Obfustopia Built on Secret-Key Functional Encryption

Fuyuki Kitagawa[1], Ryo Nishimaki[2], and Keisuke Tanaka[1]

[1] Tokyo Institute of Technology, Tokyo, Japan
{kitagaw1, keisuke}@is.titech.ac.jp
[2] Secure Platform Laboratories, NTT Corporation, Tokyo, Japan
nishimaki.ryo@lab.ntt.co.jp

Abstract. We show that indistinguishability obfuscation (IO) for all circuits can be constructed solely from secret-key functional encryption (SKFE). In the construction, SKFE need to be able to issue a-priori unbounded number of functional keys, that is, collusion-resistant. Our strategy is to replace public-key functional encryption (PKFE) in the construction of IO proposed by Bitansky and Vaikuntanathan (FOCS 2015) with *puncturable SKFE*. Bitansky and Vaikuntanathan introduced the notion of puncturable SKFE and observed that the strategy works. However, it has not been clear whether we can construct puncturable SKFE without assuming PKFE. In particular, it has not been known whether puncturable SKFE is constructed from ordinary SKFE. In this work, we show that a relaxed variant of puncturable SKFE can be constructed from collusion-resistant SKFE. Moreover, we show that the relaxed variant of puncturable SKFE is sufficient for constructing IO.
In addition, we also study the relation of collusion-resistance and succinctness for SKFE. Functional encryption is said to be weakly-succinct if the size of its encryption circuit is sub-linear in the size of functions. We show that collusion-resistant SKFE can be constructed from weakly-succinct SKFE supporting only one functional key.
By combining the above two results, we show that IO for all circuits can be constructed from weakly-succinct SKFE supporting only one functional key.

1 Introduction

1.1 Backgrounds

Program obfuscation is now one of the central topics in cryptography. Program obfuscation aims to turn programs "unintelligible" while preserving its functionality. The theoretical study of program obfuscation was initiated by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [12]. They introduced *virtual-black-box* obfuscation as a formal definition of obfuscation. The definition of virtual black-box obfuscation is intuitive and naturally captures the requirement that obfuscators hide information about programs. However, Barak et al.

showed that it is impossible to achieve virtual black-box obfuscation for all circuits. In order to avoid the impossibility result, they also defined an weaker variant of obfuscation called *indistinguishability obfuscation (IO)*. Impossibility of IO for all circuits is not known.

Garg, Gentry, Halevi, Raykova, Sahai, and Waters [33] proposed the first candidate construction of IO for all circuits. Subsequently, many works have shown that IO is powerful enough in the sense that we can achieve a wide variety of cryptographic primitives based on IO though it is weaker than virtual-black-box obfuscation [33,62,40,18,49,14,26,16,29,39].

While we know the usefulness of IO well, we know very little about how to achieve IO. Although the first candidate construction was demonstrated, we are still at the embryonic stage for constructing IO. All known constructions of IO are based on a little-studied cryptographic tool called multi-linear maps [33,11,24,5,60,65,8,10,34,51,56,4,52,32]. Moreover, security flaws were discovered in some IO constructions [30,58,7,31,28].

Thus, constructing IO based on a standard assumption is still standing as a major open question in the study of cryptography. As a stepping-stone for solving the question, it is important to find a seemingly weaker primitive that implies IO. As such a cryptographic primitive, we already have *functional encryption*.

Functional encryption is one of the most advanced cryptographic primitives which enable a system having flexibility in controlling encrypted data [63,20,59]. In functional encryption, an owner of a master secret key MSK can generate a functional decryption key sk_f for a function f belonging to a function family $\mathcal{F}$. By decrypting a ciphertext of a message m using sk_f, a holder of sk_f can learn only a value $f(m)$. No information about x except $f(m)$ is revealed from the ciphertext of m. This feature enables us to construct a cryptographic system with fine-grained access control. In addition, it is known that functional encryption is a versatile building block to construct other cryptographic primitives. In particular, we can construct IO for all circuits by using functional encryption that satisfies certain security notions and efficiency requirements [2,17,3,15].

Bitansky and Vaikuntanathan [17] and Ananth and Jain [2] independently showed that we can construct IO based on public-key functional encryption (PKFE) which supports a single functional key and whose encryption circuit size is sub-linear in the size of functions. A functional encryption scheme that supports a single key is called a *single-key* scheme. A functional encryption scheme that satisfies the efficiency property above is said to be *weakly-succinct*.

Bitansky, Nishimaki, Passelègue, and Wichs [15] subsequently showed that *collusion-resistant* secret-key functional encryption (SKFE) is powerful enough to yield IO if we additionally assume plain public key encryption. Collusion-resistant functional encryption is functional encryption that can securely issue a-priori unbounded number of functional keys.

From these results, we see that the combination of functional encryption with some property and a public-key cryptographic primitive is sufficient for achieving IO. This fact is a great progress as a stepping-stone for achieving IO based on a standard assumption.

However, one natural question arises for this situation. The question is whether we really need public-key primitives to construct IO or not. In other words, we have the following fundamental question:

Is it possible to achieve IO for all circuits based solely on secret-key primitives?

SKFE is the best possible candidate for a secret-key cryptographic primitive that gives an affirmative answer to this question. However, Asharov and Segev [9] gave a somewhat negative answer to the question. Their result can be seen as a substantial evidence that SKFE is somewhat unlikely to imply IO as long as we use *black-box* techniques.[1] Although Komargodski and Segev [48] already showed that we can construct IO for somewhat restricted class of circuits based on SKFE via non-black-box construction, it is still open whether we can construct IO for all circuits from SKFE bypassing the barrier with a non-black-box technique.

The real power of IO appears in the fact that it can transform secret-key primitives into public-key ones. Therefore, solving the above problem is a key advancement to discover the exact requirements for achieving IO.

1.2 Our Results

We give an affirmative answer to the question above. More precisely, we prove the following theorem.

Theorem 1 (Informal). *Assuming there exists sub-exponentially secure collusion-resistant SKFE for all circuits. Then, there exists IO for all circuits.*

Since our construction of IO is *non-black-box*, we can circumvent the impossibility result shown by Asharov and Segev [9].

The security loss of our construction of IO is exponential in the input length of circuits, but is independent of the size of circuits. Thus, if the input length of circuits is poly-logarithmic in the security parameter, our construction of IO incurs only quasi-polynomial security loss regardless of the size of circuits. Therefore, we can obtain IO for circuits of *polynomial size* with input of poly-logarithmic length from *quasi-polynomially secure* collusion-resistant SKFE for all circuits. This is an improvement over the IO construction by Komargodski and Segev [48]. They showed that IO for circuits of *sub-polynomial size* with input of poly-logarithmic length is constructed from quasi-polynomially secure collusion-resistant SKFE for all circuits.

We show Theorem 1 by using *puncturable SKFE.* The notion of puncturable SKFE was introduced by Bitansky and Vaikuntanathan [17]. They showed that in their construction of IO, the building block PKFE can be replaced with puncturable SKFE. However, it has been an open issue whether we can achieve puncturable SKFE without assuming the existence of PKFE.

[1] More precisely, Asharov and Segev [9] introduced an extended model for black-box reductions to include a limited class of non-black-box reductions into their impossibility results.

We show how to construct puncturable SKFE that is sufficient for constructing IO, based solely on SKFE. More precisely, we show the following theorem.

Theorem 2 (Informal). *Assuming there exists collusion-resistant SKFE for all circuits. Then, there exists single-key weakly-succinct puncturable SKFE for all circuits.*

Note that our definition of puncturable SKFE is slightly different from that proposed by Bitansky and Vaikuntanathan. Our requirement for puncturable SKFE looks weaker than that of Bitansky and Vaikuntanathan. However, they are actually incomparable. In fact, we show that puncturable SKFE defined in this paper is also sufficient for a building block of IO. See Section 2 for the details of the notion of puncturable SKFE and the difference between our definition and that of Bitansky and Vaikuntanathan.

This result makes a progress on the study of IO and functional encryption as we note in the next paragraph.

Impacts on the hierarchy of cryptographic primitives. It is known that we can classify cryptographic primitives into two hierarchies Minicrypt and Cryptomania since the beautiful work of Impagliazzo and Rudich [42] showed that public-key encryption is not implied by one-way functions via black-box reductions. The terminologies, Minicrypt and Cryptomania, were introduced by Impagliazzo [41]. In Minicrypt, one-way functions exist, but public-key encryption does not. In Cryptomania, public-key encryption also exists.

We have recently started to consider a new hierarchy called Obfustopia. Garg, Pandey, Srinivasan, and Zhandry [35] introduced the term Obfustopia, which seems to indicate the "world" where there exists IO. Garg et al. did not give a formal definition of Obfustopia. In this paper, we explicitly define Obfustopia as the "world" where there exists efficient IO for all circuits and one-way functions.[2] It is known that we can construct almost all existing cryptographic primitives which are stronger than public-key encryption by using IO. This is the reason why we consider the new hierarchy beyond Cryptomania.[3]

The landscape of Obfustopia is not clear while those of Minicrypt and Cryptomania are. In particular, we do not know how to construct IO based on standard assumptions. There has been significant effort to find out cryptographic primitives that are in Obfustopia. That is, we have been asking what

[2] Komargodski, Moran, Naor, Pass, Rosen, and Yogev [47] proved that IO implies one-way functions under a mild complexity theoretic assumption. More specifically, the complexity assumption is $\mathsf{NP} \not\subseteq \mathsf{io\text{-}BPP}$, where $\mathsf{io\text{-}BPP}$ is the class of languages that is decided by probabilistic polynomial-time algorithms for infinitely many input sizes. Therefore, under the assumption, we say that Obfustopia is the complexity spectrum where efficient IO for all circuits exists.

[3] Strictly speaking, it was known that there are stronger primitives than public-key encryption before the candidate of obfuscation appeared. For example, public-key encryption does not imply identity-based encryption [19].

kind of cryptographic primitive implies the existence of IO. We know that sub-exponentially-secure succinct PKFE exists in Obfustopia [17,2].

It is natural to ask whether SKFE is also in Obfustopia or not since SKFE seems to be a strong primitive as PKFE. Asharov and Segev [9] gave a somewhat negative answer to this question. They showed that SKFE is unlikely to imply IO as long as we use black-box techniques. They also showed that SKFE does not imply any primitive in Cryptomania via black-box reductions. Moreover, it was not known whether SKFE implies any primitive outside Minicrypt even if we use it in a non-black-box manner before the work of Bitansky et al. [15].

Bitansky et al. showed that the combination of sub-exponentially secure collusion-resistant SKFE and exponentially secure one-way functions implies quasi-polynomially secure public-key encryption. This also implies that the above combination yields quasi-polynomially secure succinct PKFE from their main result showing that the combination of collusion-resistant SKFE and public-key encryption implies succinct PKFE.

Komargodski and Segev [48] showed that quasi-polynomially secure IO for circuits of sub-polynomial size with input of poly-logarithmic length can be constructed from quasi-polynomially secure collusion-resistant SKFE for all circuits. In addition, they showed that by combining quasi-polynomially secure collusion-resistant SKFE and sub-exponentially secure one-way functions, we can construct quasi-polynomially secure succinct PKFE. However, in this construction, the resulting PKFE supports only circuits of sub-polynomial size with input of poly-logarithmic length though the building block SKFE supports all polynomial size circuits.

These two results surely demonstrated that SKFE is stronger than we thought. Nevertheless, we see that both two results involve degradation of security level or functionality. Thus, it is still open whether SKFE implies a cryptographic primitive other than those in Minicrypt without such degradation, and especially SKFE is in Obfustopia or not.

We gives an affirmative answer to this question. More concretely, we can construct sub-exponentially secure IO for all circuits from sub-exponentially secure collusion-resistant SKFE for all circuits through our transformation by setting security parameter appropriately. This result means that sub-exponentially secure collusion-resistant SKFE exists in Obfustopia. In addition, by combining this result and the result by Garg et al. [33], we see that the existence of sub-exponentially secure collusion-resistant PKFE for all circuits is equivalent to that of sub-exponentially secure collusion-resistant SKFE for all circuits.

Collusion-resistance versus succinctness for SKFE. We also study the relation of collusion-resistance and succinctness for SKFE.

Collusion-resistance and succinctness for functional encryption are seemingly incomparable notions and implications between them are non-trivial. Therefore, it is also a major concern whether we can transform a scheme satisfying one of the two properties into a collusion-resistant and succinct one.

Such a transformation is already known for PKFE. Ananth, Jain, and Sahai [3] showed how to construct collusion-resistant and succinct PKFE from collusion-resistant one. In addition, Garg and Srinivasan [36] and Li and Micciancio [50] showed a transformation from single-key weakly-succinct PKFE to collusion-resistant one with polynomial security loss. Their transformations preserve succinctness of the building block scheme.[4] From these results, collusion-resistance and succinctness are equivalent for PKFE.

On the other hand, the situation is different for SKFE. While we know how to construct collusion-resistant and succinct schemes from collusion-resistant ones [3] similarly to PKFE, we do not know how to construct such schemes from succinct ones *even if sub-exponential security loss is permitted.*

As stated above, some recent results including Theorem 1 show that SKFE is a strong cryptographic primitive beyond Minicrypt if we consider non-black-box reductions. However, one natural question arises for this situation. All of those results assume collusion-resistant SKFE as a building block. Thus, while we see that collusion-resistant SKFE is outside Minicrypt, it is still open whether succinct SKFE is also a strong cryptographic primitive beyond Minicrypt since we do not know how to construct collusion-resistant SKFE from succinct one.

Succinctness seems to be as powerful as collusion-resistance from the equivalence of them in the PKFE setting. Therefore, it is natural to ask whether succinct SKFE is also outside Minicrypt. If we have a transformation from succinct SKFE to collusion-resistant one without assuming public-key primitives, we can solve the question affirmatively. Solving the question is an advancement to understand the complexity of SKFE.

We solve the question by showing the following result.

Theorem 3 (Informal). *Assume that there exists quasi-polynomially (resp. sub-exponentially) secure single-key weakly-succinct SKFE for all circuits. Then, there also exists quasi-polynomially (resp. sub-exponentially) secure collusion-resistant SKFE for all circuits.*

We note that our transformation incurs quasi-polynomial security loss. However, we can transform any quasi-polynomially secure single-key weakly-succinct SKFE into quasi-polynomially secure collusion-resistant one, if we know the security bound of the underlying single-key SKFE. In addition, if the underlying single-key scheme is sub-exponentially secure, then so does the resulting collusion-resistant one.[5]

Our transformation preserves the succinctness of the underlying scheme. In other words, if the building block single-key scheme is succinct (resp. weakly succinct), the resulting collusion-resistant scheme is also succinct (resp. weakly succinct).

[4] The resulting scheme of the transformation proposed by Garg and Srinivasan is succinct even if the building block scheme is only weakly-succinct. The transformation proposed by Li and Micciancio preserves succinctness of the building block scheme.

[5] When transforming a sub-exponentially secure scheme, our transformation incurs sub-exponentially security loss. However, we can transform any sub-exponentially secure single-key scheme into a sub-exponentially secure collusion-resistant one.

Analogous to PKFE, we can transform collusion-resistant SKFE into collusion-resistant and succinct one [3]. From this fact and Theorem 3, we discover that the existence of collusion-resistant SKFE and that of succinct one are actually equivalent if we allow quasi-polynomial security loss. Due to this equivalence, we see that succinct SKFE is also a strong cryptographic primitive beyond Minicrypt similarly to collusion-resistant SKFE. Especially, we obtain the following corollary from Theorem 1 and 3.

Corollary 1 (Informal). *Assume that there exists sub-exponentially secure single-key weakly-succinct SKFE for all circuits. Then, there exists IO for all circuits.*

From this result, we can remove the learning with errors (LWE) assumption from recent state-of-the-art constructions of IO based on multi-linear maps and (block-wise) local pseudorandom generators [52,55].

These works first construct single-key weakly-succinct *SKFE* based on multi-linear maps and (block-wise) local pseudorandom generators. Then, assuming the LWE assumption, they transform it into IO using the result by Bitansky et al. [15]. By relying on Corollary 1 instead of the result by Bitansky et al. [15] in their construction, we can obtain IO based only on multi-linear maps and (block-wise) local pseudorandom generators.

1.3 Organization

We provide the overview of this work using the majority of the remaining part of this paper. For Theorem 1, we show only constructions and omit its security proofs. See [45] for those omitted proofs. For Theorem 3, we provide only the overview. See [44] for details of this result. The detailed organization is as follows.

In Section 2, we provide the overview of our construction of IO based on collusion-resistant SKFE via puncturable SKFE. In Section 3, we also provide the overview of how collusion-resistant SKFE is constructed based on weakly-succinct SKFE. In Section 4, we provide notations and definitions of cryptographic primitives. In Section 5, we formally define puncturable SKFE, and introduce security and efficiency notions for it. We also discuss the difference between our definition of puncturable SKFE and that of Bitansky and Vaikuntanathan [17] in Section 5. In Section 6, we show the construction of single-key non-succinct puncturable SKFE. In Section 7, we show how to transform single-key non-succinct puncturable SKFE into single-key weakly-succinct one. In Section 8, we then show how to construct IO based on SKFE.

2 Overview: IO from Collusion-Resistant SKFE

We give an overview of our construction of IO based on SKFE in this section.

Our basic strategy is to replace PKFE in the construction of Bitansky and Vaikuntanathan [17] with puncturable SKFE. Bitansky and Vaikuntanathan observed that this strategy works. However, it is not known whether puncturable SKFE is constructed from cryptographic primitives other than PKFE or IO.

In this work, we show that we can construct a relaxed variant of puncturable SKFE that is a single-key scheme and weakly-succinct from collusion-resistant SKFE. Moreover, we show that such a relaxed variant of puncturable SKFE is sufficient for constructing IO.

We give an overview of the construction of Bitansky and Vaikuntanathan [17] in Section 2.1 and explain why SKFE must be "puncturable" when we replace PKFE with SKFE in their construction in Section 2.2. Next, we give an overview of how to construct our puncturable SKFE scheme and IO in Section 2.3 and Section 2.4, respectively.

2.1 Construction of IO Based on PKFE

The main idea of Bitansky and Vaikuntanathan is to design an obfuscator $i\mathcal{O}_i$ for circuits with i-bit input from an obfuscator $i\mathcal{O}_{i-1}$ for circuits with $(i-1)$-bit input. If we can design such a bit extension construction, for any polynomial n, we can construct an obfuscator $i\mathcal{O}_n$ for circuits with n-bit input since we can easily achieve $i\mathcal{O}_1$ for circuits with 1-bit input by outputting an entire truth table of a circuit with 1-bit input. If you are familiar with the construction of Bitansky and Vaikuntanathan [17], then you can skip this section.

When we construct IO based on the bit extension construction above, it is important to avoid a circuit-size blow-up of circuits to be obfuscated at each recursive step. In fact, if we allow a circuit-size blow-up, we can obtain the bit extension construction by defining

$$i\mathcal{O}_i(C(x_1 \cdots x_i)) := i\mathcal{O}_{i-1}(C(x_1 \cdots x_{i-1} \| 0)) \| i\mathcal{O}_{i-1}(C(x_1 \cdots x_{i-1} \| 1)) \ .$$

However, this construction obviously incurs an exponential blow-up and thus we cannot rely on this solution. Bitansky and Vaikuntanathan showed how to achieve the bit extension construction without an exponential blow-up using *weakly-succinct* PKFE.

In their construction, a functional key of PKFE should hide information about the corresponding circuit. Such security notion is called function privacy. However, it is not known how to achieve function private PKFE. Then, Bitansky and Vaikuntanathan explicitly accommodated the technique for function private SKFE used by Brakerski and Segev [25] to their IO construction based on PKFE.

We review their construction based on PKFE. For simplicity, we ignore the issue of the randomness for encryption algorithms. It is generated by puncturable pseudorandom function (PRF) in the actual construction.

$i\mathcal{O}_i$ based on $i\mathcal{O}_{i-1}$ and PKFE works as follows. The construction additionally uses plain secret key encryption (SKE) to implement the technique used by Brakerski and Segev [25]. To obfuscate a circuit C with i-bit input, it first generates a key pair $(\mathsf{PK}_i, \mathsf{MSK}_i)$ of PKFE. Then, using MSK_i, it generates a functional key sk_{C^*} tied to the following circuit C^*. C^* has hardwired two SKE ciphertexts $\mathsf{CT}_0^{\mathsf{ske}}$ and $\mathsf{CT}_1^{\mathsf{ske}}$ of plaintext C under independent keys K_0 and K_1, respectively. C^* expects as an input not only an i-bit string $\boldsymbol{x}_i$ but also an SKE key K_b. On those inputs, C^* first obtains C by decrypting

$\mathsf{CT}_b^{\mathsf{ske}}$ by K_b and outputs $U(C, \boldsymbol{x}_i) = C(\boldsymbol{x}_i)$, where $U(\cdot,\cdot)$ is an universal circuit. Finally, the construction obfuscates the following encryption circuit E_{i-1} by $i\mathcal{O}_{i-1}$. E_{i-1} has hardwired PK_i and K_b. On input $(i-1)$-bit string $\boldsymbol{x}_{i-1}$, it outputs ciphertexts $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|0, K_b))$ and $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|1, K_b))$, where Enc is the encryption algorithm of PKFE. The resulting obfuscation of C is a tuple $(\mathsf{sk}_{C^*}, i\mathcal{O}_{i-1}(\mathsf{E}_{i-1}))$. Note that we always set the value of b as 0 in the actual construction. We set b as 1 only in the security proof.

```
// Description of (simplified) $C^*$
Hard-Coded Constants: $\mathsf{CT}_0^{\mathsf{ske}}$, $\mathsf{CT}_1^{\mathsf{ske}}$.
Input: $\boldsymbol{x}_i$, $K_b$

  1. Compute $C = \mathsf{D}(K_b, \mathsf{CT}_b^{\mathsf{ske}})$.
  2. Return $U(C, \boldsymbol{x}_i)$.
```

```
// Description of (simplified) $\mathsf{E}_{i-1}$
Hard-Coded Constants: $\mathsf{PK}_i$, $K_b$.
Input: $\boldsymbol{x}_{i-1} \in \{0,1\}^{i-1}$

  1. Compute $\mathsf{CT}_{i,x_i} \overset{\mathsf{r}}{\leftarrow} \mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|x_i, K_b))$.
  2. Output $\mathsf{CT}_{i,0}$ and $\mathsf{CT}_{i,1}$.
```

When evaluating the obfuscated C on input $\boldsymbol{x}_i = x_1 \cdots x_{i-1}x_i \in \{0,1\}^i$, we first invoke $i\mathcal{O}(\mathsf{E}_{i-1})$ on input $\boldsymbol{x}_{i-1} = x_1 \cdots x_{i-1}$ and obtain $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|0, K_b))$ and $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|1, K_b))$. Then, by decrypting $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1}\|x_i, K_b))$ using sk_{C^*}, we obtain $C(\boldsymbol{x}_i)$.

Consequently, by using this bit extension construction, the obfuscation of a circuit C with n-bit input consists of n functional keys $\mathsf{sk}_1, \cdots, \mathsf{sk}_n$ each of which is generated under a different master secret key MSK_i, and pair of ciphertexts of 0 and 1 under PK_1 corresponding to MSK_1. For any $\boldsymbol{x}_n = x_1 \cdots x_n \in \{0,1\}^n$, we can first compute a ciphertext of $\boldsymbol{x}_n$ by repeatedly decrypting a ciphertext of $\boldsymbol{x}_{i-1} = x_1 \cdots x_{i-1}$ by sk_{i-1} and obtaining a ciphertext of $\boldsymbol{x}_i = x_1 \cdots x_i$ for every $i \in \{2, \cdots, n\}$. We can finally obtain $C(\boldsymbol{x}_n)$ by decrypting the ciphertext of $\boldsymbol{x}_n$ by sk_n.

In this construction, each instance of PKFE needs to issue only one functional key. This is a minimum requirement for functional encryption. However, for efficiency, PKFE in the construction above should satisfy a somewhat strong requirement, that is, weak-succinctness to avoid a circuit-size blow-up of circuits to be obfuscated at each recursive step. Therefore, we need to use a single-key weakly-succinct PKFE scheme in the IO construction above.

We can prove the security of the construction recursively. More precisely, we can prove the security of $i\mathcal{O}_i$ based on those of $i\mathcal{O}_{i-1}$, PKFE, and SKE. Note that it is sufficient that PKFE satisfies a mild selective-security to complete the proof. Their security proof relies on the argument of probabilistic IO formalized by Canneti, Lin, Tessaro, and Vaikuntanathan [27], and thus the security loss of each recursive step is exponential in i, that is 2^i. This is the reason their building block PKFE must be sub-exponentially secure.

2.2 Replacing PKFE with SKFE: Need of Puncturable SKFE

The security proof of Bitansky and Vaikuntanathan relies on the fact that we can use the security of PKFE even when its encryption circuit is publicly available.

Concretely, PK_i is hardwired into obfuscated encryption circuit $i\mathcal{O}_{i-1}(\mathsf{E}_{i-1})$ and this encryption circuit is public when we use the security of PKFE under the key pair $(\mathsf{PK}_i, \mathsf{MSK}_i)$.

The above security argument might not work if ordinary SKFE is used instead of PKFE. This intuition comes from the impossibility result shown by Barak et al. [12]. In fact, Bitansky and Vaikuntanathan showed that it is impossible to instantiate their IO by using SKFE. More precisely, they showed that there exists a secure SKFE scheme such that their transformation results in insecure IO if the SKFE scheme is used as the building block. This is why they adopted PKFE as their building block. Therefore, in order to replace PKFE with SKFE in the construction above, we need SKFE whose security holds even when its encryption circuit is publicly available. As one of such primitives, Bitansky and Vaikuntanathan proposed *puncturable SKFE*.

In puncturable SKFE defined by Bitansky and Vaikuntanathan, there are a puncturing algorithm Punc and a punctured encryption algorithm PEnc in addition to algorithms of ordinary SKFE. We can generate a punctured master secret key $\mathsf{MSK}^*\{m_0, m_1\}$ at two messages m_0 and m_1 from a master secret key MSK by using Punc. Puncturable SKFE satisfies the following two properties: *functionality preserving under puncturing* and *semantic security at punctured point*. Functionality preserving under puncturing requires that

$$\mathsf{Enc}(\mathsf{MSK}, m; r) = \mathsf{PEnc}(\mathsf{MSK}^*\{m_0, m_1\}, m; r)$$

holds for any message m other than m_0 and m_1 and for any randomness r. Semantic security at punctured point requires that

$$(\mathsf{MSK}^*\{m_0, m_1\}, \mathsf{Enc}(\mathsf{MSK}, m_0)) \overset{\mathsf{c}}{\approx} (\mathsf{MSK}^*\{m_0, m_1\}, \mathsf{Enc}(\mathsf{MSK}, m_1))$$

holds for all adversaries, where $\overset{\mathsf{c}}{\approx}$ denotes computational indistinguishability.

Bitansky and Vaikuntanathan showed that single-key weakly-succinct puncturable SKFE is also a sufficient building block for their IO construction while ordinary SKFE is not. Note that weak-succinctness of puncturable SKFE requires that not only the encryption circuit but also the punctured encryption circuit should be weakly-succinct. However, as stated earlier, there was no instantiation of puncturable SKFE other than regarding PKFE as puncturable SKFE at that point. In particular, it was not clear whether we can construct puncturable SKFE based on ordinary SKFE.

2.3 Puncturable SKFE from SKFE

In this work, we show we can construct single-key weakly-succinct puncturable SKFE from collusion-resistant SKFE. More specifically, we show the following two results. First, we show how to construct single-key non-succinct puncturable SKFE based only on one-way functions. In addition, we show that we can transform it into single-key weakly-succinct one using collusion-resistant SKFE. Our formalization of puncturable SKFE is different from that of Bitansky and

Vaikuntanathan [17] in several aspects. Nevertheless, we show that our puncturable SKFE is also sufficient for constructing IO.

Below, we give the overview of these two constructions.

Single-Key Non-Succinct Puncturable SKFE Based on One-Way Functions Our starting point is the SKFE variant of the single-key non-succinct PKFE scheme proposed by Sahai and Seyalioglu [61]. It is constructed from garbled circuit and SKE, which are implied by one-way functions. Their construction is as follows.

Setup: A master secret key consists of $2s$ secret keys $\{K_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$ of SKE, where s is the length of a binary representation of functions supported by the resulting SKFE scheme.

Enc: When we encrypt a message m, we first generates a garbled circuit $\widetilde{U}_m$ with labels $\{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$ by garbling an universal circuit $U(\cdot,m)$ into which m is hardwired. Then, we encrypt $L_{j,\alpha}$ under $K_{j,\alpha}$ and obtain an SKE ciphertext $c_{j,\alpha}$ for every $j\in[s]$ and $\alpha\in\{0,1\}$. The resulting ciphertext of the scheme is $(\widetilde{U}_m,\{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.

KeyGen: A functional key sk_f for a function f consists of $\{K_{j,f[j]}\}_{j\in[s]}$, where $f[1]\cdots f[s]$ is the binary representation of f and each $f[j]$ is a single bit.

Dec: A decryptor who has a ciphertext $(\widetilde{U}_m,\{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$ and a functional key $\{K_{j,f[j]}\}_{j\in[s]}$ can compute $\{L_{j,f[j]}\}_{j\in[s]}$ by decrypting each $c_{j,f[j]}$ by $K_{j,f[j]}$ and obtain $\widetilde{U}_m(\{L_{j,f[j]}\}_{j\in[s]})=U(f,m)=f(m)$.

In the construction above, we observe that if we use puncturable PRF instead of SKE, the resulting scheme is puncturable in some sense. More specifically, a master secret key now consists of $2s$ puncturable PRF keys $\{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$. When we encrypt a message m, we first generate $(\widetilde{U}_m,\{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$ and encrypt each label by using a puncturable PRF value. That is, $c_{j,\alpha}\leftarrow L_{j,\alpha}\oplus\mathsf{F}_{S_{j,\alpha}}(\mathsf{tag})$, where F is puncturable PRF and tag is a public tag chosen in some way.

In this case, we can generate a punctured master secret key $\mathsf{MSK}^*\{\mathsf{tag}\}$ at a tag tag. Thus, we define an encryption algorithm in a tag-based manner. The encryption algorithm Enc, given MSK, tag, and m, outputs a ciphertext of m under the tag tag. That is, $\mathsf{Enc}(\mathsf{MSK},\mathsf{tag},m)=(\widetilde{U}_m,\{L_{j,\alpha}\oplus\mathsf{F}_{S_{j,\alpha}}(\mathsf{tag})\}_{j\in[s],\alpha\in\{0,1\}})$. A punctured master secret key $\mathsf{MSK}^*\{\mathsf{tag}\}$ consists of $2s$ puncturable PRF keys $\{S^*_{j,\alpha}\{\mathsf{tag}\}\}_{j\in[s],\alpha\in\{0,1\}}$ all of which are punctured at tag.

By using $\mathsf{MSK}^*\{\mathsf{tag}\}$, we can generate a ciphertext of any message m under a tag tag' different from tag, that is, $\mathsf{PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\},\mathsf{tag}',m)=(\widetilde{U}_m,\{L_{j,\alpha}\oplus\mathsf{F}_{S^*_{j,\alpha}\{\mathsf{tag}\}}(\mathsf{tag}')\}_{j\in[s],\alpha\in\{0,1\}})$. Then, we have

$$\mathsf{Enc}(\mathsf{MSK},\mathsf{tag}',m;r)=\mathsf{PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\},\mathsf{tag}',m;r)$$

for any tag tag and tag' such that $\mathsf{tag}\neq\mathsf{tag}'$, message m, and randomness r due to the functionality preserving property of puncturable PRF. Namely, this scheme satisfies functionality preserving under puncturing.

In addition, we can prove that $\mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}, m_0)$ and $\mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}, m_1)$ are indistinguishable for adversaries that have $\mathsf{MSK}^*\{\mathsf{tag}\}$ based on the security of puncturable PRF. In other words, it satisfies semantic security at punctured tag.

This formalization is different from that proposed by Bitansky and Vaikuntanathan. Nevertheless, our formalization of puncturable SKFE is sufficient for constructing IO. In fact, when we construct IO, we set the tag same as the message to be encrypted itself. Then, our formalization is conceptually the same as that of Bitansky and Vaikuntanathan. Our tag-based definition is well-suited for our constructions.

Achieving Weak-Succinctness via Collusion-Succinctness We cannot directly use the puncturable SKFE scheme above as a building block of IO since it is non-succinct. We need to transform it into an weakly-succinct scheme while preserving security and functionality.

We extend the work by Kitagawa, Nishimaki, and Tanaka [46] that showed how to transform non-succinct PKFE into weakly-succinct one using collusion-resistant SKFE. They accomplished the transformation via a *collusion-succinct* scheme. We try to accommodate their transformation techniques into the context of puncturable SKFE.

Collusion-succinctness requires that each size of the encryption circuit and punctured encryption circuit is sub-linear in the number of functional keys that the scheme can issue. Note that when we consider collusion-succinctness, the size of these circuits can be polynomial of the size of functions.

We first show that we can construct collusion-succinct puncturable SKFE based on single-key non-succinct puncturable SKFE and collusion-resistant SKFE. Then, we transform the collusion-succinct scheme into an weakly-succinct scheme via a transformation based on decomposable randomized encoding. The latter transformation based on decomposable randomized encoding is similar to that proposed by Bitansky and Vaikuntanathan [17] and that proposed by Ananth, Jain, and Sahai [3]. We give an illustration of our construction path in Figure 1.

The general picture is similar to that of Kitagawa et al. [46] and we can accomplish the latter transformation based on a known technique, but there is a technical hurdle in the former transformation. The most biggest issue is how to define punctured master secret keys and the punctured encryption algorithm. We show the overview of the former transformation and explain the technical hurdle below.

Construction of collusion-succinct scheme. Our goal of this step is to construct a collusion-succinct scheme, that is, a scheme which supports q functional keys and the size of whose encryption and punctured encryption circuits are sub-linear in q, where q is an a-priori fixed polynomial. The key tool for achieving this goal is strong exponentially-efficient IO (SXIO) proposed by Lin, Pass, Seth, and Telang [53].

SXIO is a relaxed variant of IO. SXIO is required that, given a circuit C with n-bit input, it runs in $2^{\gamma n} \cdot \mathrm{poly}(\lambda, |C|)$-time, where γ is a constant smaller

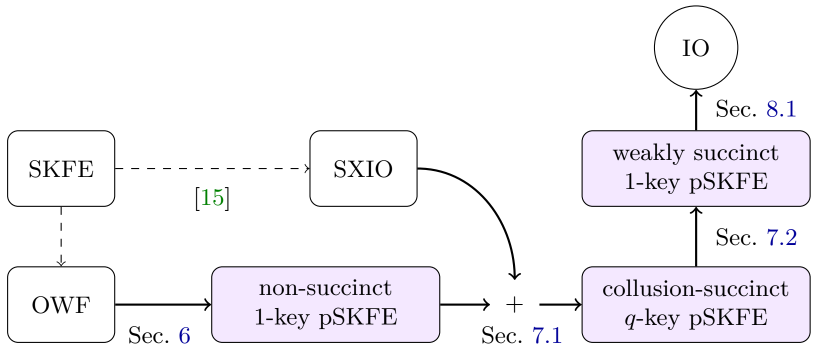

Fig. 1. Illustration of our construction path. pSKFE denotes puncturable SKFE. Dashed lines denote known or trivial implications. White boxes denote our ingredients or goal. Purple boxes denote our core schemes. A transformation from an object in a rectangle to one in a rectangle incurs only polynomial security loss. A transformation from an object in a rectangle to one in a circle incurs super-polynomial security loss.

than 1, poly is some polynomial, and λ is the security parameter. We call γ the compression factor since it represents how SXIO can compress the truth table of the circuit to be obfuscated. SXIO with arbitrarily small constant compression factor can be constructed from collusion-resistant SKFE [15].

We show how to construct collusion-succinct puncturable SKFE from single-key non-succinct one and SXIO. To achieve a collusion-succinct scheme, we need to increase the number of functional keys to some polynomial q while compressing the size of its encryption circuits into sub-linear in q.

The most naive way to increase the number of functional keys is to run multiple instances of the single-key scheme. If we have q master secret keys $\mathsf{MSK}_1, \cdots, \mathsf{MSK}_q$, we can generate q functional keys since we can generate one functional key under each master secret key. In this case, to ensure that we can decrypt a ciphertext using every functional key under different master secret keys MSK_i for every $i \in [q]$, a ciphertext should be composed of q ciphertexts each of which is generated under MSK_i for every $i \in [q]$. In addition, when we generate a punctured master secret key punctured at $\mathtt{tag}$, we generate q punctured master secret keys $\mathsf{MSK}_i^*\{\mathtt{tag}\}$ for every $i \in [q]$ all of which are punctured at $\mathtt{tag}$.

In the naive construction above, we see that if the single-key scheme satisfies functionality preserving under puncturing and semantic security at punctured tag, then so does the resulting scheme since a ciphertext of the resulting scheme consists of only those of the single-key scheme. However, if a ciphertext of the resulting scheme consists of q ciphertexts of the single-key scheme, the encryption time is obviously at least linear in q. Therefore, we cannot construct a collusion-succinct scheme based on this naive idea.

We then consider to compress the encryption time by using SXIO. We extend the technique used in some previous results [53,15,46]. Let $\mathsf{sxi}\mathcal{O}$ be SXIO. We set a ciphertext as a circuit computing q ciphertexts obfuscated by $\mathsf{sxi}\mathcal{O}$ instead of setting it as q ciphertexts themselves. Concretely, we obfuscate the following circuit $\mathsf{E}_{\mathsf{1Key}}$ using $\mathsf{sxi}\mathcal{O}$. $\mathsf{E}_{\mathsf{1Key}}$ has hardwired message m, tag tag, and puncturable PRF key S, and on input $i \in [q]$, it first generates MSK_i pseudorandomly from S and i, and then outputs a ciphertext of m under MSK_i and tag. Note that the

```
Hard-Coded Constants: $S, \mathsf{tag}, m$.        // Description of (simplified) $\mathsf{E}_{\mathsf{1Key}}$
Input: $i \in [q]$

  1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$.
  2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
  3. Return $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \mathsf{tag}, m)$.
```

master secret key of this scheme is now one puncturable PRF key S. In other words, the scheme generates q master secret keys of the single-key scheme from one puncturable PRF key. For the formal description of $\mathsf{E}_{\mathsf{1Key}}$, see Figure 4 in Section 7.1.

The size of $\mathsf{E}_{\mathsf{1Key}}$ is independent of q since $\mathsf{E}_{\mathsf{1Key}}$ consists of one PRF evaluation and setup and encryption procedure of the single-key scheme.[6] Therefore, the time needed to compute $\mathsf{sxi}\mathcal{O}(\mathsf{E}_{\mathsf{1Key}})$ is bounded by $2^{\gamma \log q} \cdot \mathrm{poly}(\lambda, |m|) = q^\gamma \cdot \mathrm{poly}(\lambda, |m|)$ for some constant $\gamma < 1$ and polynomial poly, that is, sub-linear in q. Namely, we succeeds in reducing the encryption time from linear to sub-linear in q.

However, we need more complicated structure to compress the running-time of a punctured encryption algorithm into sub-linear in q. The main reason is that we cannot give master secret key S in the clear in the punctured encryption circuit to reduce the security to that of the building block single-key scheme.

We first argue how to set a punctured master secret key. We cannot rely on the trivial way that sets q punctured master secret keys of the single-key scheme as a punctured master secret key since the size of the punctured encryption circuit becomes linear in q in this trivial way.

Our solution is to set a punctured master secret key as also an obfuscated circuit under SXIO. More precisely, we obfuscate the following circuit $\mathsf{P}_{\mathsf{1Key}}$. $\mathsf{P}_{\mathsf{1Key}}$ has hardwired tag tag and puncturable PRF key S. Note that S is the master secret key thus is the same puncturable PRF key as that hardwired into $\mathsf{E}_{\mathsf{1Key}}$. On input $i \in [q]$, $\mathsf{P}_{\mathsf{1Key}}$ first generates MSK_i pseudorandomly from S and i, and then outputs a punctured master secret key $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ of the single-key scheme. For the formal description of $\mathsf{P}_{\mathsf{1Key}}$, see Figure 5 in Section 7.1.

[6] Strictly speaking, the domain of PRF is $[q]$, and thus the size of $\mathsf{E}_{\mathsf{1Key}}$ depends on q in logarithmic. However, it does not matter since logarithmic factor is absorbed by sub-linear factor. We ignore this issue here for simplicity.

```
// Description of (simplified) $\mathsf{P}_{\mathsf{1Key}}$
Hard-Coded Constants: $S, \mathsf{tag}$.
Input: $i \in [q]$

  1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$.
  2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
  3. Return $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{Punc}(\mathsf{MSK}_i, \mathsf{tag})$.
```

```
// Description of (simplified) $\mathsf{PE}_{\mathsf{1Key}}$
Hard-Coded Constants: $\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{tag}', m$.
Input: $i \in [q]$

  1. Parse $\mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}}) \leftarrow \mathsf{MSK}^*\{\mathsf{tag}\}$.
  2. Compute $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}})(i)$.
  3. Return $\mathsf{CT}_i \leftarrow \mathsf{PEnc}(\mathsf{MSK}^*_i\{\mathsf{tag}\}, \mathsf{tag}', m)$.
```

In addition, we define the punctured encryption algorithm as follows. On input $\mathsf{MSK}^*\{\mathsf{tag}\}$ that is $\mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}})$, tag tag', and message m, the punctured encryption algorithm obfuscates the following circuit $\mathsf{PE}_{\mathsf{1Key}}$ using $\mathsf{sxi}\mathcal{O}$ and outputs the obfuscated circuit. $\mathsf{PE}_{\mathsf{1Key}}$ has hardwired $\mathsf{MSK}^*\{\mathsf{tag}\}$, tag', and m, and on input $i \in [q]$, it first generates the i-th punctured key $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ by feeding i into $\mathsf{MSK}^*\{\mathsf{tag}\} = \mathsf{sxi}\mathcal{O}(\mathsf{PE}_{\mathsf{1Key}})$, and then outputs a ciphertext of m under $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ and tag' using the punctured encryption algorithm of the single-key scheme. If the compression factor of $\mathsf{sxi}\mathcal{O}$ is *sufficiently small*, we ensure that the running time of this punctured encryption algorithm is sub-linear in q. For the formal description of $\mathsf{PE}_{\mathsf{1Key}}$, see Figure 6 in Section 7.1.

We can prove the semantic security at punctured tag by the punctured programming technique proposed by Sahai and Waters [62]. However, the construction above does not satisfy functionality preserving under puncturing. This is because ciphertexts output by the encryption and punctured encryption algorithms are different. The ciphertexts are obfuscation of different circuits $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$, respectively.

In fact, it seems difficult to avoid this problem as long as we use SXIO to gain succinctness. To the best of our knowledge, how to achieve succinctness in a generic way without using SXIO is not known.

Indistinguishability of functionality under puncturing. To overcome the problem above, we introduce a relaxed variant functionality preserving property that is compatible with the construction based on SXIO. We call it *indistinguishability of functionality under puncturing*. Informally speaking, the property requires that

$$(\mathsf{MSK}, \mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}', m)) \overset{\mathsf{c}}{\approx} (\mathsf{MSK}, \mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{tag}', m))$$

holds for any tag tag and tag' such that $\mathsf{tag} \neq \mathsf{tag}'$, and message m, where $\overset{\mathsf{c}}{\approx}$ denotes computational indistinguishability. In other words, it requires that no distinguisher can distinguish ciphertexts output by Enc and PEnc even given both the master secret key and punctured master secret key.

We see that the collusion-succinct construction based on SXIO above satisfies indistinguishability of functionality under puncturing. This comes from the security guarantee of SXIO and the fact that $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$ are functionally equivalent as long as the above tag and tag' are different.

Overall, we can construct collusion-succinct puncturable SKFE with indistinguishability of functionality under puncturing from a single-key non-succinct scheme and SXIO.

Transforming into an weakly-succinct scheme. As stated earlier, we can in turn transform a collusion-succinct scheme into an weakly-succinct one using decomposable randomized encoding. This transformation is based on those proposed by Bitansky and Vaikuntanathan [17] and Ananth et al. [3].

In this transformation, a ciphertext of the weakly-succinct scheme is a ciphertext of the collusion-succinct scheme itself. Thus, if the collusion-succinct scheme satisfies semantic security at punctured tag and indistinguishability of functionality under puncturing, then so does the weakly-succinct scheme. Therefore, we can construct a single-key weakly-succinct puncturable SKFE with indistinguishability of functionality under puncturing.

Indistinguishability of functionality under puncturing looks to be insufficient for constructing IO. Nevertheless, we show that we can replace PKFE in the construction of IO proposed by Bitansky and Vaikuntanathan with our puncturable SKFE that satisfies only indistinguishability of functionality under puncturing if we allow more but asymptotically the same security loss.

2.4 IO from Puncturable SKFE

Finally, we give an overview of our IO construction below.

The construction of IO based on puncturable SKFE is almost the same as that based on PKFE proposed by Bitansky and Vaikuntanathan [17]. It does not depend on which functionality preserving property puncturable SKFE satisfies. Recall that, in their construction, a key pair $(\mathsf{PK}_i, \mathsf{MSK}_i)$ of PKFE is generated and the circuit E_{i-1} that has hardwired PK_i is obfuscated at every recursive step. In our construction based on puncturable SKFE, a master secret key MSK_i of puncturable SKFE is generated and E_{i-1} that has hardwired MSK_i is obfuscated at each recursive step. Concretely, we construct E_{i-1} as a circuit that has hardwired MSK_i and an SKE key K, and on $(i-1)$-bit input $\boldsymbol{x}_{i-1}$, it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K)$ for $x_i \in \{0,1\}$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$, that is, $\mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1} \| x_i, K))$ for $x_i \in \{0,1\}$. In the proof, we replace MSK_i hardwired into E_{i-1} with the tuple of a punctured master secret key $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ punctured at $\boldsymbol{j} \in \{0,1\}^{i-1}$ and a ciphertext of $(\boldsymbol{j} \| x_i, K)$ for $x_i \in \{0,1\}$, where $\boldsymbol{j}$ is a string in $\{0,1\}^{i-1}$ that we focus on at that time.

Outline of Security Proof We give an overview of the security proof of IO based on puncturable SKFE. If the building block puncturable SKFE satisfies functionality preserving under puncturing, the security proof is almost the same as that of Bitansky and Vaikuntanathan. However, our puncturable SKFE satisfies only indistinguishability of functionality under puncturing, and thus we need more complicated arguments. The first half of the following overview is similar to that of Bitansky and Vaikuntanathan. The rest is an overview of proofs that we additionally need due to indistinguishability of functionality under puncturing.

Analogous to IO based on PKFE, we can accomplish this proof recursively. More precisely, we can prove the security of $i\mathcal{O}_i$ based on those of $i\mathcal{O}_{i-1}$, puncturable SKFE, and plain SKE. We proceed the proof as follows. Note again that, we ignore the issue of the randomness for the encryption algorithm and punctured encryption algorithm for simplicity. It is generated by puncturable PRF in the actual construction.

Suppose that we have two functionally equivalent circuits C_0 and C_1 both of which expect an i-bit input. We show that no efficient distinguisher $\mathcal{D}$ can distinguish $i\mathcal{O}_i(C_0)$ and $i\mathcal{O}_i(C_1)$. We consider the following sequence of hybrid experiments. Below, for two hybrids $\mathcal{H}$ and $\mathcal{H}'$, we write $\mathcal{H} \sim \mathcal{H}'$ to denote that the behavior of $\mathcal{D}$ does not change between $\mathcal{H}$ and $\mathcal{H}'$.

In the first hybrid $\mathcal{H}_0$, $\mathcal{D}$ is given $i\mathcal{O}_i(C_0)$. Recall that $i\mathcal{O}_i(C_0)$ consists of sk_{C^*} and $i\mathcal{O}_{i-1}(\mathsf{E}_{i-1})$. C^* has hardwired two SKE ciphertexts $\mathsf{CT}_0^{\mathsf{ske}}$ and $\mathsf{CT}_1^{\mathsf{ske}}$ of C_0 under independent keys K_0 and K_1. On i-bit input $\boldsymbol{x}_i$ and SKE key K_b, C^* first obtains C by decrypting $\mathsf{CT}_b^{\mathsf{ske}}$ by K_b and outputs $C(\boldsymbol{x}_i)$.

In the next hybrid $\mathcal{H}_1$, we change how $\mathsf{CT}_1^{\mathsf{ske}}$ hardwired in C^* is generated. Concretely, we generate $\mathsf{CT}_1^{\mathsf{ske}}$ as a ciphertext of C_1 under the key K_1. It holds that $\mathcal{H}_0 \sim \mathcal{H}_1$ due to the security of SKE. Then, in the next hybrid $\mathcal{H}_2$, we change the circuit E_{i-1} so that, on $(i-1)$-bit input $\boldsymbol{x}_{i-1}$, it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K_1)$ instead of $(\boldsymbol{x}_{i-1} \| x_i, K_0)$ for $x_i \in \{0,1\}$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$.

If we prove $\mathcal{H}_1 \sim \mathcal{H}_2$, we also prove $\mathcal{H}_0 \sim \mathcal{H}_2$ and almost complete the security proof. This is because we can argue that the behavior of $\mathcal{D}$ does not change between $\mathcal{H}_2$ and the hybrid where $\mathcal{D}$ is given $i\mathcal{O}_i(C_1)$ by a similar argument for $\mathcal{H}_0 \sim \mathcal{H}_2$.

Therefore, the main part of the proof is how we change the circuit E_{i-1} from encrypting K_0 in $\mathcal{H}_1$ to encrypting K_1 in $\mathcal{H}_2$. As mentioned earlier, we accomplish this task by relying on the argument of probabilistic IO formalized by Canneti et al. [27].

Concretely, we consider $2^{i-1}+1$ intermediate hybrid experiments $\mathcal{H}_{1,j}$ for $j \in \{0, \cdots, 2^{i-1}\}$ between $\mathcal{H}_1$ and $\mathcal{H}_2$. Between $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j+1}$, we change E_{i-1} so that on input $\boldsymbol{j} \in \{0,1\}^{i-1}$, it outputs ciphertexts of $(\boldsymbol{j} \| x_i, K_1)$ instead of $(\boldsymbol{j} \| x_i, K_0)$ for $x_i \in \{0,1\}$, where $\boldsymbol{j}$ is the binary representation of j. More precisely, we construct E_{i-1} in $\mathcal{H}_{1,j}$ as follows. E_{i-1} has hardwired MSK_i, K_0, and K_1. On $(i-1)$-bit input $\boldsymbol{x}_{i-1}$,

- if $\boldsymbol{x}_{i-1} < \boldsymbol{j}$, it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K_1)$ for $x_i \in \{0,1\}$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$.
- Otherwise, it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K_0)$ for $x_i \in \{0,1\}$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$.

We see that E_{i-1} in $\mathcal{H}_1$ has the same functionality as E_{i-1} in $\mathcal{H}_{1,0}$. In addition, E_{i-1} in $\mathcal{H}_2$ has the same functionality as E_{i-1} in $\mathcal{H}_{1,2^{i-1}}$. Therefore, we have $\mathcal{H}_1 \sim \mathcal{H}_{1,0}$ and $\mathcal{H}_2 \sim \mathcal{H}_{1,2^{i-1}}$ from the security guarantee of $i\mathcal{O}_{i-1}$.

We show how to prove $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j+1}$. For simplicity, we first assume that puncturable SKFE satisfies functionality preserving under puncturing. In this case, we show $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j+1}$ by the following three steps.

(1) In the first step, we hardwire ciphertexts of $(\boldsymbol{j} \| x_i, K_0)$ under MSK_i and a tag $\boldsymbol{j}$ for $x_i \in \{0,1\}$ in E_{i-1}. In addition, we replace hardwired MSK_i in

E_{i-1} with $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ that is a master secret key punctured at a tag $\boldsymbol{j}$. On $(i-1)$-bit input $\boldsymbol{x}_{i-1}$,

- if $\boldsymbol{x}_{i-1} = \boldsymbol{j}$, E_{i-1} outputs hardwired ciphertexts of $(\boldsymbol{j}\|x_i, K_0)$ for $x_i \in \{0,1\}$.
- if $\boldsymbol{x}_{i-1} \neq \boldsymbol{j}$, it generates ciphertexts of $(\boldsymbol{x}_{i-1}\|x_i, K_\beta)$ under $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ and a tag $\boldsymbol{x}_{i-1}$ and outputs them, where $\beta = 1$ if $\boldsymbol{x}_{i-1} < \boldsymbol{j}$ and $\beta = 0$ otherwise.

We see that this change does not affect the functionality of E_{i-1} if puncturable SKFE satisfies functionality preserving under puncturing. Thus, this step is done by the security of $i\mathcal{O}_{i-1}$.

(2) In the second step, we change the hardwired ciphertexts to ciphertexts of $(\boldsymbol{j}\|x_i, K_1)$ for $x_i \in \{0,1\}$. This is done by the semantic security at punctured tag of puncturable SKFE.

(3) In the final step, we change E_{i-1} so that it does not have hardwired ciphertexts of $(\boldsymbol{j}\|x_i, K_1)$ for $x_i \in \{0,1\}$. Moreover, we change E_{i-1} so that E_{i-1} has hardwired MSK_i and use it to generate the output ciphertexts. This change also does not affect the functionality of E_{i-1}, and thus we can accomplish this step by relying on the security of $i\mathcal{O}_{i-1}$ again.

From the above, if puncturable SKFE satisfies functionality preserving under puncturing, we have $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j+1}$ for every $j \in \{0, \cdots, 2^{i-1}-1\}$. By combining $\mathcal{H}_1 \sim \mathcal{H}_{1,0}$ and $\mathcal{H}_{1,2^{i-1}} \sim \mathcal{H}_2$, we obtain $\mathcal{H}_1 \sim \mathcal{H}_2$.

Therefore, we complete the entire proof. In fact, in this case, the proof is essentially the same as that for the case where PKFE is used as a building block shown by Bitansky and Vaikuntanathan.

Additional hybrids for the case of indistinguishability of functionality under puncturing. Recall that our puncturable SKFE satisfies only indistinguishability of functionality under puncturing. Thus, the above argument for steps 1 and 3 do not work straightforwardly. This is because if puncturable SKFE satisfies only indistinguishability of functionality under puncturing, the functionality of E_{i-1} might change at each step of 1 and 3. Therefore, we cannot directly use the security of $i\mathcal{O}_{i-1}$.

Nevertheless, even if puncturable SKFE satisfies only indistinguishability of functionality under puncturing, we can proceed steps 1 and 3 by introducing more additional hybrids. Since steps 1 and 3 are symmetric, we focus on proceeding the step 1. We can apply the following argument for the step 3. Below, we let $\mathcal{H}_{1,j}^0$ denote the hybrid experiment after applying the step 1 to $\mathcal{H}_{1,j}$.

To accomplish the step 1, we introduce the additional intermediate hybrids $\mathcal{H}_{1,j,k}$ for every $k \in \{0, \cdots, 2^{i-1}\} \setminus \{j\}$ between $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j}^0$. Between $\mathcal{H}_{1,j,k}$ and $\mathcal{H}_{1,j,k+1}$, we change E_{i-1} so that, on input $\boldsymbol{k} \in \{0,1\}^{i-1}$, it outputs ciphertexts under $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ instead of ciphertexts under MSK_i, where $\boldsymbol{k}$ is the binary representation of k. More precisely, we construct E_{i-1} in $\mathcal{H}_{1,j,k}$ as follows. E_{i-1} has hardwired $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ in addition to MSK_i, K_0, and K_1. On $(i-1)$-bit input $\boldsymbol{x}_{i-1}$, it runs as follows.

- If $\boldsymbol{x}_{i-1} < \boldsymbol{j}$, it sets $\beta = 1$ and $\beta = 0$ otherwise.

- If $\boldsymbol{x}_{i-1} < k$ and $\boldsymbol{x}_{i-1} \neq j$, it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K_\beta)$ under $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ and a tag $\boldsymbol{x}_{i-1}$, that is, $\mathsf{PEnc}\left(\mathsf{MSK}_i^*\{\boldsymbol{j}\}, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1} \| x_i, K_\beta)\right)$ for $x_i \in \{0,1\}$.
- Otherwise ($\boldsymbol{x}_{i-1} \geq k$ or $\boldsymbol{x}_{i-1} = j$), it outputs a ciphertext of $(\boldsymbol{x}_{i-1} \| x_i, K_\beta)$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$, that is, $\mathsf{Enc}\left(\mathsf{MSK}_i, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1} \| x_i, K_\beta)\right)$ for $x_i \in \{0,1\}$.

We see that E_{i-1} in $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j}^0$ have the same functionality as that in $\mathcal{H}_{1,j,0}$ and $\mathcal{H}_{1,j,2^{i-1}}$, respectively. In addition, E_{i-1} in $\mathcal{H}_{1,j,j}$ has the same functionality as that in $\mathcal{H}_{1,j,j+1}$. Therefore, we have $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j,0}$, $\mathcal{H}_{1,j}^0 \sim \mathcal{H}_{1,j,2^{i-1}}$, and $\mathcal{H}_{1,j,j} \sim \mathcal{H}_{1,j,j+1}$ from the security guarantee of $i\mathcal{O}_{i-1}$.

We can prove $\mathcal{H}_{1,j,k} \sim \mathcal{H}_{1,j,k+1}$ for every $k \in \{0, \cdots, 2^{i-1}\} \setminus \{j\}$ by three steps again based on indistinguishability of functionality under puncturing.

(1) We hardwire ciphertexts of $(\boldsymbol{k} \| x_i, K_\beta)$ under MSK_i and a tag $\boldsymbol{k}$, that is, $\mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{k}, (\boldsymbol{k} \| x_i, K_\beta))$ for $x_i \in \{0,1\}$ in E_{i-1} in the first step. In addition, we change E_{i-1} so that it outputs the hardwired ciphertext of $(\boldsymbol{k} \| x_i, K_0)$ for $x_i \in \{0,1\}$ if the input is $\boldsymbol{k}$. We see that this change does not affect the functionality of E_{i-1}. Thus, this step is done by the security of $i\mathcal{O}_{i-1}$.

(2) In the second step, we change the hardwired ciphertexts to a ciphertext of $(\boldsymbol{k} \| x_i, K_\beta)$ under $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$, that is $\mathsf{PEnc}(\mathsf{MSK}_i^*\{\boldsymbol{j}\}, \boldsymbol{k}, (\boldsymbol{k} \| x_i, K_\beta))$ for $x_i \in \{0,1\}$. This is done by the indistinguishability of functionality under puncturing of puncturable SKFE.

(3) In the final step, we change E_{i-1} so that it does not have hardwired ciphertexts of $(\boldsymbol{k} \| x_i, K_1)$ for $x_i \in \{0,1\}$. Namely, we change E_{i-1} so that on input $\boldsymbol{k}$, E_{i-1} generates ciphertexts of $\boldsymbol{k}$ under $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ and outputs them. This change does not affect the functionality of E_{i-1}, and thus we can accomplish this step by relying on the security of $i\mathcal{O}_{i-1}$ again.

From these, $\mathcal{H}_{1,j,k} \sim \mathcal{H}_{1,j,k+1}$ holds for every $k \in \{0, \cdots, 2^{i-1}\} \setminus \{j\}$. By combining $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j,0}$, $\mathcal{H}_{1,j}^0 \sim \mathcal{H}_{1,j,2^{i-1}}$, and $\mathcal{H}_{1,j,j} \sim \mathcal{H}_{1,j,j+1}$, we obtain $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j}^0$.

Therefore, we obtain $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j}^0$ even if puncturable SKFE satisfies only indistinguishability of functionality under puncturing. Overall, we can complete the entire security proof.

We note that our security proof incurs more security loss than those of Bitansky and Vaikuntanathan [17] and the case where puncturable SKFE satisfies functionality preserving under puncturing. Our security proof incurs roughly $2^{2 \cdot i}$ security loss while the latter proofs incurs 2^i security loss when we prove the security of $i\mathcal{O}_i$ based on that of $i\mathcal{O}_{i-1}$. Nevertheless, this difference is not an issue in the sense that if the building block primitives are roughly $2^{\Omega(n^2)}$-secure, we can prove the security of our indistinguishability obfuscator, where n is the input length of circuits to be obfuscated. This requirement is the same as that of Bitansky and Vaikuntanathan.

3 Overview: Collusion-Resistant SKFE from Weakly-Succinct One

In this section, we give a high-level overview of our technique for increasing the number of functional decryption keys that an SKFE scheme supports. The basic idea behind our proposed construction is that we combine multiple instances of a functional encryption scheme and use functional decryption keys tied to a function that outputs a re-encrypted ciphertext under a different encryption key. Several re-encryption techniques have been studied in the context of functional encryption [2,17,23,36,50], but we cannot directly use such techniques as we see below.

3.1 First Attempt: Applying Re-encryption Techniques in The Public-Key Setting.

It is natural to try using the techniques in the public-key setting. In particular, it was shown that single-key weakly succinct PKFE implies collusion-resistant PKFE by Garg and Srinivasan [36] and Li and Micciancio [50]. Their techniques are different, but the core idea seems to be the same. Both techniques use functional decryption keys for a re-encryption function that outputs a ciphertext under a different encryption key.

We give more details of the technique by Li and Micciancio since it is our starting point. It is unclear whether the technique by Garg and Srinivasan is applicable in the secret-key setting since it seems that they use plain public-key encryption in an essential way.

The main technical tool of Li and Micciancio is the PRODUCT construction. Given two PKFE schemes, the PRODUCT construction combines them into a new PKFE scheme. The most notable feature of the PRODUCT construction is that the number of functional decryption keys of the resulting scheme is the product of those of the building block schemes. For example, if we have a λ-key PKFE scheme, by combining two instances of it via the PRODUCT construction, we can construct a λ^2-key PKFE scheme, where λ is the security parameter.

By applying the PRODUCT construction k times iteratively, we can construct a λ^k-key PKFE scheme from a λ-key PKFE scheme. Note that we can in turn construct a λ-key PKFE scheme by simply running λ instances of a single-key PKFE scheme in parallel. Moreover, if the underlying single-key scheme is weakly succinct, the running time of the λ^k-key scheme depends only on k instead of λ^k. Thus, by setting $k = \omega(1)$, we can construct a $\lambda^{\omega(1)}$-key PKFE scheme and achieve collusion-resistance from a single-key weakly succinct one.

Li and Micciancio proceeded with the above series of transformations via a stateful variant of PKFE, and thus the resulting collusion-resistant scheme is also a stateful scheme. Therefore, after achieving collusion-resistance, they converted the stateful PKFE scheme into a standard PKFE scheme. For simplicity, we ignore the issue here.

One might think that we can construct a collusion-resistant SKFE scheme from a single-key SKFE scheme by using the PRODUCT construction. However, we encounter several difficulties in the SKFE setting.

The PRODUCT construction involves *the chaining of re-encryption by functional decryption keys* used in many previous works [2,17,23,36]. This technique causes several difficulties when we adopt the PRODUCT construction in the SKFE setting. This is also the reason why the building block single-key PKFE scheme must satisfy (weak) succinctness property.

We now look closer at the technique of Li and Micciancio to see difficulties in the SKFE setting. Let PKFE be a 2-key PKFE scheme. As stated above, for functional key generation in this construction, we need state information called index, which indicates how many functional keys generated so far and which master secret and public key should be used to generate the next functional key. A simplified version of the PRODUCT construction proposed by Li and Micciancio is as follows.

(2×2)-key scheme from 2-key scheme.

Setup: Generates PKFE key pairs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{Setup}(1^\lambda)$ for $i \in [2]$. MPK is the master public key and $(\mathsf{MSK}, \mathsf{MSK}_1, \mathsf{MSK}_2, \mathsf{MPK}_1, \mathsf{MPK}_2)$ is the master secret key of this scheme, respectively. In the actual construction, we maintain $(\mathsf{MPK}_i, \mathsf{MSK}_i)$ for $i \in [2]$ as one PRF key to avoid blow-ups.[7]

Functional Key: For n-th functional key generation, a positive integer $n \in [4]$ is interpreted as a pair of index $(i, j) \in [2] \times [2]$. Generates two keys $\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]} \leftarrow \mathsf{KG}(\mathsf{MSK}, , \mathcal{E}[\mathsf{MPK}_i], i)$ and $\mathsf{sk}^{(i,j)}_f \leftarrow \mathsf{KG}(\mathsf{MSK}_i, f, j)$ where $\mathcal{E}$ is a re-encryption circuit described below. A functional key is $(\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]}, \mathsf{sk}^{(i,j)}_f)$.

Encryption: A ciphertext is $\mathsf{ct}_{\mathsf{pre}} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$.

Decryption: First, applies the decryption algorithm with MPK, that is, $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Dec}(\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]}, \mathsf{ct}_{\mathsf{pre}})$. Next, applies it with MPK_i, $f(m) \leftarrow \mathsf{Dec}(\mathsf{sk}^{(i,j)}_f, \mathsf{ct}_{\mathsf{post}})$.

The description of $\mathcal{E}$ defined at the functional key generation phase is as in the figure below. Re-encryption circuit $\mathcal{E}[\mathsf{MPK}_i]$ takes as an input a message m and outputs $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$ by using a hard-wired master public-key MPK_i.

```
Hard-Coded Constants: $\mathsf{MPK}_i$.          // Description of (simplified) $\mathcal{E}$
Input: $m$

  1. Return $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$.
```

[7] In fact, $(\mathsf{MPK}_i, \mathsf{MSK}_i)$ for $i \in [2]$ are generated at the functional key generation phase by computing $r_i \leftarrow \mathsf{PRF}(K, i)$ and $(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{Setup}(1^\lambda; r_i)$, where K is a PRF key and is stored as a part of the master secret key.

Using the master secret-key MSK_1, we can generate two functional keys $\mathsf{sk}_{f_1}^{1,1}, \mathsf{sk}_{f_2}^{1,2}$ since PKFE is a 2-key scheme. Similarly, we can generate two functional keys using MSK_2. Moreover, since MSK is also a master secret-key of the 2-key scheme, we can generate two functional keys $\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_1]}$ and $\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_2]}$ using MSK at the functional key generation step. By these combinations, we can generate 2×2 keys

$$(\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_1]}, \mathsf{sk}_{f_1}^{1,1}), (\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_1]}, \mathsf{sk}_{f_2}^{1,2}), (\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_2]}, \mathsf{sk}_{f_3}^{2,1}), (\mathsf{sk}_{\mathcal{E}[\mathsf{MPK}_2]}, \mathsf{sk}_{f_4}^{2,2}).$$

This is generalized to the case where the underlying schemes are a p-key scheme and q-key scheme for any p and q. That is, for n-th functional key generation where $n \le p \cdot q$, n is interpreted as $(i,j) \in [p] \times [q]$. Thus, by applying the PRODUCT construction to a λ-key scheme k times iteratively, we can obtain a λ^k-key scheme. Note again that we can construct a λ-key weakly succinct SKFE scheme from a single-key weakly succinct one by simple parallelization.

While such a re-encryption technique is widely used in the context of PKFE, it is difficult to use it directly in the SKFE setting. The main cause of the difficulty is the fact that we have to release a functional key implementing the encryption circuit in which a *master secret key* of an SKFE scheme is hardwired to achieve the re-encryption by functional decryption keys. The fact seems to be a crucial problem for the security proof since sk_f might leak information about f. In the PKFE setting, this issue does not arise since an encryption key is publicly available.

3.2 Second Attempt: Applying Techniques in A Different Context of SKFE.

To solve the above issue, we try using a technique in the secret-key setting but in a different context from the collusion-resistance.

Brakerski, Komargodski, and Segev [23] introduced a new re-encryption technique by functional decryption keys in the context of multi-input SKFE [38]. They showed that we can overcome the difficulty above by using the security notion of function privacy [25].

By function privacy, we can hide the information about a master-secret key embedded in a re-encryption circuit $\mathcal{E}[\mathsf{MSK}^*]$. With their technique, we embed a post-re-encrypted ciphertext $\mathsf{ct}_{\mathsf{post}}$ as a trapdoor into a pre-re-encrypted ciphertext $\mathsf{ct}_{\mathsf{pre}}$ in advance in the simulation for the security proof. By embedding this trapdoor, we can remove MSK^* from the re-encryption circuit $\mathcal{E}$ when we reduce the security of the resulting scheme to that of the underlying scheme corresponding to MSK^*.

Their technique is useful, but it incurs a polynomial blow-up of the running time of the encryption circuit for each application of a construction with the re-encryption procedure by a functional decryption key. This is because it embeds a ciphertext into another ciphertext (we call this nested-ciphertext-embedding).

Such a nest does not occur with the technique of Li and Micciancio in the PKFE setting since a post-re-encrypted ciphertext as a trapdoor is embedded in a functional decryption key. One might think we can avoid the issue of nested-ciphertext embedding by embedding ciphertexts in a functional key. However, this is not the case because the number of ciphertext queries is not a-priori bounded in the secret-key setting.

In fact, we obtain a new PRODUCT construction by accommodating the function privacy and nested-ciphertext-embedding technique to the PRODUCT construction of Li and Micciancio. However, if we use our new PRODUCT construction in a naive way, each application of the new PRODUCT construction incurs a polynomial blow-up of the encryption time. In general, k applications of our new PRODUCT construction with nested-ciphertext-embedding incur a double exponential blow-up $\lambda^{2^{O(k)}}$.

Thus, in a naive way, we can apply our new PRODUCT construction iteratively *only constant times*. This is not sufficient for our goal since we must apply our new PRODUCT construction $\omega(1)$ times to achieve collusion-resistant SKFE.

3.3 Our Solution: Sandwiched Size-shifting.

To solve the difficulty of size blow-up, we propose a new construction technique called "sandwiched size-shifting". In this new technique, we use a *hybrid encryption methodology* to reduce the exponential blow-up of the encryption time caused by our new PRODUCT construction with nested-ciphertext-embedding.

A hybrid encryption methodology is used in many encryption schemes. In particular, Ananth, Brakerski, Segev, and Vaikuntanathan [1] showed that a hybrid encryption construction is useful in designing adaptively secure functional encryption from selectively secure one without any additional assumption. In fact, Brakerski et al. [23] also used a hybrid encryption construction to achieve an input aggregation mechanism for multi-input SKFE.

In this study, we propose a new hybrid encryption construction for functional encryption to *reduce the encryption time* of a functional encryption scheme without any additional assumption. Our key tool is a single-ciphertext collusion-resistant SKFE scheme called 1CT, which is constructed only from one-way functions. The notable features of 1CT are as follows.

1. The size of a master secret key of 1CT is independent of the length of a message to be encrypted.
2. The encryption is fully succinct.
3. The size of a functional decryption key is only linear in the size of a function.

The drawback of 1CT is that we can release *only one ciphertext*. However, this is not an issue for our purpose since a master secret key of 1CT is freshly chosen at each ciphertext generation in our hybrid construction.

1CT is based on a garbled circuit [64]. A functional decryption key is a garbled circuit of f with encrypted labels by a standard secret-key encryption scheme.[8] A ciphertext consists of a randomly masked message and keys of the

[8] Each pair of labels is shuffled by a random masking.

secret-key encryption scheme that corresponds to the randomly masked message. Thus, we can generate only one ciphertext since if two ciphertexts are generated, then labels for both bits are revealed and the security of the garbled circuit is completely broken. Note that $\mathsf{1CT}$ is selectively secure. In fact, this construction is a flipped variant of the single-key SKFE by Sahai and Seyalioglu [61].

We then modify the SKFE variant of the hybrid construction proposed by Ananth et al. [1].[9] We use $\mathsf{1CT}$ as data encapsulation mechanism and a q-key weakly succinct SKFE scheme SKFE as key encapsulation mechanism. In our hybrid construction, the encryption algorithm of SKFE encrypts only short values (concretely, a one-time master secret-key of $\mathsf{1CT}$), which are independent of the length of a message to be encrypted. A one-time encryption key (short and fixed length) of $\mathsf{1CT}$ is encrypted by SKFE.

That is, by this hybrid construction, a real message part is shifted onto $\mathsf{1CT}$, whose ciphertext has the full succinctness property. In other words, we can separate the blow-up due to recursion from nested-ciphertext-embedding part. Therefore, we call our new hybrid construction technique "size-shifting".

The third property of $\mathsf{1CT}$ is also important. The size of a functional key of $\mathsf{1CT}$ affects the encryption time of the hybrid construction. This is because a functional key for f of the hybrid construction consists of a functional key of SKFE for a function G, which generates a functional key of $\mathsf{1CT}$ for f. A simplified description of G is below. Due to the third property of $\mathsf{1CT}$, the

```
Hard-Coded Constants: $f$.                // Description of (simplified) $G$
Input: $\mathsf{1CT.MSK}$

  1. Return $\mathsf{1CT.sk}_f \leftarrow \mathsf{1CT.KG}(\mathsf{1CT.MSK}, f)$.
```

hybrid construction preserves weak succinctness.

Moreover, from the above construction of the key generation algorithm, the number of issuable functional keys of the resulting scheme is minimum of those of building block SKFE and $\mathsf{1CT}$. Therefore, since $\mathsf{1CT}$ is collusion-resistant, if SKFE supports q functional keys, then so does the resulting scheme, where q is any fixed polynomial of λ.

Thus, we can apply the hybrid construction after each application of our new PRODUCT construction, preserving the weak succinctness and the number of functional keys that can be released.

The size-shifting procedure is "sandwiched" by each our new PRODUCT construction. As a result, we can reduce the blow-up of the encryption time after k iterations to $k \cdot \lambda^{O(1)}$ if the underlying single-key scheme is weakly succinct while the naive k iterated applications of our new PRODUCT construction incurs $\lambda^{2^{O(k)}}$ size blow-up. Therefore, we can iterate our new PRODUCT construction

[9] Their goal is to construct an adaptively secure scheme. They used *adaptively secure* single-ciphertext functional encryption that is *non-succinct* as data encapsulation mechanism.

$\omega(1)$ times via the size-shifting and construct a collusion-resistant SKFE scheme based only on a single-key (weakly) succinct SKFE scheme.[10]

Our analysis is highly non-trivial though our transformation consists of relatively simple transformations. We believe that it is better to achieve non-trivial results by using simple techniques than complex ones.

Figure 2 illustrates how to construct our building blocks. An illustration of our sandwiched size-shifting procedure is described in Figure 3.

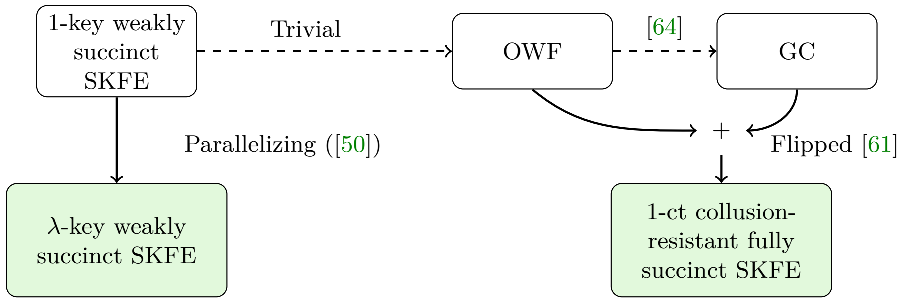
1-key weakly succinct SKFE
Trivial
OWF
[64]
GC
Parallelizing ([50])
+
Flipped [61]
λ-key weakly succinct SKFE
1-ct collusion-resistant fully succinct SKFE

Fig. 2. Our building blocks. Green boxes denote our core schemes used in our iterated construction in Figure 3.

4 Preliminaries

We define some notations and cryptographic primitives.

4.1 Notations

We write $x \xleftarrow{r} X$ to denote that an element x is chosen from a finite set X uniformly at random and $y \leftarrow \mathsf{A}(x;r)$ to denote that the output of an algorithm A on an input x and a randomness r is assigned to y. When there is no need to write the randomness explicitly, we omit it and simply write $y \leftarrow \mathsf{A}(x)$.

Throughout this paper, λ denotes a security parameter. poly denotes an unspecified polynomial. A function $f(\lambda)$ is a negligible function if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. PPT stands for probabilistic polynomial time. Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$.

[10] While we can reduce the blow-up of the encryption time, we cannot reduce the security loss caused by each iteration step. As a result, $\lambda^{\omega(1)}$ security loss occurs after $\omega(1)$ times iterations. This is the reason our transformation incurs quasi-polynomial security loss.

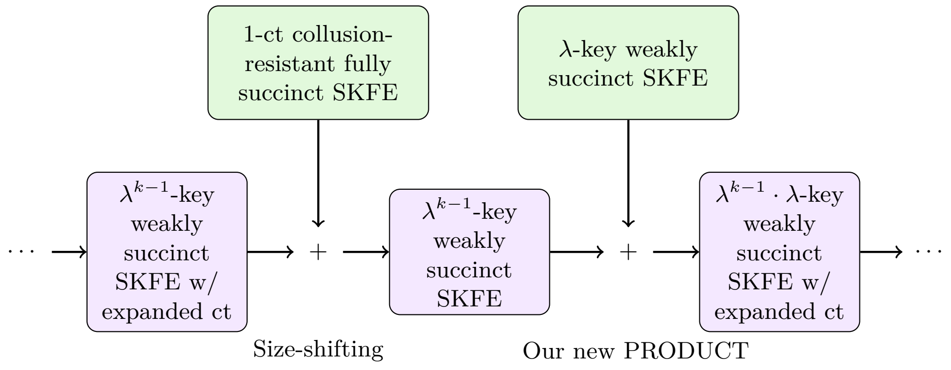
1-ct collision-resistant fully succinct SKFE
λ-key weakly succinct SKFE
λ^{k-1}-key weakly succinct SKFE w/ expanded ct
λ^{k-1}-key weakly succinct SKFE
$\lambda^{k-1} \cdot \lambda$-key weakly succinct SKFE w/ expanded ct
Size-shifting
Our new PRODUCT

Fig. 3. An illustration of our iteration technique, in which our size-shifting procedure is sandwiched. For k-th iteration, first, we apply the size-shifting procedure to a λ^{k-1}-key weakly succinct SKFE scheme with expanded ciphertexts incurred by nested-ciphertext-embedding (the result of $(k-1)$-th iteration). Second, we apply our new PRODUCT construction to increase the number of issuable keys.

4.2 Standard Cryptographic Tools

In this section, we review standard cryptographic tools, pseudorandom function (PRF), puncturable PRF, secret-key encryption (SKE), garbling scheme, and decomposable randomized encoding.

Definition 1 (Pseudorandom functions). *For sets $\mathcal{D}$ and $\mathcal{R}$, let $\{\mathsf{F}_S(\cdot) : \mathcal{D} \to \mathcal{R} | S \in \{0,1\}^\lambda\}$ be a family of polynomially computable functions. We say that F is pseudorandom if for any PPT adversary $\mathcal{A}$, it holds that*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{F},\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}^{\mathsf{F}_S(\cdot)}(1^\lambda) = 1 : S \xleftarrow{\mathsf{r}} \{0,1\}^\lambda] \\ - \Pr[\mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1 : \mathsf{R} \xleftarrow{\mathsf{r}} \mathcal{U}]| = \mathsf{negl}(\lambda) ,$$

where $\mathcal{U}$ is the set of all functions from $\mathcal{D}$ to $\mathcal{R}$. Moreover, for some concrete negligible function $\epsilon(\cdot)$, we say that F is ϵ-secure if for any PPT $\mathcal{A}$ the above indistinguishability gap is smaller than $\epsilon(\lambda)^{\Omega(1)}$.

Theorem 4 ([37]). *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a pseudorandom function that maps $n(\lambda)$ bits to $m(\lambda)$ bits (i.e., $\mathcal{D} := \{0,1\}^{n(\lambda)}$ and $\mathcal{R} := \{0,1\}^{m(\lambda)}$).*

Definition 2 (Puncturable pseudorandom function). *For sets $\mathcal{D}$ and $\mathcal{R}$, a puncturable pseudorandom function PPRF consists of a tuple of algorithms $(\mathsf{F}, \mathsf{Punc})$ that satisfies the following two conditions.*

Functionality preserving under puncturing: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\mathcal{D}$, and for all $x \in \mathcal{D} \setminus \{x_i\}_{i\in[k]}$, we have $\Pr[\mathsf{F}_S(x) = \mathsf{F}_{S^*}(x) : S \leftarrow \{0,1\}^\lambda, S^* \leftarrow \mathsf{Punc}(S, \{x_i\}_{i\in[k]})] = 1$.*

Pseudorandomness at punctured points: *For all polynomial size subset $\{x_i\}_{i\in[k]}$ of $\mathcal{D}$, and any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr[\mathcal{A}(S^*, \{\mathsf{F}_S(x_i)\}_{i\in[k]}) = 1] - \Pr[\mathcal{A}(S^*, U^k) = 1] = \mathsf{negl}(\lambda) \ ,$$

where $S \xleftarrow{r} \{0,1\}^\lambda$, $S^* \leftarrow \mathsf{Punc}(S, \{x_i\}_{i\in[k]})$, and U denotes the uniform distribution over $\mathcal{R}$.
Moreover, for some concrete negligible function $\epsilon(\cdot)$, we say that PPRF *is ϵ-secure if for any $\mathcal{A}$ the above indistinguishability gap is smaller than $\epsilon(\lambda)^{\Omega(1)}$.*

Theorem 5 ([37,21,22,43]). *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a puncturable pseudorandom function that maps $n(\lambda)$ bits to $m(\lambda)$ bits (i.e., $\mathcal{D} := \{0,1\}^{n(\lambda)}$ and $\mathcal{R} := \{0,1\}^{m(\lambda)}$).*

Definition 3 (Secret key encryption). *An SKE scheme* SKE *is a two tuple* (E, D) *of PPT algorithms.*

- *The encryption algorithm* E*, given a key $K \in \{0,1\}^\lambda$ and a message $m \in \mathcal{M}$, outputs a ciphertext c, where $\mathcal{M}$ is the plaintext space of* SKE*.*
- *The decryption algorithm* D*, given a key K and a ciphertext c, outputs a message $\tilde{m} \in \{\bot\} \cup \mathcal{M}$. This algorithm is deterministic.*

Correctness: *We require $\mathsf{D}(K, \mathsf{E}(K, m)) = m$ for every $m \in \mathcal{M}$ and key $K \in \{0,1\}^\lambda$.*

CPA security: *We define the security game between a challenger and an adversary $\mathcal{A}$ as follows.*
1. *The challenger generates $K \xleftarrow{r} \{0,1\}^\lambda$ and chooses the challenge bit $b \xleftarrow{r} \{0,1\}$. Then, the challenger sends 1^λ to $\mathcal{A}$.*
2. *$\mathcal{A}$ may make polynomially many encryption queries adaptively. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$ to the challenger. Then, the challenger returns $c \leftarrow \mathsf{E}(K, m_b)$.*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

In this game, we define the advantage of the adversary $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda) = 2|\Pr[b = b'] - \frac{1}{2}| = |\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \ .$$

For a negligible function $\epsilon(\cdot)$, We say that SKE *is ϵ-secure if for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},\mathcal{A}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.*

Theorem 6 ([54]). *If there exist one-way functions, there exists CPA-secure SKE.*

Definition 4 (Garbling scheme). *Let $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes an n-bit input. A circuit garbling scheme* GC *is a two tuple* $(\mathsf{Grbl}, \mathsf{Eval})$ *of PPT algorithms.*

- *The garbling algorithm* Grbl*, given a security parameter 1^λ and a circuit $C \in \mathcal{C}_n$, outputs a garbled circuit $\widetilde{C}$, together with $2n$ labels $\{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}$.*

– *The evaluation algorithm, given a garbled circuit $\widetilde{C}$ and n labels $\{L_j\}_{j\in[n]}$, outputs y.*

Correctness: *We require $\mathsf{Eval}(\widetilde{C},\{L_{j,x_j}\}_{j\in[n]}) = C(x)$ for every $n\in\mathbb{N}$, $C\in\mathcal{C}_n$, and $x\in\{0,1\}^n$, where $(\widetilde{C},\{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C)$ and x_j is the j-th bit of x for every $j\in[n]$.*

Security: *Let Sim be a PPT simulator. We define the following game between a challenger and an adversary $\mathcal{A}$ as follows.*

1. *The challenger chooses the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends security parameter 1^λ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends a circuit $C\in\mathcal{C}_n$ and an input $x\in\{0,1\}^n$ for the challenger.*
3. *If $b=0$, the challenger computes $(\widetilde{C},\{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C)$ and returns $(\widetilde{C},\{L_{j,x_j}\}_{j\in[n]})$ to $\mathcal{A}$. Otherwise, the challenger returns $(\widetilde{C},\{L_j\}_{j\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))$.*
4. *$\mathcal{A}$ outputs $b'\in\{0,1\}$.*

In this game, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A},\mathsf{Sim}}(\lambda) = 2|\Pr[b=b'] - \frac{1}{2}| = |\Pr[b'=1|b=0] - \Pr[b'=1|b=1]| \ .$$

For a concrete negligible function $\epsilon(\cdot)$, We say that GC is ϵ-secure if there exists a PPT Sim such that for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{gc}}_{\mathsf{GC},\mathcal{A},\mathsf{Sim}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

Theorem 7 ([64,13,57]). *If there exist one-way functions, there exists secure garbling scheme for any polynomial size circuits.*

Definition 5 (Decomposable randomized encoding). *Let $c\geq 1$ be an integer constant. A c-local decomposable randomized encoding RE, given security parameter 1^λ and a function f of size s and n-bit input, outputs a function $\widehat{f}:\{0,1\}^n\times\{0,1\}^\rho\rightarrow\{0,1\}^\mu$ with the following properties. ρ and μ are polynomials bounded by $s\cdot\mathrm{poly}_{\mathsf{RE}}(\lambda,n)$, where $\mathrm{poly}_{\mathsf{RE}}$ is a fixed polynomial.*

Correctness: *There is a polynomial time decoder that, given $\widehat{f}(x;r)$, outputs $f(x)$ for any $x\in\{0,1\}^n$ and $r\in\{0,1\}^\rho$.*

Decomposability: *Computation of $\widehat{f}$ can be decomposed into computation of μ functions. That is, there exist μ functions $\widehat{f}_1,\cdots,\widehat{f}_\mu$ such that $\widehat{f}(x;r) = (\widehat{f}_1(x;r),\cdots,\widehat{f}_\mu(x;r))$. Each $\widehat{f}_i$ depends on a single bit of x at most and c bits of r. We write $\widehat{f}(x;r) = (\widehat{f}_1(x;r_{S_1}),\cdots,\widehat{f}_\mu(x;r_{S_\mu}))$, where S_i denotes the subset of bits of r that $\widehat{f}_i$ depends on.*

Security: *Let Sim be a PPT simulator. We define the following game between a challenger and an adversary $\mathcal{A}$ as follows.*

1. *The challenger chooses a bit $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends security parameter 1^λ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends a function f of size s and n-bit input and an input $x\in\{0,1\}^n$ to the challenger.*

3. *If $b = 0$, the challenger computes $\widehat{f} \leftarrow \mathsf{RE}(1^\lambda, f)$, generates $r \leftarrow \{0,1\}^\rho$, and returns $\widehat{f}(x; r)$ to $\mathcal{A}$. Otherwise, the challenger returns $\mathsf{Sim}(1^\lambda, s, f(x))$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

In this game, we define the advantage of $\mathcal{A}$ as

$$\mathsf{Adv}^{\mathsf{re}}_{\mathsf{RE},\mathsf{Sim},\mathcal{A}}(\lambda) = |\Pr[b' = 1 | b = 0] - \Pr[b' = 1 | b = 1]| \ .$$

For a negligible function $\epsilon(\cdot)$, we say that RE is ϵ-secure if there exists a PPT Sim such that for any PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{re}}_{\mathsf{RE},\mathsf{Sim},\mathcal{A}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

It is known that a decomposable randomized encoding can be based on one-way functions.

Theorem 8 ([64,6]). *If there exist one-way functions, there exists secure decomposable randomized encoding for all polynomial size functions.*

4.3 Secret-Key Functional Encryption

We review the definition of ordinary secret-key functional encryption (SKFE).

Definition 6 (Secret-key functional encryption). *An SKFE scheme SKFE is a four tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. Below, let $\mathcal{M}$ and $\mathcal{F}$ be the message space and function space of SKFE, respectively.*

- *The setup algorithm Setup, given a security parameter 1^λ, outputs a master secret key MSK.*
- *The key generation algorithm KG, given a master secret key MSK and a function $f \in \mathcal{F}$, outputs a functional decryption key sk_f.*
- *The encryption algorithm Enc, given a master secret key MSK and a message $m \in \mathcal{M}$, outputs a ciphertext CT.*
- *The decryption algorithm Dec, given a functional decryption key sk_f and a ciphertext CT, outputs a message $\tilde{m} \in \{\bot\} \cup \mathcal{M}$.*

Correctness: *We require $\mathsf{Dec}(\mathsf{KG}(\mathsf{MSK}, f), \mathsf{Enc}(\mathsf{MSK}, m)) = f(m)$ for every $m \in \mathcal{M}$, $f \in \mathcal{F}$, and $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.*

Next, we introduce selective-message message privacy for SKFE schemes.

Definition 7 (Selective-message message privacy). *Let SKFE be an SKFE scheme whose message space and function space are $\mathcal{M}$ and $\mathcal{F}$, respectively. Let q be a polynomial of λ. We define the selective-message message privacy game between a challenger and an adversary $\mathcal{A}$ as follows.*

1. *The challenger generates a master secret key $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ and chooses the challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. Then, the challenger sends security parameter 1^λ to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [p]}$ to the challenger, where p is an a-priori unbounded polynomial of λ.*

3. *The challenger generates ciphertexts* $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b^{\ell})(\ell \in [p])$ *and sends them to* $\mathcal{A}$.
4. $\mathcal{A}$ *may adaptively make key queries* q *times at most. For a key query* $f \in \mathcal{F}$ *from* $\mathcal{A}$, *the challenger generates* $sk_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f)$, *and returns* sk_f *to* $\mathcal{A}$. *Here,* f *needs to satisfy* $f(m_0^{\ell}) = f(m_1^{\ell})$ *for all* $\ell \in [p]$.
5. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$.

In this game, we define the advantage of $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{sm\text{-}mp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) = 2|\Pr[b = b'] - \frac{1}{2}| = |\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \ .$$

$\mathcal{A}$ *is said to be valid if each function query* f *made by* $\mathcal{A}$ *satisfies that* $f(m_0^{\ell}) = f(m_1^{\ell})$ *for all* $\ell \in [p]$ *in the above game. For a negligible function* $\epsilon(\cdot)$, *We say that* SKFE *is* (q, ϵ)*-selective-message message private if for any valid PPT* $\mathcal{A}$, *we have* $\mathsf{Adv}^{\mathsf{sm\text{-}mp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

We further say that an SKFE scheme is ϵ-secure collusion-resistant SKFE if it is (q, ϵ)-selective-message message private for any polynomial q.

Next, we define the succinctness for SKFE.

Definition 8 (Succinctness). *Let* $\mathcal{F}$ *be a function family. Let* s *and* n *be the maximum size and input length of functions contained in* $\mathcal{F}$, *respectively. We say that SKFE for* $\mathcal{F}$ *is weakly succinct if the size of the encryption circuit is bounded by* $s^{\gamma} \cdot \mathrm{poly}(\lambda, n)$, *where* $\gamma < 1$ *is a fixed constant.*

4.4 Indistinguishability Obfuscation

We review the definition of indistinguishability obfuscation (IO).

Definition 9 (Indistinguishability obfuscation). *A PPT algorithm* $i\mathcal{O}$ *is an indistinguishability obfuscator (IO) for a circuit class* $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ *if it satisfies the following two conditions.*

Functionality: *for all security parameters* $\lambda \in \mathbb{N}$, *for all* $C \in \mathcal{C}_\lambda$, *for all inputs* x, *we have that* $\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(1^\lambda, C)] = 1$.

Indistinguishability: *for any PPT distinguisher* $\mathcal{D}$, *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds: for all security parameters* $\lambda \in \mathbb{N}$, *for all pairs of circuits* $C_0, C_1 \in \mathcal{C}_\lambda$ *of the same size and such that* $C_0(x) = C_1(x)$ *for all inputs* x, *then*

$$|\Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_0)) = 1\right] - \Pr\left[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1\right]| = \mathsf{negl}(\lambda) \ .$$

We further say that $i\mathcal{O}$ *is* ϵ*-secure, for some concrete negligible function* $\epsilon(\cdot)$, *if for any PPT distinguisher the above advantage is smaller than* $\epsilon(\lambda)^{\Omega(1)}$.

4.5 Strong Exponentially-Efficient Indistinguishability Obfuscation

We next define strong exponentially-efficient IO (SXIO).

Definition 10 (Strong exponentially-efficient indistinguishability obfuscation). *Let $\gamma < 1$ be a constant. A PPT algorithm $\mathsf{sxi}\mathcal{O}$ is a γ-compressing strong exponentially-efficient indistinguishability obfuscator (SXIO) for a circuit class $\{\mathcal{C}\}_{\lambda \in \mathbb{N}}$ if it satisfies the functionality and indistinguishability in Definition 9 and the following efficiency requirement:*

Non-trivial time efficiency *We require that the running time of $\mathsf{sxi}\mathcal{O}$ on input $(1^\lambda, C)$ is at most $2^{n\gamma} \cdot \mathrm{poly}(\lambda, |C|)$ for every $\lambda \in \mathbb{N}$ and circuit $C \in \{C_\lambda\}_{\lambda \in \mathbb{N}}$ with input length n.*

We have the following theorem.

Theorem 9 ([15]). *Assuming there exists ϵ-secure collusion-resistant SKFE for all circuits, where $\epsilon(\cdot)$ is a negligible function. Then, for any constant $\gamma < 1$, there exists ϵ-secure γ-compressing SXIO for polynomial-size circuits with logarithmic size input.*

5 Puncturable Secret-Key Functional Encryption

We introduce puncturable secret-key functional encryption (puncturable SKFE).

The notion of puncturable SKFE was introduced by Bitansky and Vaikuntanathan [17]. They showed that in their construction of IO, the building block PKFE can be replaced with puncturable SKFE. However, it has been open whether we can achieve puncturable SKFE without assuming PKFE.

In this work, we answer the question affirmatively. We show how to construct a relaxed variant of puncturable SKFE scheme that is single-key weakly-succinct. Our relaxed variant is sufficient for constructing IO. Our construction consists of two steps.

1. We prove that a single-key non-succinct puncturable SKFE scheme is constructed only from one-way functions.
2. We prove that we can transform the non-succinct scheme into an weakly-succinct one by using SXIO.

We can construct SXIO based on standard (i.e., not puncturable) SKFE by Theorem 9. Thus, we can construct our puncturable SKFE from standard SKFE.

5.1 Syntax

Our definition of puncturable SKFE introduced below is slightly different from that proposed by Bitansky and Vaikuntanathan [17]. However, we show that puncturable SKFE defined in this paper is also a sufficient building block of IO. We state differences between our definition and theirs after describing the syntax and security of our puncturable SKFE.

Definition 11 (Puncturable secret-key functional encryption). *A puncturable SKFE scheme* pSKFE *is a tuple* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *of six PPT algorithms. Below, let* $\mathcal{M}$*,* $\mathcal{F}$*, and* $\mathcal{T}$ *be the message space, function space, and tag space of* pSKFE*, respectively. In addition, let* q *be a polynomial denoting the upper bound of the number of issuable functional keys.*

- *The setup algorithm* Setup*, given a security parameter* 1^λ*, outputs a master secret key* MSK*.*
- *The key generation algorithm* KG*, given a master secret key* MSK*, function* $f \in \mathcal{F}$*, and an index* $i \in [q]$*, outputs a functional key* sk_f*.*
- *The encryption algorithm* Enc*, given a master secret key* MSK*, a tag* tag*, and a message* $m \in \mathcal{M}$*, outputs a ciphertext* CT*.*
- *The decryption algorithm* Dec*, given a functional key* sk_f*, a tag* tag*, and a ciphertext* CT*, outputs a message* $\tilde{m} \in \{\bot\} \cup \mathcal{M}$*.*
- *The puncturing algorithm* Punc*, given a master secret key* MSK *and a tag* tag*, outputs a punctured master secret key* $\mathsf{MSK}^*\{\mathsf{tag}\}$
- *The punctured encryption algorithm* PEnc*, given a punctured master secret key* MSK^**, a tag* tag'*, and a message* m*, outputs a ciphertext* CT*.*

Correctness: *For every* $m \in \mathcal{M}$*,* $f \in \mathcal{F}$*,* $i \in [q]$*,* $\mathsf{tag} \in \mathcal{T}$*, and* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$*, we require that* $\mathsf{Dec}\left(\mathsf{KG}\left(\mathsf{MSK}, f, i\right), \mathsf{tag}, \mathsf{Enc}\left(\mathsf{MSK}, \mathsf{tag}, m\right)\right) = f(m)$*.*

5.2 Security

In this section, we introduce two variants of security. Their difference is the functionality of punctured encryption algorithms.

Definition 12 (Secure puncturable SKFE). *Let* $\mathsf{pSKFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *be puncturable SKFE. Below, let* $\mathcal{M}$*,* $\mathcal{F}$*, and* $\mathcal{T}$ *be the message space, function space, and tag space of* pSKFE*, respectively. In addition, let* q *be a polynomial denoting the upper bound of the number of issuable functional keys. We say that* pSKFE *is secure puncturable SKFE if it satisfies the following properties.*

Functionality preserving under puncturing: *For every* $m \in \mathcal{M}$*,* $(\mathsf{tag}, \mathsf{tag}') \in \mathcal{T} \times \mathcal{T}$ *such that* $\mathsf{tag} \neq \mathsf{tag}'$*, randomness* r*,* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$*, and* $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{Punc}(\mathsf{MSK}, \mathsf{tag})$*, it holds that*

$$\mathsf{PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{tag}', m; r) = \mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}', m; r) \ .$$

Semantic security at punctured tag: *We define punctured semantic security game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates a master secret key* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. The challenger sends security parameter* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$*,* $\mathsf{tag} \in \mathcal{T}$*, and* $\{f_i\}_{i \in [q]} \in \mathcal{F}^q$ *to the challenger. We require that for every* $i \in [q]$ *it holds that* $f_i(m_0) = f_i(m_1)$*.*

3. *The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}, m_b)$*,* $\mathsf{sk}_{f_i} \leftarrow \mathsf{KG}(\mathsf{MSK}, f_i, i)$ *for every* $i \in [q]$*, and* $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{Punc}(\mathsf{MSK}, \mathsf{tag})$*.*
   Then, the challenger returns $(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT}, \{\mathsf{sk}_{f_i}\}_{i\in[q]})$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

In this game, we define the advantage of the adversary $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = 2|\Pr[b = b'] - \frac{1}{2}| = |\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \ .$$

$\mathcal{A}$ *is said to be valid if* $f_i(m_0) = f_i(m_1)$ *holds for every* $i \in [q]$ *in the above game. We say that* pSKFE *satisfies semantic security at punctured tag if for any valid PPT* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*.*
We further say that pSKFE *satisfies* ϵ*-semantic security at punctured tag, for some concrete negligible function* $\epsilon(\cdot)$*, if for any valid PPT* $\mathcal{A}$ *the above advantage* $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda)$ *is smaller than* $\epsilon(\lambda)^{\Omega(1)}$*.*

In addition, we say that pSKFE *is* ϵ*-secure puncturable SKFE if it satisfies functionality preserving under puncturing and* ϵ*-semantic security at punctured tag.*

Instead of functionality preserving under puncturing, we can consider a relaxed variant which we call *indistinguishability of functionality under puncturing.* This property requires that any PPT distinguisher cannot distinguish ciphertexts output by Enc and PEnc even given both master secret key and punctured master secret key. The formal definition is as follows.

Definition 13 (Indistinguishability of functionality under puncturing). *Let* $\mathsf{pSKFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *be puncturable SKFE whose message space and tag space are* $\mathcal{M}$ *and* $\mathcal{T}$*, respectively. We define indistinguishability of functionality game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates a master secret key* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. The challenger sends security parameter* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *sends* $m \in \mathcal{M}$ *and* $(\mathsf{tag}, \mathsf{tag}') \in \mathcal{T} \times \mathcal{T}$ *such that* $\mathsf{tag} \neq \mathsf{tag}'$ *to the challenger.*
3. *The challenger first computes* $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{Punc}(\mathsf{MSK}, \mathsf{tag})$*. Then, the challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \mathsf{tag}', m)$ *if* $b = 0$*, and otherwise* $\mathsf{CT} \leftarrow \mathsf{PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{tag}', m)$*.*
   Then, the challenger returns $(\mathsf{MSK}, \mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT})$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*.*

In this game, we define the advantage of the adversary $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathsf{if}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = 2|\Pr[b = b'] - \frac{1}{2}| = |\Pr[b' = 1|b = 0] - \Pr[b' = 1|b = 1]| \ .$$

We say that pSKFE *satisfies indistinguishability of functionality under puncturing if for any PPT* $\mathcal{A}$*, we have* $\mathsf{Adv}^{\mathsf{if}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$*.*

We further say that pSKFE *satisfies* ϵ*-indistinguishability of functionality under puncturing, for some concrete negligible function* $\epsilon(\cdot)$*, if for any PPT* $\mathcal{A}$ *the above advantage* $\mathsf{Adv}^{\mathsf{if}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda)$ *is smaller than* $\epsilon(\lambda)^{\Omega(1)}$*.*

Definition 14 (Secure puncturable SKFE with indistinguishability of functionality). *Let* pSKFE *be puncturable SKFE. Let* $\epsilon_1(\cdot)$ *and* $\epsilon_2(\cdot)$ *be some negligible functions. If* pSKFE *satisfies* ϵ_1*-semantic security at punctured tag and* ϵ_2*-indistinguishability of functionality under puncturing, then we say that* pSKFE *is* (ϵ_1, ϵ_2)*-secure puncturable SKFE with indistinguishability of functionality.*

5.3 Efficiency

We introduce the notion of succinctness for puncturable SKFE.

Definition 15 (Succinctness). *Let* $\mathcal{F}$ *be a function family. Let* s *and* n *be the maximum size and input length of functions contained in* $\mathcal{F}$*, respectively.*

Weakly-succinct: *Puncturable SKFE for* $\mathcal{F}$ *is said to be weakly-succinct if the size of both the encryption circuit and punctured encryption circuit are bounded by* $s^{\gamma} \cdot \mathrm{poly}(\lambda, n)$*, where* $\gamma < 1$ *is a fixed constant. We call* γ *the compression factor.*

Collusion-succinct: *Puncturable SKFE for* $\mathcal{F}$ *is said to be collusion-succinct if the size of both the encryption circuit and punctured encryption circuit are bounded by* $q^{\gamma} \cdot \mathrm{poly}(n, \lambda, s)$*, where* q *is the upper bound of issuable functional decryption keys and* $\gamma < 1$ *is a fixed constant. We call* γ *the compression factor.*

5.4 Difference from The Definition of Bitansky and Vaikuntanathan

There are three main differences between our definition of puncturable SKFE and that of Bitansky and Vaikuntanathan [17]. Two are about syntax. The other is about security.

Syntactical differences are as follows.

Tag-based encryption and decryption: In the definition of Bitansky and Vaikuntanathan, a master secret key is punctured at *two messages*. Their semantic security requires that no PPT adversary can distinguish ciphertexts of these two messages given the punctured master secret key.

We adopt the tag based syntax for the encryption and decryption algorithms while Bitansky and Vaikuntanathan do not. A tag-based definition is well-suited for our non-succinct puncturable SKFE scheme. When our non-succinct scheme encrypts a message, it generates a garbled circuit of an universal circuit into which the message is hardwired, and then masks labels of the garbled circuit by a string generated by puncturable PRF. A tag fed to the encryption algorithm is used as an input to puncturable PRF. See Section 6 for details.

In our construction of IO in Section 8, we use an input to an obfuscated circuit as a tag for ciphertexts of puncturable SKFE. Therefore, our IO construction is not significantly different from the IO construction based on puncturable SKFE by Bitansky and Vaikuntanathan from the syntactical point of view though ours is based on tag-based puncturable SKFE.

Index based key generation: We define the key generation algorithm as a stateful algorithm. In other words, for the i-th invocation, we need to feed an index i to the key generation algorithm in addition to a master secret key and a function. This is because we transform a non-succinct scheme into an weakly-succinct one *via a collusion-succinct scheme whose key generation algorithm is stateful* in Section 7.

We note that our stateful collusion-succinct scheme is just an intermediate scheme to achieve IO. We also emphasize the fact that the index-based key generation is not an issue to construct IO because our main building block is a *single-key* weakly-succinct puncturable SKFE scheme. For a single-key scheme, we do not need any state for key generation because it can issue only a single functional key.

Below, we omit the index of single-key schemes in the syntax for simplicity.

Functionality under puncturing. In addition to the syntactic differences above, there is a difference about security. We defined indistinguishability of functionality under puncturing in Definition 13. The reason why we introduce the relaxed notion of functionality preserving property is that our weakly-succinct scheme does not satisfy functionality preserving under puncturing in Definition 12 but the relaxed one. Our *non-succinct* scheme satisfies functionality preserving under puncturing.

One might think that puncturable SKFE satisfying indistinguishability of functionality under puncturing is not sufficient to construct IO. This is not the case. We show that indistinguishability of functionality under puncturing suffices for constructing IO and our weakly-succinct scheme satisfies the property.

6 Single-Key Non-Succinct Puncturable SKFE

We show we can construct a single-key (non-succinct) puncturable SKFE scheme assuming only one-way functions. This construction is similar to that of single-key non-succinct PKFE proposed by Sahai and Seyalioglu [61]. Their construction is based on garbling scheme and public-key encryption. In our construction, we use puncturable PRF instead of public-key encryption, and, as a result, achieve the puncturable property. We recall that we can realize both garbling scheme and puncturable PRF assuming only one-way functions. We give the construction below.

Let $\mathsf{GC} = (\mathsf{Grbl}, \mathsf{Eval})$ be garbling scheme, and $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc}_{\mathsf{F}})$ be puncturable PRF. Using GC and PPRF, we construct puncturable SKFE $\mathsf{OneKey} = (\mathsf{1Key.Setup}, \mathsf{1Key.KG}, \mathsf{1Key.Enc}, \mathsf{1Key.Dec}, \mathsf{1Key.Punc}, \mathsf{1Key.PEnc})$ supporting only one functional key as follows. Note that the tag space of OneKey is the same as the domain of PPRF. In addition, the index space of OneKey is $[1]$, and thus we omit the index from the description by assuming the index is always fixed to 1. Below, we assume that we can represent every function f by an s-bit string $(f[1], \cdots, f[s])$.

Construction. The scheme consists of the following algorithms.

```
$\mathsf{1Key.Setup}(1^\lambda)$ :
    – Generate $S_{j,\alpha} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for every $j \in [s]$ and $\alpha \in \{0,1\}$.
    – Return $\mathsf{MSK} \leftarrow \{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$.
$\mathsf{1Key.KG}(\mathsf{MSK}, f)$ :
    – Parse $\{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}$ and $(f[1], \cdots, f[s]) \leftarrow f$.
    – Return $\mathsf{sk}_f \leftarrow (f, \{S_{j,f[j]}\}_{j\in[s]})$.
$\mathsf{1Key.Enc}(\mathsf{MSK}, \mathsf{tag}, m)$ :
    – Parse $\{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}$.
    – Compute $(\widetilde{U}, \{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, U(\cdot, m))$.
    – For every $j \in [s]$ and $\alpha \in \{0,1\}$, compute $R_{j,\alpha} \leftarrow \mathsf{F}(S_{j,\alpha}, \mathsf{tag})$ and
      $c_{j,\alpha} \leftarrow L_{j,\alpha} \oplus R_{j,\alpha}$.
    – Return $\mathsf{CT} \leftarrow (\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.
$\mathsf{1Key.Dec}(\mathsf{sk}_f, \mathsf{tag}, \mathsf{CT})$ :
    – Parse $(f, \{S_j\}_{j\in[s]}) \leftarrow \mathsf{sk}_f$ and $(\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{CT}$.
    – For every $j \in [s]$, compute $R_j \leftarrow \mathsf{F}(S_j, \mathsf{tag})$ and $L_j \leftarrow c_{j,f[j]} \oplus R_j$.
    – Return $y \leftarrow \mathsf{Eval}(\widetilde{U}, \{L_j\}_{j\in[s]})$.
$\mathsf{1Key.Punc}(\mathsf{MSK}, \mathsf{tag})$ :
    – Parse $\{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}$.
    – For every $j \in [s]$ and $\alpha \in \{0,1\}$, compute $S^*_{j,\alpha}\{\mathsf{tag}\} \leftarrow \mathsf{Punc}_\mathsf{F}(S_{j,\alpha}, \mathsf{tag})$.
    – Return $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \{S^*_{j,\alpha}\{\mathsf{tag}\}\}_{j\in[s],\alpha\in\{0,1\}}$.
$\mathsf{1Key.PEnc}(\mathsf{MSK}^*, \mathsf{tag}', m)$
    – Parse $\{S^*_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}^*$.
    – Compute $(\widetilde{U}, \{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, U(\cdot, m))$.
    – For every $j \in [s]$ and $\alpha \in \{0,1\}$, compute $R_{j,\alpha} \leftarrow \mathsf{F}_{S^*_{j,\alpha}}(\mathsf{tag}')$ and $c_{j,\alpha} \leftarrow$
      $L_{j,\alpha} \oplus R_{j,\alpha}$.
    – Return $\mathsf{CT} \leftarrow (\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.
```

Then, we have the following theorem.

Theorem 10. *Let* GC *be* δ*-secure garbling scheme, and* PPRF δ*-secure puncturable PRF, where* $\delta(\cdot)$ *is some negligible function. Then,* OneKey *is* δ*-secure single-key puncturable SKFE.*

See [45] for the formal proof of this theorem.

7 From Non-Succinct Puncturable SKFE to Weakly-Succinct One

In this section, we show how to transform single-key non-succinct puncturable SKFE into single-key weakly-succinct one using SXIO. Note that the resulting scheme satisfies only indistinguishability of functionality under puncturing property even if we start the transformation with a non-succinct scheme satisfying functionality preserving under puncturing property.

The transformation consists of 2 steps. First, we show how to construct collusion-succinct puncturable SKFE from single-key non-succinct puncturable SKFE and SXIO. Then, we give the transformation from collusion-succinct puncturable SKFE to weakly-succinct one.

In fact, the intermediate collusion-succinct scheme satisfies only indistinguishability of functionality under puncturing property. This is because we adopt a construction technique similar to that proposed by Lin et al. [53] (and extended by Bitansky et al. [15] and Kitagawa et al. [46]), and thus we use an obfuscated encryption circuit of the building block scheme by SXIO as a ciphertext of the resulting scheme. This fact is the reason the resulting weakly-succinct scheme satisfies only indistinguishability of functionality under puncturing property.

7.1 From Non-Succinct to Collusion-Succinct by Using SXIO

For any q which is a fixed polynomial of λ, we show how to construct a puncturable SKFE scheme whose index space is $[q]$ based on a single-key puncturable SKFE scheme. The resulting scheme is collusion-succinct, that is, the running time of both the encryption algorithm and the punctured encryption algorithm are sub-linear in q. We show the construction below.

Let $\mathsf{OneKey} = (\mathsf{1Key.Setup}, \mathsf{1Key.KG}, \mathsf{1Key.Enc}, \mathsf{1Key.Dec}, \mathsf{1Key.Punc}, \mathsf{1Key.PEnc})$ be puncturable SKFE that we constructed in Section 6. Let $\mathsf{sxi}\mathcal{O}$ be SXIO and $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc_F})$ puncturable PRF. Using OneKey, $\mathsf{sxi}\mathcal{O}$, and PPRF, we construct puncturable SKFE $\mathsf{CollSuc} = (\mathsf{CS.Setup}, \mathsf{CS.KG}, \mathsf{CS.Enc}, \mathsf{CS.Dec}, \mathsf{CS.Punc}, \mathsf{CS.PEnc})$ as follows. We again note that q is a fixed polynomial of λ. Let the tag space of $\mathsf{CollSuc}$ be $\mathcal{T}$. Then, the tag space of OneKey is also $\mathcal{T}$.

Construction. The scheme consists of the following algorithms.

$\mathsf{CS.Setup}(1^\lambda)$:
- Generate $S \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and return $\mathsf{MSK} \leftarrow S$.

$\mathsf{CS.KG}(\mathsf{MSK}, f, i)$:
- Parse $S \leftarrow \mathsf{MSK}$.
- Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$ and $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
- Compute $\mathsf{1Key.sk}_f \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f)$ and return $\mathsf{sk}_f \leftarrow (i, \mathsf{1Key.sk}_f)$.

$\mathsf{CS.Enc}(\mathsf{MSK}, \mathsf{tag}, m)$:
- Parse $S \leftarrow \mathsf{MSK}$.
- Generate $S_{\mathsf{Enc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and return $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E_{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m])$. The circuit $\mathsf{E_{1Key}}$ is defined in Figure 4.

$\mathsf{CS.Dec}(\mathsf{sk}_f, \mathsf{tag}, \mathsf{CT})$:
- Parse $(i, \mathsf{1Key.sk}_f) \leftarrow \mathsf{sk}_f$.
- Compute $\mathsf{CT}_i \leftarrow \mathsf{CT}(i)$ and return $y \leftarrow \mathsf{1Key.Dec}(\mathsf{1Key.sk}_f, \mathsf{tag}, \mathsf{CT}_i)$.

$\mathsf{CS.Punc}(\mathsf{MSK}, \mathsf{tag})$:
- Parse $S \leftarrow \mathsf{MSK}$.
- Generate $S_{\mathsf{Punc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and compute $\widetilde{\mathsf{P}} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P_{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}])$. The circuit $\mathsf{P_{1Key}}$ is defined in Figure 5.
- Return $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \widetilde{\mathsf{P}}$.

$\mathsf{CS.PEnc}(\mathsf{MSK}^*, \mathsf{tag}', m)$:

- Parse $\widetilde{\mathsf{P}} \leftarrow \mathsf{MSK}^*$.
- Generate $S_{\mathsf{Enc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and return $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{PE}_{\mathsf{1Key}}[\widetilde{\mathsf{P}}, S_{\mathsf{Enc}}, \mathsf{tag}', m])$. The circuit $\mathsf{PE}_{\mathsf{1Key}}$ is defined in Figure 6.

Encryption circuit $\mathsf{E}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m](i)$:

Hardwired: Two PRF keys S and S_{Enc}, a tag tag, and a message m.
Input: An index $i \in [q]$.
Padding: This circuit is padded to size $\mathsf{pad}_{\mathsf{E}} \coloneqq \mathsf{pad}_{\mathsf{E}}(\lambda, n, s)$, which is determined in analysis.

1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$ and $r_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_{\mathsf{Enc}}}(i)$.
2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
3. Return $\mathsf{CT}_i \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_i, \mathsf{tag}, m; r_{\mathsf{Enc}})$.

Fig. 4. The description of $\mathsf{E}_{\mathsf{1Key}}$.

Punctured key generation circuit $\mathsf{P}_{\mathsf{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}](i)$:

Hardwired: Two PRF keys S and S_{Punc}, and a tag tag.
Input: An index $i \in [q]$.
Padding: This circuit is padded to size $\mathsf{pad}_{\mathsf{P}} \coloneqq \mathsf{pad}_{\mathsf{P}}(\lambda, n, s)$, which is determined in analysis.

1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$ and $r_{\mathsf{Punc}} \leftarrow \mathsf{F}_{S_{\mathsf{Punc}}}(i)$.
2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
3. Return $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{1Key.Punc}(\mathsf{MSK}_i, \mathsf{tag}; r_{\mathsf{Punc}})$.

Fig. 5. The description of $\mathsf{P}_{\mathsf{1Key}}$.

Then, we have the following theorem.

Theorem 11. *Let $\delta(\cdot)$ be some negligible function. Let OneKey be δ-secure single-key puncturable SKFE constructed in Section 6. Let $\mathsf{sxi}\mathcal{O}$ be δ-secure γ-compressing SXIO, where γ is a sufficiently small constant such that $\gamma < 1$. Let PPRF be δ-secure puncturable PRF. Then, $\mathsf{CollSuc}$ is (δ, δ)-secure puncturable SKFE with indistinguishability of functionality that is collusion-succinct with compression factor $\hat{\gamma}$, which is a constant smaller than 1.*

Punctured encryption circuit $\mathsf{PE}_{\mathsf{1Key}}[\widetilde{\mathsf{P}}, S_{\mathsf{Enc}}, \mathsf{tag}, m](i)$:

Hardwired: A circuit $\widetilde{\mathsf{P}}$, a PRF key S_{Enc}, a tag tag, and a message m.

Input: An index $i \in [q]$.

Padding: This circuit is padded to size $\mathsf{pad}_{\mathsf{E}} \coloneqq \mathsf{pad}_{\mathsf{E}}(\lambda, n, s)$, which is determined in analysis.

1. Compute $\mathsf{MSK}_i^* \leftarrow \widetilde{\mathsf{P}}(i)$ and $r_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_{\mathsf{Enc}}}(i)$.
2. Return $\mathsf{CT}_i \leftarrow \mathsf{1Key.PEnc}(\mathsf{MSK}_i^*, \mathsf{tag}, m; r_{\mathsf{Enc}})$.

Fig. 6. The description of $\mathsf{PE}_{\mathsf{1Key}}$.

See [45] for the formal proof of this theorem.

The requirement for γ and the concrete value of $\hat{\gamma}$ is determined in the efficiency analysis in the proof of Theorem 11. We can make $\hat{\gamma}$ smaller than 1 by using SXIO with sufficiently small compression factor γ as the building block. Such SXIO is constructed from collusion-resistant SKFE [15].

7.2 From Collusion-Succinct to Weakly-Succinct

In this section, we show how to construct a single-key weakly-succinct puncturable SKFE scheme from a collusion-succinct one.

This transformation is based on those proposed by Bitansky and Vaikuntanathan [17] and Ananth et al. [3], and thus utilizes a decomposable randomized encoding. The difference is that we must consider puncturing and punctured encryption algorithms since we construct a puncturable SKFE scheme. In fact, we show their construction works for puncturable SKFE schemes. In addition, we consider semantic security defined in the weakly selective security manner while they considered selective security. Below, we give the construction.

We construct single-key puncturable SKFE $\mathsf{WeakSuc} = (\mathsf{WS.Setup}, \mathsf{WS.KG}, \mathsf{WS.Enc}, \mathsf{WS.Dec}, \mathsf{WS.Punc}, \mathsf{WS.PEnc})$. Let s and n be the maximum size and input length of functions supported by $\mathsf{WeakSuc}$. Let RE be c-local decomposable randomized encoding, where c is a constant. We suppose that the number of decomposed encodings of RE for a function of size s is μ. Then, μ is a polynomial bounded by $s \cdot \mathrm{poly}_{\mathsf{RE}}(\lambda, n)$, where $\mathrm{poly}_{\mathsf{RE}}(\lambda, n)$ is a fixed polynomial. We also suppose that the randomness space of RE is $\{0,1\}^{\rho}$, where ρ is a polynomial bounded by $s \cdot \mathrm{poly}_{\mathsf{RE}}(\lambda, n)$. Let $\mathsf{CollSuc} = (\mathsf{CS.Setup}, \mathsf{CS.KG}, \mathsf{CS.Enc}, \mathsf{CS.Dec}, \mathsf{CS.Punc}, \mathsf{CS.PEnc})$ be puncturable SKFE whose index space and tag space are $[\mu]$ and $\mathcal{T}$, respectively. Let $\mathsf{SKE} = (\mathsf{E}, \mathsf{D})$ be SKE and PRF PRF. In the scheme, we use $\mathsf{PRF} : \{0,1\}^{\lambda} \times (\{0,1\}^{\lambda} \times [\rho]) \rightarrow \{0,1\}$. Using $\mathsf{CollSuc}$, RE, SKE, and PRF, we construct $\mathsf{WeakSuc}$ as follows. The tag space of $\mathsf{WeakSuc}$ is $\mathcal{T}$.

$\mathsf{WS.Setup}(1^{\lambda})$:
- Return $\mathsf{MSK} \leftarrow \mathsf{CS.Setup}(1^{\lambda})$.

$\mathsf{WS.KG}(\mathsf{MSK}, f)$:
- Generate $K \xleftarrow{r} \{0,1\}^\lambda$ and $t \leftarrow \{0,1\}^\lambda$.
- Compute $\widehat{f} \leftarrow \mathsf{RE}(1^\lambda, f)$ and decomposed encodings $\widehat{f}_1, \cdots \widehat{f}_\mu$ together with sets of integers $(R_1, \cdots, R_\mu)$. R_i indicates which bit of a randomness $\widehat{f}_i$ depends on for every $i \in [\mu]$. Note that $R_i \subseteq [\rho]$ and $|R_i| = c$ for every $i \in [\mu]$.
- Generate $\mathsf{CT}_i^{\mathsf{ske}} \leftarrow \mathsf{E}(K, 0^{|\widehat{f}_i(\cdot,\cdot)|})$, and compute $\mathsf{sk}_{\mathsf{En}_i} \leftarrow \mathsf{CS.KG}(\mathsf{MSK}, \mathsf{En}_{\mathsf{dre}}[\widehat{f}_i, R_i, t, \mathsf{CT}_i^{\mathsf{ske}}], i)$ for every $i \in [\mu]$. $\mathsf{En}_{\mathsf{dre}}$ defined in Figure 7.
- Return $\mathsf{sk}_f \leftarrow (\mathsf{sk}_{\mathsf{En}_1}, \cdots, \mathsf{sk}_{\mathsf{En}_\mu})$.

$\mathsf{WS.Enc}(\mathsf{MSK}, \mathsf{tag}, m)$:
- Generate $S_{\mathsf{encd}} \leftarrow \{0,1\}^\lambda$.
- Return $\mathsf{CT} \leftarrow \mathsf{CS.Enc}(\mathsf{MSK}, \mathsf{tag}, (m, S_{\mathsf{encd}}, \bot))$.

$\mathsf{WS.Dec}(\mathsf{sk}_f, \mathsf{tag}, \mathsf{CT})$:
- Parse $(\mathsf{sk}_{\mathsf{En}_1}, \cdots, \mathsf{sk}_{\mathsf{En}_\mu}) \leftarrow \mathsf{sk}_f$.
- For every $i \in [\mu]$, compute $e_i \leftarrow \mathsf{CS.Dec}(\mathsf{sk}_{\mathsf{En}_i}, \mathsf{tag}, \mathsf{CT})$.
- Decode y from $(e_1, \cdots, e_\mu)$.
- Return y.

$\mathsf{WS.Punc}(\mathsf{MSK}, \mathsf{tag})$:
- Return $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{CS.Punc}(\mathsf{MSK}, \mathsf{tag})$.

$\mathsf{WS.PEnc}(\mathsf{MSK}^*, \mathsf{tag}', m)$:
- Generate $S_{\mathsf{encd}} \leftarrow \{0,1\}^\lambda$.
- Return $\mathsf{CT} \leftarrow \mathsf{CS.PEnc}(\mathsf{MSK}^*, \mathsf{tag}', (m, S_{\mathsf{encd}}, \bot))$.

Decomposable Randomized Encoding Circuit

$\mathsf{En}_{\mathsf{dre}}[\widehat{f}_i, R_i, t, \mathsf{CT}_i^{\mathsf{ske}}](m, S_{\mathsf{encd}}, K)$

Hardwired: A randomized encoding $\widehat{f}_i$, a set R_i, a string t, and a ciphertext $\mathsf{CT}_i^{\mathsf{ske}}$.

Input: A message m, a PRF key S_{encd}, and an SKE secret key K.

1. If $m = \bot$, return $e_i \leftarrow \mathsf{D}(K, \mathsf{CT}_i^{\mathsf{ske}})$.
2. Else, compute as follows:
   - For $j \in R_i$, compute $r_j \leftarrow \mathsf{PRF}(S_{\mathsf{encd}}, t \| j)$, set $r_{R_i} \leftarrow \{r_j\}_{j \in R_i}$.
   - Return $e_i \leftarrow \widehat{f}_i(m; r_{R_i})$.

Fig. 7. The description of $\mathsf{En}_{\mathsf{dre}}$.

Then, we have the following theorem.

Theorem 12. *Let $\delta(\cdot)$ be negligible function. Let* CollSuc *be (δ,δ)-secure punc-turable SKFE with indistinguishability of functionality that can issue μ functional keys and is collusion-succinct with compression factor γ, where $\gamma < 1$ is a con-stant. Let* RE*,* SKE*, and* PRF *be δ-secure decomposable randomized encoding,* *SKE, and PRF, respectively. Then,* WeakSuc *be (δ,δ)-secure single-key puncturable SKFE with indistinguishability of functionality that is weakly-succinct with compression factor γ', where γ' is a constant such that $\gamma < \gamma' < 1$.*

See [45] for the formal proof of this theorem.

8 Indistinguishability Obfuscation from SKFE

We show how to obtain IO based on SKFE via puncturable SKFE.

8.1 IO from Collusion-Resistant SKFE

We construct IO from puncturable SKFE satisfying only indistinguishability of functionality under puncturing. Formally, we have the following theorem.

Theorem 13. *Let $\delta(\lambda) = 2^{-\lambda^{\epsilon}}$, where $\epsilon < 1$ is a constant. Assuming there exists (δ,δ)-secure single-key weakly-succinct puncturable SKFE with indistinguishability of functionality for all circuits. Then, there exists secure IO for all circuits.*

We omit the formal proof of it. See Section 2.4 for the overview of it. In [45], we formally prove it by first providing the construction of IO based on puncturable SKFE, and then analyzing its security and efficiency.

In addition, by combining Theorems 9, 10, 11, and 12, we also obtain the following theorem.

Theorem 14. *Assuming there exists δ-secure collusion-resistant SKFE for all circuits, where $\delta(\cdot)$ is a negligible function. Then, there exists (δ,δ)-secure single-key weakly-succinct puncturable SKFE with indistinguishability of functionality for all circuits.*

In order to obtain Theorem 14, we also use δ-secure PRF, puncturable PRF, plain SKE, garbling scheme, and decomposable randomized encoding as building blocks. From Theorems 4, 5, 6, 7, and 8, all of these primitives are implied by δ-secure one-way functions thus implied by δ-secure collusion-resistant SKFE for all circuits.

From Theorems 13 and 14, we obtain the following main theorem.

Theorem 15. *Let $\delta(\lambda) = 2^{-\lambda^{\epsilon}}$, where $\epsilon < 1$ is a constant. Assuming there exists δ-secure collusion-resistant SKFE for all circuits. Then, there exists secure IO for all circuits.*

Remark 1 (IO for circuits with input of poly-logarithmic length). The security loss of our IO construction is exponential in the input length of circuits, but is independent of the size of circuits. Thus, if the input length of circuits is poly-logarithmic in the security parameter, our IO construction incurs only quasi-polynomial security loss regardless of the size of circuits. Therefore, we can obtain IO for circuits of *polynomial size* with input of poly-logarithmic length from *quasi-polynomially secure* collusion-resistant SKFE for all circuits. This is an improvement over the IO construction by Komargodski and Segev [48]. They showed that IO for circuits of *sub-polynomial size* with input of poly-logarithmic length is constructed from quasi-polynomially secure collusion-resistant SKFE for all circuits.

Komargodski and Segev also showed that the combination of their IO and sub-exponentially secure one-way functions yields succinct and collusion-resistant PKFE for circuits of *sub-polynomial size* with input of poly-logarithmic length. We observe that our IO for circuits of polynomial size with input of poly-logarithmic length leads to succinct and collusion-resistant PKFE for circuits of *polynomial size* with input of poly-logarithmic length by combining sub-exponentially secure one-way functions from the result of Komargodski and Segev.

8.2 Collusion-Resistant SKFE from Weakly-Succinct One

We also show that collusion-resistant SKFE is constructed from single-key weakly-succinct SKFE. Formally, we have the following theorem.

Theorem 16. *Let $\delta(\lambda) = \lambda^{-\zeta}$, where $\zeta = \omega(1)$. Assuming there exists $(1, \delta)$-selective-message message private SKFE for all circuits that is weakly succinct. Then, there exists δ'-secure collusion-resistant SKFE for all circuits, where $\delta'(\lambda) = \lambda^{-\zeta^{1/2}}$.*[11]

In [44], we formally show Theorem 16. See Section 3 for the overview for this result.

Theorem 16 states that if the underlying single-key scheme is sub-exponentially secure, then so is the resulting scheme. Therefore, from Theorems 15 and 16, we have the following corollary.

Corollary 2. *Assuming there exists sub-exponentially secure single-key weakly-succinct SKFE for all circuits. Then, there exists IO for all circuits.*

Acknowledgement. The first and third authors are supported by NTT Secure Platform Laboratories, JST CREST JPMJCR14D6, JST OPERA, JSPS KAKENHI JP16H01705, JP16J10322, JP17H01695.

References

1. P. Ananth, Z. Brakerski, G. Segev, and V. Vaikuntanathan. From selective to adaptive security in functional encryption. *CRYPTO 2015, Part II*, *LNCS* 9216, pp. 657–677. 2015.
2. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. *CRYPTO 2015, Part I*, *LNCS* 9215, pp. 308–326. 2015.

[11] We can slightly generalize the result. By setting $\eta = \zeta^{1/c}$ in the construction for any constant $c > 1$, we can achieve $\delta'(\lambda) = \lambda^{-\zeta^{1/c}}$.

3. P. Ananth, A. Jain, and A. Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730.
4. P. Ananth and A. Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. *EUROCRYPT 2017, Part I*, *LNCS* 10210, pp. 152–181. 2017.
5. P. V. Ananth, D. Gupta, Y. Ishai, and A. Sahai. Optimizing obfuscation: Avoiding Barrington's theorem. *ACM CCS 14*, pp. 646–658. 2014.
6. Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
7. D. Apon, N. Döttling, S. Garg, and P. Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In *44rd International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland (to appear)*, 2017.
8. B. Applebaum and Z. Brakerski. Obfuscating circuits via composite-order graded encoding. *TCC 2015, Part II*, *LNCS* 9015, pp. 528–556. 2015.
9. G. Asharov and G. Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *56th FOCS*, pp. 191–209. 2015.
10. S. Badrinarayanan, E. Miles, A. Sahai, and M. Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. *EUROCRYPT 2016, Part II*, *LNCS* 9666, pp. 764–791. 2016.
11. B. Barak, S. Garg, Y. T. Kalai, O. Paneth, and A. Sahai. Protecting obfuscation against algebraic attacks. *EUROCRYPT 2014*, *LNCS* 8441, pp. 221–238. 2014.
12. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. P. Vadhan, and K. Yang. On the (im)possibility of obfuscating programs. *CRYPTO 2001*, *LNCS* 2139, pp. 1–18. 2001.
13. M. Bellare, V. T. Hoang, and P. Rogaway. Foundations of garbled circuits. *ACM CCS 12*, pp. 784–796. 2012.
14. N. Bitansky, S. Garg, H. Lin, R. Pass, and S. Telang. Succinct randomized encodings and their applications. *47th ACM STOC*, pp. 439–448. 2015.
15. N. Bitansky, R. Nishimaki, A. Passelègue, and D. Wichs. From cryptomania to obfustopia through secret-key functional encryption. *TCC 2016-B, Part II*, *LNCS* 9986, pp. 391–418. 2016.
16. N. Bitansky, O. Paneth, and D. Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. *TCC 2016-A, Part I*, *LNCS* 9562, pp. 474–502. 2016.
17. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *56th FOCS*, pp. 171–190. 2015.
18. D. Boneh, D. Gupta, I. Mironov, and A. Sahai. Hosting services on an untrusted cloud. *EUROCRYPT 2015, Part II*, *LNCS* 9057, pp. 404–436. 2015.
19. D. Boneh, P. A. Papakonstantinou, C. Rackoff, Y. Vahlis, and B. Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *49th FOCS*, pp. 283–292. 2008.
20. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. *TCC 2011*, *LNCS* 6597, pp. 253–273. 2011.
21. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. *ASIACRYPT 2013, Part II*, *LNCS* 8270, pp. 280–300. 2013.
22. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. *PKC 2014*, *LNCS* 8383, pp. 501–519. 2014.

23. Z. Brakerski, I. Komargodski, and G. Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *EUROCRYPT 2016, Part II*, *LNCS* 9666, pp. 852–880. 2016.
24. Z. Brakerski and G. N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. *TCC 2014*, *LNCS* 8349, pp. 1–25. 2014.
25. Z. Brakerski and G. Segev. Function-private functional encryption in the private-key setting. *TCC 2015, Part II*, *LNCS* 9015, pp. 306–324. 2015.
26. R. Canetti, J. Holmgren, A. Jain, and V. Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. *47th ACM STOC*, pp. 429–437. 2015.
27. R. Canetti, H. Lin, S. Tessaro, and V. Vaikuntanathan. Obfuscation of probabilistic circuits and applications. *TCC 2015, Part II*, *LNCS* 9015, pp. 468–497. 2015.
28. Y. Chen, C. Gentry, and S. Halevi. Cryptanalyses of candidate branching program obfuscators. *EUROCRYPT 2017, Part II*, *LNCS* 10211, pp. 278–307. 2017.
29. A. Cohen, J. Holmgren, R. Nishimaki, V. Vaikuntanathan, and D. Wichs. Watermarking cryptographic capabilities. *48th ACM STOC*, pp. 1115–1127. 2016.
30. J.-S. Coron, C. Gentry, S. Halevi, T. Lepoint, H. K. Maji, E. Miles, M. Raykova, A. Sahai, and M. Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. *CRYPTO 2015, Part I*, *LNCS* 9215, pp. 247–266. 2015.
31. J.-S. Coron, M. S. Lee, T. Lepoint, and M. Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. *PKC 2017, Part I*, *LNCS* 10174, pp. 41–58. 2017.
32. R. Fernando, P. M. R. Rasmussen, and A. Sahai. Preventing CLT attacks on obfuscation with linear overhead. Cryptology ePrint Archive, Report 2016/1070.
33. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pp. 40–49. 2013.
34. S. Garg, E. Miles, P. Mukherjee, A. Sahai, A. Srinivasan, and M. Zhandry. Secure obfuscation in a weak multilinear map model. *TCC 2016-B, Part II*, *LNCS* 9986, pp. 241–268. 2016.
35. S. Garg, O. Pandey, A. Srinivasan, and M. Zhandry. Breaking the sub-exponential barrier in obfustopia. *EUROCRYPT 2017, Part II*, *LNCS* 10211, pp. 156–181. 2017.
36. S. Garg and A. Srinivasan. Single-key to multi-key functional encryption with polynomial loss. *TCC 2016-B, Part II*, *LNCS* 9986, pp. 419–442. 2016.
37. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807.
38. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. *EUROCRYPT 2014*, *LNCS* 8441, pp. 578–602. 2014.
39. D. Hofheinz, T. Jager, D. Khurana, A. Sahai, B. Waters, and M. Zhandry. How to generate and use universal samplers. *ASIACRYPT 2016, Part II*, *LNCS* 10032, pp. 715–744. 2016.
40. S. Hohenberger, A. Sahai, and B. Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. *EUROCRYPT 2014*, *LNCS* 8441, pp. 201–220. 2014.
41. R. Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pp. 134–147. 1995.
42. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pp. 44–61. 1989.

43. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. *ACM CCS 13*, pp. 669–684. 2013.
44. F. Kitagawa, R. Nishimaki, and K. Tanaka. From single-key to collusion-resistant secret-key functional encryption by leveraging succinctness. Cryptology ePrint Archive, Report 2017/638.
45. F. Kitagawa, R. Nishimaki, and K. Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. Cryptology ePrint Archive, Report 2017/361.
46. F. Kitagawa, R. Nishimaki, and K. Tanaka. Simple and generic constructions of succinct functional encryption. Cryptology ePrint Archive, Report 2017/275. To appear in PKC 2018
47. I. Komargodski, T. Moran, M. Naor, R. Pass, A. Rosen, and E. Yogev. One-way functions and (im)perfect obfuscation. In *55th FOCS*, pp. 374–383. 2014.
48. I. Komargodski and G. Segev. From minicrypt to obfustopia via private-key functional encryption. *EUROCRYPT 2017, Part I*, *LNCS* 10210, pp. 122–151. 2017.
49. V. Koppula, A. B. Lewko, and B. Waters. Indistinguishability obfuscation for turing machines with unbounded memory. *47th ACM STOC*, pp. 419–428. 2015.
50. B. Li and D. Micciancio. Compactness vs collusion resistance in functional encryption. *TCC 2016-B, Part II*, *LNCS* 9986, pp. 443–468. 2016.
51. H. Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. *EUROCRYPT 2016, Part I*, *LNCS* 9665, pp. 28–57. 2016.
52. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. *CRYPTO 2017, Part I*, *LNCS* 10401, pp. 599–629. 2017.
53. H. Lin, R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation with non-trivial efficiency. *PKC 2016, Part II*, *LNCS* 9615, pp. 447–462. 2016.
54. Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.
55. H. Lin and S. Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. *CRYPTO 2017, Part I*, *LNCS* 10401, pp. 630–660. 2017.
56. H. Lin and V. Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. *57th FOCS*, pp. 11–20. 2016.
57. Y. Lindell and B. Pinkas. A proof of security of yao's protocol for two-party computation. *J. Cryptology*, 22(2):161–188.
58. E. Miles, A. Sahai, and M. Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. *CRYPTO 2016, Part II*, *LNCS* 9815, pp. 629–658. 2016.
59. A. O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556.
60. R. Pass, K. Seth, and S. Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. *CRYPTO 2014, Part I*, *LNCS* 8616, pp. 500–517. 2014.
61. A. Sahai and H. Seyalioglu. Worry-free encryption: functional encryption with public keys. *ACM CCS 10*, pp. 463–472. 2010.
62. A. Sahai and B. Waters. How to use indistinguishability obfuscation: deniable encryption, and more. *46th ACM STOC*, pp. 475–484. 2014.
63. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. *EUROCRYPT 2005*, *LNCS* 3494, pp. 457–473. 2005.
64. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pp. 162–167. 1986.
65. J. Zimmerman. How to obfuscate programs directly. *EUROCRYPT 2015, Part II*, *LNCS* 9057, pp. 439–467. 2015.