_{\mathsf{OWF}}(t(\lambda)/2)\} \geq 2^{(\log\lambda)^{1+\zeta'}},$$

its probability of solving the SVL problem is at most

$$\begin{aligned}
&\mu_{\mathsf{OWF}}(\log T(\lambda)) + T(\lambda) \cdot (\mu_{i\mathcal{O}}(\lambda) + \mu_{\mathsf{PRF}}(t(\lambda)) + \mu_{\mathsf{OWF}}(t(\lambda)/2)) \\
&\quad = \mu_{\mathsf{OWF}}((\log\lambda)^{1+\delta/3}) + 2^{(\log\lambda)^{1+\delta/3}} \\
&\qquad \cdot (2^{-(\log\lambda)^2} + \mu_{\mathsf{PRF}}((\log\lambda)^{1+\delta}) + \mu_{\mathsf{OWF}}((\log\lambda)^{1+\delta}/2))
\end{aligned}$$

$$\leq 2^{-2(\log\lambda)^{(1+\delta/3)\cdot\epsilon'}} + 2^{(\log\lambda)^{1+\delta/3}} \cdot (2^{-(\log\lambda)^2} + 2^{-(\log\lambda)^{(1+\delta)\cdot\epsilon'}} + 2^{-2(\log\lambda)^{(1+\delta)\cdot\epsilon'}/2^{\epsilon'}})$$
$$\leq 2^{-(\log\lambda)^{1+\zeta''}}.$$

□

Acknowledgements

We thank Zvika Brakerski and the anonymous referees for many valuable comments. The first author thanks his advisor Moni Naor for his support and guidance.

References

[1] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, A. Sahai, Function private functional encryption and property preserving encryption: new definitions and positive results. Cryptology ePrint Archive, Report 2013/744 (2013)

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

[3] P. Ananth, Z. Brakerski, G. Segev, V. Vaikuntanathan, From selective to adaptive security in functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 657–677

[4] P. Ananth, A. Jain, Indistinguishability obfuscation from compact functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 308–326

[5] P. Ananth, A. Jain, M. Naor, A. Sahai, E. Yogev, Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption, in *Advances in Cryptology—CRYPTO '16* (2016), pp. 491–520

[6] P. Ananth, A. Jain, A. Sahai, Achieving compactness generically: indistinguishability obfuscation from non-compact functional encryption. Cryptology ePrint Archive, Report 2015/730 (2015)

[7] T. Abbot, D. Kane, P. Valiant, On algorithms for Nash equilibria (2004)

[8] G. Asharov, G. Segev, Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM J. Comput.*, **45**(6), 2117–2176 (2016)

[9] E. Boyle, K. Chung, R. Pass, On extractability obfuscation, in *Proceedings of the 11th Theory of Cryptography Conference, TCC* (2014), pp. 52–73

[10] Z. Brakerski, C. Gentry, S. Halevi, T. Lepoint, A. Sahai, M. Tibouchi, Cryptanalysis of the quadratic zero-testing of GGH. Cryptology ePrint Archive, Report 2015/845 (2015)

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

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

[13] Z. Brakerski, I. Komargodski, G. Segev, Multi-input functional encryption in the private-key setting: stronger security from weaker assumptions. *J. Cryptol.*, **31**(2), 434–520 (2018)

[14] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, J. Zimmerman, Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation, in *Advances in Cryptology—EUROCRYPT '15* (2015), pp. 563–594

[15] N. Bitansky, R. Nishimaki, A. Passelègue, D. Wichs, From Cryptomania to Obfustopia through secret-key functional encryption, in *Theory of Cryptography—14th International Conference, TCC 2016-B* (2016), pp. 391–418

[16] N. Bitansky, O. Paneth, A. Rosen, On the cryptographic hardness of finding a Nash equilibrium, in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science* (2015), pp. 1480–1498

[17] D. Boneh, A. Raghunathan, G. Segev, Function-private identity-based encryption: hiding the function in functional encryption, in *Advances in Cryptology—CRYPTO '13* (2013), pp. 461–478
[18] D. Boneh, A. Raghunathan, G. Segev, Function-private subspace-membership encryption and its applications, in *Advances in Cryptology—ASIACRYPT '13* (2013), pp. 255–275
[19] Z. Brakerski, G. Segev, Function-private functional encryption in the private-key setting, in *Proceedings of the 12th Theory of Cryptography Conference, TCC* (2015), pp. 306–324
[20] D. Boneh, A. Sahai, B. Waters, Functional encryption: definitions and challenges, in *Proceedings of the 8th Theory of Cryptography Conference, TCC* (2011), pp. 253–273
[21] D. Boneh, A. Sahai, B. Waters, Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, **55**(11), 56–64 (2012)
[22] N. Bitansky, V. Vaikuntanathan, Indistinguishability obfuscation from functional encryption, in *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science* (2015), pp. 171–190
[23] D. Boneh, B. Waters, Constrained pseudorandom functions and their applications, in *Advances in Cryptology—ASIACRYPT '13* (2013), pp. 280–300
[24] X. Chen, X. Deng, S. Teng, Settling the complexity of computing two-player Nash equilibria. *J. ACM*, **56**(3), 14 (2009)
[25] J.H. Cheon, P. Fouque, C. Lee, B. Minaud, H. Ryu, Cryptanalysis of the new CLT multilinear map over the integers, in *Advances in Cryptology—EUROCRYPT* (2016), pp. 509–536
[26] J. Coron, C. Gentry, S. Halevi, T. Lepoint, H.K. Maji, E. Miles, M. Raykova, A. Sahai, M. Tibouchi, Zeroizing without low-level zeroes: new MMAP attacks and their limitations, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 247–266
[27] J.H. Cheon, K. Han, C. Lee, H. Ryu, D. Stehlé, Cryptanalysis of the multilinear map over the integers, in *Advances in Cryptology—EUROCRYPT '15* (2015), pp. 3–12
[28] J.H. Cheon, J. Jeong, C. Lee, An algorithm for NTRU problems and cryptanalysis of the GGH multilinear map without an encoding of zero. Cryptology ePrint Archive, Report 2016/139 (2016)
[29] C. Daskalakis, P. W. Goldberg, C.H. Papadimitriou, The complexity of computing a Nash equilibrium. *Commun. ACM*, **52**(2), 89–97 (2009)
[30] C. Daskalakis, P.W. Goldberg, C.H. Papadimitriou, The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, **39**(1), 195—259 (2009)
[31] C. Daskalakis, C.H. Papadimitriou, Continuous local search, in *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms* (2011), pp. 790–804
[32] S. Goldwasser, S.D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, H.-S. Zhou, Multi-input functional encryption, in *Advances in Cryptology—EUROCRYPT '14* (2014), pp. 578–602
[33] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters, Candidate indistinguishability obfuscation and functional encryption for all circuits, in *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science* (2013), pp. 40–49
[34] S. Garg, C. Gentry, S. Halevi, M. Zhandry, Functional encryption without obfuscation, in *Proceedings of the 13th Theory of Cryptography Conference, TCC* (2016), pp. 480–511
[35] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *J. ACM*, **33**(4), 792-807 (1986)
[36] S. Goldwasser, Y. Kalai, R.A. Popa, V. Vaikuntanathan, N. Zeldovich, Reusable garbled circuits and succinct functional encryption, in *Proceedings of the 45th Annual ACM Symposium on Theory of Computing* (2013), pp. 555–564
[37] S. Garg, O. Pandey, A. Srinivasan, Revisiting the cryptographic hardness of finding a Nash equilibrium, in *Advances in Cryptology—CRYPTO '16* (2016), pp. 579–604
[38] S. Garg, A. Srinivasan, Single-key to multi-key functional encryption with polynomial loss, in *Theory of Cryptography—14th International Conference, TCC* (2016), pp. 419–442
[39] S. Gorbunov, V. Vaikuntanathan, H. Wee, Functional encryption with bounded collusions via multi-party computation, in *Advances in Cryptology—CRYPTO '12* (2012), pp. 162–179
[40] Y. Hu, H. Jia, Cryptanalysis of GGH map, in *Advances in Cryptology—EUROCRYPT* (2016), pp. 537–565
[41] P. Hubácek, E. Yogev, Hardness of continuous local search: Query complexity and cryptographic lower bounds, in *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA* (2017), pp. 1352–1371

[42] R. Impagliazzo, A personal view of average-case complexity, in *Proceedings of the 10th Annual Structure in Complexity Theory Conference* (1995), pp. 134–147
[43] I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, E. Yogev, One-way functions and (im)perfect obfuscation, in *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science* (2014), pp. 374–383
[44] F. Kitagawa, R. Nishimaki, K. Tanaka, Obfustopia built on secret-key functional encryption, in *Advances in Cryptology—EUROCRYPT* (2018), pp. 603–648
[45] A. Kiayias, S. Papadopoulos, N. Triandopoulos, T. Zacharias, Delegatable pseudorandom functions and applications, in *Proceedings of the 20th Annual ACM Conference on Computer and Communications Security* (2013), pp. 669–684
[46] I. Komargodski, G. Segev, E. Yogev, Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. *J. Cryptol.*, **31**(1), 60–100 (2018)
[47] B. Li, D. Micciancio, Compactness vs collusion resistance in functional encryption, in *Theory of Cryptography—14th International Conference, TCC* (2016), pp. 443–468
[48] E. Miles, A. Sahai, M. Zhandry, Annihilation attacks for multilinear maps: cryptanalysis of indistinguishability obfuscation over GGH13, in *Advances in Cryptology—CRYPTO* (2016), pp. 629–658
[49] A. O'Neill, Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
[50] C.H. Papadimitriou, On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. Syst. Sci.*, **48**(3), 498-532 (1994)
[51] E. Shen, E. Shi, B. Waters, Predicate privacy in encryption systems, in *Proceedings of the 6th Theory of Cryptography Conference, TCC* (2009), pp. 457–473
[52] A. Sahai, B. Waters, Slides on functional encryption (2008). http://www.cs.utexas.edu/~bwaters/presentations/files/functional.ppt
[53] A. Sahai, B. Waters, How to use indistinguishability obfuscation: deniable encryption, and more, in *Proceedings of the 46th Annual ACM Symposium on Theory of Computing* (2014), pp. 475–484
[54] B. Waters, A punctured programming approach to adaptively secure functional encryption, in *Advances in Cryptology—CRYPTO '15* (2015), pp. 678–697