

Obfustopia Built on Secret-Key Functional Encryption*

Fuyuki Kitagawa¹

Ryo Nishimaki¹

Keisuke Tanaka²

¹ NTT Corporation, Japan

{fuyuki.kitagawa.yh,ryo.nishimaki.zk}@hco.ntt.co.jp

² Tokyo Institute of Technology, Japan

keisuke@is.titech.ac.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 secure against an unbounded number of functional key queries, 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 standard 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.

Keywords: Indistinguishability obfuscation, secret-key functional encryption, puncturable secret-key functional encryption, succinctness, collusion-resistance

*An extended abstract of this paper appears in the proceedings of Eurocrypt 2018 as “Obfustopia Built on Secret-Key Functional Encryption [KNT18]”.

Contents

1	Introduction	2
1.1	Backgrounds	2
1.2	Our Results	3
1.3	More on Related Works	6
1.4	Organization	6
2	Overview of Our Technique	7
2.1	Construction of IO based on PKFE	7
2.2	Replacing PKFE with SKFE: Need of Puncturable SKFE	9
2.3	Puncturable SKFE from SKFE	9
2.4	IO from Puncturable SKFE	14
2.5	Technical Overview for Achieving Collusion-Resistance	17
3	Preliminaries	19
3.1	Standard Cryptographic Tools	20
3.2	Secret-Key Functional Encryption	23
3.3	Indistinguishability Obfuscation	27
4	Puncturable Secret-Key Functional Encryption	28
4.1	Syntax	28
4.2	Security	29
4.3	Difference from Definition of Bitansky and Vaikuntanathan	30
5	Single-Key Non-Succinct Puncturable SKFE	31
6	From Non-Succinct to Weakly Succinct Puncturable SKFE	34
6.1	From Non-Succinct to Collusion-Succinct by Using SXIO	34
6.2	From Collusion-Succinct to Weakly Succinct	40
7	Indistinguishability Obfuscation from Puncturable SKFE	43
7.1	Construction	44
7.2	Security Analysis	46
7.3	Efficiency Analysis	55
8	Basic Tools for Transformation from Succinctness to Collusion-Resistance	57
8.1	Parallel Construction	57
8.2	Single-Ciphertext Collusion-Resistant Fully Succinct SKFE	59
8.3	Hybrid Encryption Construction	62
9	New PRODUCT Construction for iSKFE	67
10	Collusion-Resistant SKFE via Size-Shifting	74
10.1	Intuition of Size-Shifting	75
10.2	Construction of Collusion-Resistant iSKFE	76
10.3	Analysis of Our Collusion-Resistant iSKFE	77
10.4	Converting iSKFE into SKFE	80
10.5	From Single-Key SKFE to Collusion-Resistant SKFE	81

11 Upgrading Succinctness and Security of SKFE	82
11.1 Transforming Weakly Succinct SKFE into Succinct One	82
11.2 Transforming Selective-message-function secure SKFE into Selective-secure One	83

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 their functionality. The theoretical study of program obfuscation was initiated by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI⁺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 a weaker variant of obfuscation called *indistinguishability obfuscation (IO)*. Impossibility of IO for all circuits is not known.

Garg, Gentry, Halevi, Raykova, Sahai, and Waters [GGH⁺16] 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 and one-way functions though it is weaker than virtual-black-box obfuscation [GGH⁺16, SW14, HSW14, BGMS15, K LW15, BCG⁺18, BPW16, CHN⁺18, HJK⁺16].

While we know the usefulness of IO well, we know very little about how to achieve IO. Many candidate constructions had been proposed. At the earlier stage, most IO constructions are based on little-studied cryptographic tools such as multi-linear maps [GGH⁺16, BGK⁺14, BR14, AGIS14, PST14, Zim15, GGH15, AB15, BMSZ16, GMM⁺16, Lin16, LV16, AS17, Lin17, FRS17, MZ18, CVW18, BGMZ18, CHVW19]. In most of them, security flaws were discovered [CGH⁺15, MSZ16, ADGM17, CLLT17, CGH17, CVW18] and none of them are proven to be secure from standard assumptions. Later, several candidates that do not rely on multilinear maps have been proposed. However, they still rely on some non-standard cryptographic tools such as perturbation-resilient generators [AJL⁺19, AJS18, JLMS19, JLS19], pseudo flawed-smudging generators [LM18, AJL⁺19], or correlated noise generators [Agr19]. Some candidates are based on non-standard lattice-based assumptions [BDGM20a, WW21, GP21, BDGM20b], but those assumptions were found to be false [HJL21]. In 2021, Jain, Lin, and Sahai finally propose an IO construction from well-founded assumptions [JLS21]. More specifically, they achieve perturbation-resilient generators from well-founded assumptions. In the IO constructions that do not rely multilinear maps, *functional encryption*, which is a seemingly weaker primitive than IO, plays a crucial role.

Functional encryption is one of the most advanced cryptographic primitives which enable a system having flexibility in controlling encrypted data [SW05, BSW11, O’N10]. 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 m 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 public-key functional encryption (PKFE) 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 [AJ15, BV18, AJS15, BNPW20].

Bitansky and Vaikuntanathan [BV18] and Ananth and Jain [AJ15] independently showed that we can construct IO based on 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 and Vaikuntanathan [BV18] and Ananth, Jain, and Sahai [AJS15] also showed that we can construct IO based on (non-succinct) *collusion-resistant* PKFE. Collusion-resistant functional encryption is functional encryption that can support a-priori unbounded number of functional keys.

Lin, Pass, Seth, and Telang [LPST16] showed that we can achieve single-key weakly succinct PKFE

from the LWE assumption and exponentially-efficient (XIO). XIO is a relaxed variant of IO such that the size of obfuscated circuits is sub-linear in the size of the input space. Bitansky, Nishimaki, Passelègue, and Wichs [BNPW20] subsequently showed that *collusion-resistant secret-key* functional encryption (SKFE) is powerful enough to yield IO if we additionally assume plain public key encryption.

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 [AS16] 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 Brakerski, Komargodski, and Segev [BKS18] and Komargodski and Segev [KS20] already showed that we can construct IO for somewhat restricted class of circuits based on SKFE via non-black-box constructions, it is still open whether we can construct IO for all circuits from SKFE bypassing the barrier with a non-black-box technique. Here, a non-black-box technique means the description of encryption algorithms is used in functional decryption keys. See Section 1.2 for more details.

1.2 Our Results

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

Theorem 1.1 (Informal). *Assuming that 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 [AS16].

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 [KS20]. 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 1.1 by using *puncturable SKFE*. The notion of puncturable SKFE was introduced by Bitansky and Vaikuntanathan [BV15] (See Section 1.3 for the difference between preliminary and updated versions [BV15, BV18]). 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.

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 1.2 (Informal). *Assuming that there exists collusion-resistant SKFE for all circuits, then there exists single-key weakly succinct puncturable SKFE for all circuits.*

¹More precisely, Asharov and Segev [AS16] introduced an extended model for black-box reductions to include a limited class of non-black-box reductions into their impossibility results. We will explain more on this impossibility result later.

Note that our definition of puncturable SKFE is a relaxed version of that proposed by Bitansky and Vaikuntanathan [BV15]. Our requirement for puncturable SKFE looks weaker than that of Bitansky and Vaikuntanathan. However, they are actually incomparable. In fact, we show that our puncturable SKFE defined in this paper is also sufficient as 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 [BV15].

Impacts on the hierarchy of cryptographic primitives Our result makes a progress on the study of IO and functional encryption as explained below. We have been classifying cryptographic primitives into two hierarchies MINICRYPT and CRYPTOMANIA since the beautiful work of Impagliazzo and Rudich [IR89] 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 [Imp95]. In MINICRYPT, one-way functions exist, but public-key encryption does not. In CRYPTOMANIA, public-key encryption also exists.

We started to consider a new hierarchy called OBFUSTOPIA after we found that many advanced cryptographic primitives can be constructed from IO. Although Garg, Pandey, Srinivasan, and Zhandry [GPSZ17] introduced the term OBFUSTOPIA, they 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 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 less clear than those of MINICRYPT and CRYPTOMANIA are. There has been significant effort to find out cryptographic primitives that are in OBFUSTOPIA. That is, we have been asking what kinds of cryptographic primitives imply the existence of IO. We know that sub-exponentially secure succinct PKFE exists in OBFUSTOPIA [BV18, AJ15]. We also know that sub-exponentially secure collusion-resistant SKFE is a bridge between CRYPTOMANIA and OBFUSTOPIA [BNPW20].

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 [AS16] gave a somewhat negative answer to this question. They showed that SKFE is unlikely to imply IO as long as we use black-box reductions. Although their impossibility result also holds for a limited class of non-black-box reductions, it does not exclude the possibility by using some non-black-box reduction outside the class. In particular, their impossibility result does not take self-feeding [GMM17] into account. Self-feeding means that an algorithm of a cryptographic primitive takes code descriptions of another algorithm of the cryptographic primitive as an input.⁴ Asharov and Segev also showed that SKFE does not imply any primitive outside MINICRYPT via the limited class of non-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. [BNPW20].

Bitansky et al. [BNPW20] 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. They circumvent the impossibility result by Asharov and Segev since they use the self-feeding technique.

² Komargodski, Moran, Naor, Pass, Rosen, and Yogev [KMN⁺14] proved that IO implies one-way functions under a mild complexity theoretic assumption. See the reference for the detail.

³Strictly speaking, it was known that there are stronger primitives (such as identity-based encryption) than public-key encryption before the candidate of obfuscation appeared [BPR⁺08, KY09]. Moreover, it is not clear whether public-key encryption implies fully homomorphic encryption or not so far.

⁴For example, in some works [AJ15, BV18, BKS18, BNPW20, KS20, GS16, LM16, KNT17a], we use a functional encryption scheme and generate a functional decryption key for a circuit that computes the encryption circuit of the functional encryption scheme.

Komargodski and Segev [KS20] 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 involves 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. 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. [GGH⁺16], 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 in this study.

Collusion-resistance and succinctness 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 [AJS15] showed how to construct collusion-resistant and succinct PKFE from a collusion-resistant and non-succinct one.⁵ In addition, Garg and Srinivasan [GS16] and Li and Micciancio [LM16] showed a transformation from single-key weakly succinct PKFE to collusion-resistant one with polynomial security loss.⁶ 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. From these results, collusion-resistance and succinctness are equivalent in PKFE.

On the other hand, the situation is different in SKFE. While we know how to construct collusion-resistant and succinct schemes from collusion-resistant and non-succinct ones [AJS15] similarly to PKFE, we do not know how to construct such schemes from succinct ones *even if sub-exponential security loss is permitted*.

There is a long line of research on collusion-resistant SKFE/PKFE for circuits, but all schemes based on standard assumptions are bounded collusion-resistant [SS10, Gvw12, GKP⁺13, AR17, Agr17, AV19].

As stated above, previous results including Theorem 1.1 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 the above 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

⁵Bitansky and Vaikuntanathan showed how to construct single-key succinct PKFE from a bounded collusion-resistant and *weakly succinct* PKFE [BV18].

⁶ Before their results, it was known that a single-key weakly succinct PKFE scheme implies a collusion-resistant and succinct one via IO [GGH⁺16, Wat15, BV15] though it incurs sub-exponential security loss.

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 1.3 (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.⁷

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).

Analogous to PKFE, we can transform collusion-resistant SKFE into collusion-resistant and succinct one [AJS15]. From this fact and Theorem 1.3, 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. In particular, we obtain the following corollary from Theorem 1.1 and 1.3.

Corollary 1.4 (Informal). *Assume that there exists sub-exponentially secure single-key weakly succinct SKFE for all circuits. Then, there exists IO for all circuits.*

1.3 More on Related Works

We explain the differences between the preliminary and journal versions of Bitansky and Vaikuntanathan [BV15, BV18].

In the preliminary version [BV15], they introduced puncturable SKFE (though they called symmetric-key instead of secret-key). They gave a proof sketch for IO from puncturable SKFE since their main IO construction is directly constructed from PKFE. Their puncturable SKFE is a natural functional encryption analog of puncturable pseudo random function (PRF) [SW14]. We denote their results by BV15.

In the journal version [BV18], they significantly revised the definition and called the new definition puncturable functional encryption (dropped “symmetric-key”). They showed how to construct puncturable functional encryption from PKFE and IO from puncturable functional encryption. The revised definition reflects the properties that their construction based on PKFE has. It is more complex than the original one. We denote their results by BV18.

We follow and relax the original definition of puncturable SKFE in this paper since it is simpler and fits to our constructions.

1.4 Organization

In Section 2, we provide the overview of our constructions. That is, IO based on collusion-resistant SKFE via puncturable SKFE and collusion-resistant SKFE based on single-key weakly succinct SKFE. In Section 3, we provide notations and definitions of cryptographic primitives. In Section 4, we formally define puncturable SKFE, and introduce security and efficiency notions for it. We also discuss the difference among our definition of puncturable SKFE, that of BV15 [BV15], and puncturable functional

⁷When 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.

encryption by BV18 [BV18] in Section 4. In Section 5, we show the construction of single-key non-succinct puncturable SKFE. In Section 6, we show how to transform single-key non-succinct puncturable SKFE into single-key weakly succinct one. In Section 7, we then show how to construct IO based on SKFE.

Next, in Section 8, we introduce some basic constructions and transformations for achieving collusion-resistant SKFE from single-key weakly succinct SKFE. In Section 9, we show a transformation from two SKFE schemes that support q_1 and q_2 functional key queries respectively to an SKFE scheme that supports $q_1 \cdot q_2$ functional key queries. Then, in Section 10, we show how to transform single-key weakly succinct SKFE into collusion resistant one by using constructions in Sections 8 and 9. That is, we prove Theorem 1.3. In Section 11, we show the additional feature of our transformation for SKFE.

2 Overview of Our Technique

Before we introduce formal definitions and constructions, we give a technical overview of our constructions in this section. This section has two parts. One is how to construct IO from SKFE. The other one is how to construct collusion-resistant SKFE from single-key weakly succinct SKFE.

IO from SKFE Our basic strategy is to replace PKFE in the construction of BV15 [BV15] 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. In addition, we show that such a relaxed variant of puncturable SKFE is sufficient for constructing IO. We follow the style of BV15 [BV15] since it fits to our puncturable SKFE.

We give an overview of the BV15 construction [BV15] 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.

Collusion-resistant SKFE from single-key weak succinctness Our basic strategy is using the transformation from single-key weakly succinct PKFE to collusion-resistant PKFE by Li and Micciancio [LM16]. However, the transformation does not work in the SKFE setting. Thus, we modify the transformation and combine it with single-ciphertext succinct and collusion-resistant SKFE, which can be instantiated with OWFs, via the hybrid encryption technique. We give an overview of this construction in Section 2.5.

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. Readers that are familiar with the BV15 construction [BV15] can skip this section.

Although Ananth and Jain [AJ15] also constructed IO from PKFE, it is not trivial whether our technique is applicable in the construction by Ananth and Jain since their IO construction is obtained via multi input functional encryption [GGG⁺14]. Their technique is significantly different from that of Bitansky and Vaikuntanathan [BV15, BV18].

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 [BS18] 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 PRF in the actual construction.

The construction of $i\mathcal{O}_i$ 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 [BS18]. To obfuscate a circuit C with i -bit input, it first generates a key pair $(\text{PK}_i, \text{MSK}_i)$ 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 CT_0^{ske} and CT_1^{ske} of plaintext C under independent keys K_0 and K_1 , respectively. C^* expects as an input not only an i -bit string \mathbf{x}_i but also an SKE key K_b . On those inputs, C^* first obtains C by decrypting CT_b^{ske} by K_b and outputs $U(C, \mathbf{x}_i) = C(\mathbf{x}_i)$, where $U(\cdot, \cdot)$ is a 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 \mathbf{x}_{i-1} , it outputs ciphertexts $\text{Enc}(\text{PK}_i, (\mathbf{x}_{i-1} \| 0, K_b))$ and $\text{Enc}(\text{PK}_i, (\mathbf{x}_{i-1} \| 1, K_b))$, where Enc is the encryption algorithm of PKFE. The resulting obfuscation of C is a tuple $(\text{sk}_{C^*}, i\mathcal{O}_{i-1}(E_{i-1}))$. 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^*	// Description of (simplified) E_{i-1}
Hard-Coded Constants: $\text{CT}_0^{\text{ske}}, \text{CT}_1^{\text{ske}}$. Input: \mathbf{x}_i, K_b	Hard-Coded Constants: PK_i, K_b . Input: $\mathbf{x}_{i-1} \in \{0, 1\}^{i-1}$
<ol style="list-style-type: none"> 1. Compute $C = D(K_b, \text{CT}_b^{\text{ske}})$. 2. Return $U(C, \mathbf{x}_i)$. 	<ol style="list-style-type: none"> 1. Compute $\text{CT}_{i, \mathbf{x}_i} \leftarrow \text{Enc}(\text{PK}_i, (\mathbf{x}_{i-1} \ \mathbf{x}_i, K_b))$. 2. Output $\text{CT}_{i,0}$ and $\text{CT}_{i,1}$.

Figure 1: Descriptions of (simplified) C^* and E_{i-1} .

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

Consequently, by using this bit extension construction, the obfuscation of a circuit C with n -bit input consists of n functional keys $\text{sk}_1, \dots, \text{sk}_n$ 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 $\mathbf{x}_n = x_1 \cdots x_n \in \{0, 1\}^n$, we can first compute a ciphertext of \mathbf{x}_n by repeatedly decrypting a ciphertext of $\mathbf{x}_{i-1} = x_1 \cdots x_{i-1}$ by sk_{i-1} and obtaining a ciphertext of $\mathbf{x}_i = x_1 \cdots x_i$ for every $i \in \{2, \dots, n\}$. We can finally obtain $C(\mathbf{x}_n)$ by decrypting the ciphertext of \mathbf{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 Canetti, Lin, Tessaro, and Vaikuntanathan [CLTV15], 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. See Section 2.4 for the concrete security loss.

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 $i\mathcal{O}_{i-1}(E_{i-1})$ and this encryption circuit is public when we use the security of PKFE under the key pair (PK_i, 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. [BGI⁺12]. In fact, we do not know how to prove the security of the IO based on SKFE because we cannot hardwire a secret encryption key into obfuscated encryption circuit in security reductions. This is why they adopted PKFE as their building block. 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* [BV15].

In puncturable SKFE defined by Bitansky and Vaikuntanathan [BV15], there are a puncturing algorithm Punc and a punctured encryption algorithm PEnc in addition to algorithms of standard SKFE. We can generate a punctured master secret key $MSK^*\{m_0, m_1\}$ 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

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

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

$$(MSK^*\{m_0, m_1\}, \text{Enc}(MSK, m_0)) \stackrel{\text{c}}{\approx} (MSK^*\{m_0, m_1\}, \text{Enc}(MSK, m_1))$$

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

Bitansky and Vaikuntanathan informally proved that single-key weakly succinct puncturable SKFE is also a sufficient building block for their IO construction [BV15]. 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 how to 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 BV15 [BV15] 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 [SS10]. 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 generate a garbled circuit \tilde{U}_m with labels $\{L_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}$ by garbling a 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 $(\tilde{U}_m, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$.

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 $(\tilde{U}_m, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$ 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 $\tilde{U}_m(\{L_{j,f[j]}\}_{j \in [s]}) = U(f, m) = f(m)$.

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 $(\tilde{U}_m, \{L_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$ and encrypt each label by using a puncturable PRF value. That is, $c_{j,\alpha} \leftarrow L_{j,\alpha} \oplus F_{S_{j,\alpha}}(\text{tag})$, where F is puncturable PRF and tag is a public tag chosen in some way.

Hereafter, we switch the standard syntax of functional encryption to a tag-based one since our actual construction is tag-based. In this case, we can generate a punctured master secret key $\text{MSK}^*\{\text{tag}\}$ 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, $\text{Enc}(\text{MSK}, \text{tag}, m) = (\tilde{U}_m, \{L_{j,\alpha} \oplus F_{S_{j,\alpha}}(\text{tag})\}_{j \in [s], \alpha \in \{0,1\}})$. A punctured master secret key $\text{MSK}^*\{\text{tag}\}$ consists of $2s$ puncturable PRF keys $\{S_{j,\alpha}^*\{\text{tag}\}\}_{j \in [s], \alpha \in \{0,1\}}$ all of which are punctured at tag .

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

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

for any tag tag and tag' such that $\text{tag} \neq \text{tag}'$, 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 $\text{Enc}(\text{MSK}, \text{tag}, m_0)$ and $\text{Enc}(\text{MSK}, \text{tag}, m_1)$ are indistinguishable for adversaries that have $\text{MSK}^*\{\text{tag}\}$ based on the security of puncturable PRF. In other words, it satisfies semantic security at punctured tag.

This formalization is a slightly different from that proposed by BV15 [BV15] in the sense that we allow the use of tags. Nevertheless, our formalization of puncturable SKFE is also 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 BV15. 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 a weakly succinct scheme while preserving security and functionality.

We extend the work by Kitagawa, Nishimaki, and Tanaka [KNT21] 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 the size of the encryption circuit and puncturable 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 in the size of the largest function in a function class.

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 a 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 [BV18] and that proposed by Ananth, Jain, and Sahai [AJS15]. We give an illustration of our construction path in Figure 2.

The general picture is similar to that of Kitagawa et al. [KNT21] and we can accomplish the latter transformation based on a known technique, but there is a technical hurdle in the former transformation. The 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.

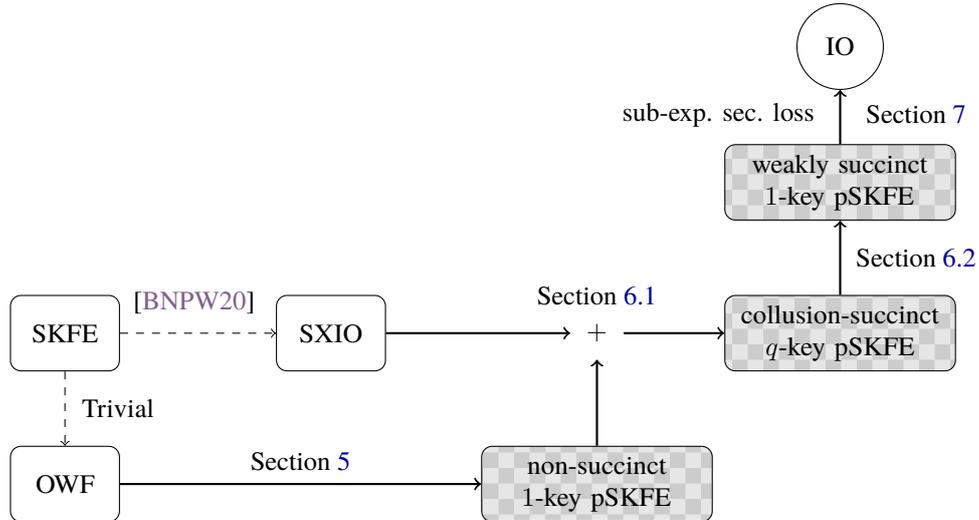


Figure 2: Illustration of our construction path. pSKFE denotes *tag-based* puncturable SKFE. Dashed lines denote known or trivial implications. White boxes denote our ingredients or goal. Checker board background boxes denote our core schemes. A transformation from an object in a rectangle to one in a rectangle incurs only polynomial security loss. A transformation from an object in a rectangle to one in a circle incurs sub-exponential security loss (denoted by sub-exp. sec. loss).

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 Bitansky et al. [BNPW20].

SXIO is a relaxed variant of IO. SXIO is required that, given a circuit C with n -bit input, it runs in $2^{\gamma n} \cdot \text{poly}(\lambda, |C|)$ -time, where γ is a constant smaller 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 [BNPW20, BKS18].

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 allow stateful constructions. If we have q master secret keys $\text{MSK}_1, \dots, \text{MSK}_q$, 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 tag, we generate q punctured master secret keys $\text{MSK}_i^*\{\text{tag}\}$ for every $i \in [q]$ all of which are punctured at 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 [LPST16, BNPW20, KNT21]. Let sxiO be SXIO. We set a ciphertext as a circuit computing q ciphertexts obfuscated by sxiO instead of setting it as q ciphertexts themselves. Concretely, we obfuscate the following circuit $E_{1\text{Key}}$ using sxiO . $E_{1\text{Key}}$ 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 master secret key of this scheme is now

<p>Hard-Coded Constants: S, tag, m.</p> <p>Input: $i \in [q]$</p> <ol style="list-style-type: none"> 1. Compute $r_{\text{Setup}}^i \leftarrow F_S(i)$. 2. Compute $\text{MSK}_i \leftarrow \text{Setup}(1^\lambda; r_{\text{Setup}}^i)$. 3. Return $\text{CT}_i \leftarrow \text{Enc}(\text{MSK}_i, \text{tag}, m)$. 	<p>// Description of (simplified) $E_{1\text{Key}}$</p>
--	--

Figure 3: Description of (simplified) $E_{1\text{Key}}$.

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 $E_{1\text{Key}}$, see Figure 7 in Section 6.1.

The size of $E_{1\text{Key}}$ is independent of q since $E_{1\text{Key}}$ consists of one PRF evaluation and setup and encryption procedure of the single-key scheme.⁸ Therefore, the time needed to compute $\text{sxiO}(E_{1\text{Key}})$ is bounded by $2^{\gamma \log q} \cdot \text{poly}(\lambda, |m|) = q^\gamma \cdot \text{poly}(\lambda, |m|)$ 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

⁸Strictly speaking, the domain of PRF is $[q]$, and thus the size of $E_{1\text{Key}}$ 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.

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 $P_{1\text{Key}}$. $P_{1\text{Key}}$ 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 $E_{1\text{Key}}$. On input $i \in [q]$, $P_{1\text{Key}}$ first generates MSK_i pseudorandomly from S and i , and then outputs a punctured master secret key $\text{MSK}_i^*\{\text{tag}\}$ of the single-key scheme. For the formal description of $P_{1\text{Key}}$, see Figure 8 in Section 6.1.

<p style="text-align: center;">// Description of (simplified) $P_{1\text{Key}}$</p> <p>Hard-Coded Constants: S, tag.</p> <p>Input: $i \in [q]$</p> <ol style="list-style-type: none"> 1. Compute $r_{\text{Setup}}^i \leftarrow F_S(i)$. 2. Compute $\text{MSK}_i \leftarrow \text{Setup}(1^\lambda; r_{\text{Setup}}^i)$. 3. Return $\text{MSK}_i^*\{\text{tag}\} \leftarrow \text{Punc}(\text{MSK}_i, \text{tag})$. 	<p style="text-align: center;">// Description of (simplified) $PE_{1\text{Key}}$</p> <p>Hard-Coded Constants: $\text{MSK}^*\{\text{tag}\}, \text{tag}', m$.</p> <p>Input: $i \in [q]$</p> <ol style="list-style-type: none"> 1. Parse $\text{sxiO}(P_{1\text{Key}}) \leftarrow \text{MSK}^*\{\text{tag}\}$. 2. Compute $\text{MSK}_i^*\{\text{tag}\} \leftarrow \text{sxiO}(P_{1\text{Key}})(i)$. 3. Return $\text{CT}_i \leftarrow \text{PEnc}(\text{MSK}_i^*\{\text{tag}\}, \text{tag}', m)$.
---	--

Figure 4: Descriptions of (simplified) $P_{1\text{Key}}$ and $PE_{1\text{Key}}$.

In addition, we define the punctured encryption algorithm as follows. On input $\text{MSK}^*\{\text{tag}\}$ that is $\text{sxiO}(P_{1\text{Key}})$, tag tag' , and message m , the punctured encryption algorithm obfuscates the following circuit $PE_{1\text{Key}}$ using sxiO and outputs the obfuscated circuit. $PE_{1\text{Key}}$ has hardwired $\text{MSK}^*\{\text{tag}\}$, tag' , and m , and on input $i \in [q]$, it first generates the i -th punctured key $\text{MSK}_i^*\{\text{tag}\}$ by feeding i into $\text{MSK}^*\{\text{tag}\} = \text{sxiO}(PE_{1\text{Key}})$, and then outputs a ciphertext of m under $\text{MSK}_i^*\{\text{tag}\}$ and tag' using the punctured encryption algorithm of the single-key scheme. If the compression factor of sxiO is *sufficiently small*, we ensure that the running time of this punctured encryption algorithm is sub-linear in q . For the formal description of $PE_{1\text{Key}}$, see Figure 9 in Section 6.1.

We can prove the semantic security at punctured tag by the punctured programming technique proposed by Sahai and Waters [SW14]. 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 $E_{1\text{Key}}$ and $PE_{1\text{Key}}$, 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

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

holds for any tag tag and tag' such that $\text{tag} \neq \text{tag}'$, and message m , where $\stackrel{\text{c}}{\approx}$ 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 $E_{1\text{Key}}$ and $PE_{1\text{Key}}$ 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 a weakly succinct scheme As stated earlier, we can in turn transform a collusion-succinct scheme into a weakly succinct one using decomposable randomized encoding. This transformation is based on those proposed by Bitansky and Vaikuntanathan [BV18] and Ananth et al. [AJS15].

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 by BV15 [BV15]. It does not depend on which functionality preserving property puncturable SKFE satisfies. Recall that, in their construction, a key pair (PK_i, 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 x_{i-1} , it outputs a ciphertext of $(x_{i-1}||x_i, K)$ for $x_i \in \{0, 1\}$ under MSK_i and a tag x_{i-1} , that is, $\text{Enc}(MSK_i, x_{i-1}, (x_{i-1}||x_i, K))$ 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 $MSK_i^*\{j\}$ punctured at $j \in \{0, 1\}^{i-1}$ and a ciphertext of $(j||x_i, K)$ for $x_i \in \{0, 1\}$, where 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 $i\mathcal{O}_{i-1}(E_{i-1})$. C^* has hardwired two SKE ciphertexts CT_0^{ske} and CT_1^{ske} of C_0 under independent keys K_0 and K_1 . On i -bit input x_i and SKE key K_b , C^* first obtains C by decrypting CT_b^{ske} by K_b and outputs $C(x_i)$.

In the next hybrid \mathcal{H}_1 , we change how CT_1^{ske} hardwired in C^* is generated. Concretely, we generate CT_1^{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 x_{i-1} , it outputs a ciphertext of $(x_{i-1} \| x_i, K_1)$ instead of $(x_{i-1} \| x_i, K_0)$ for $x_i \in \{0, 1\}$ under MSK_i and a tag 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 Canetti et al. [CLTV15].

Concretely, we consider $2^{i-1} + 1$ intermediate hybrid experiments $\mathcal{H}_{1,j}$ for $j \in \{0, \dots, 2^{i-1}\}$ 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 $\mathbf{j} \in \{0, 1\}^{i-1}$, it outputs ciphertexts of $(\mathbf{j} \| x_i, K_1)$ instead of $(\mathbf{j} \| x_i, K_0)$ for $x_i \in \{0, 1\}$, where \mathbf{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 x_{i-1} ,

- if $x_{i-1} < \mathbf{j}$, it outputs a ciphertext of $(x_{i-1} \| x_i, K_1)$ for $x_i \in \{0, 1\}$ under MSK_i and a tag x_{i-1} .
- Otherwise, it outputs a ciphertext of $(x_{i-1} \| x_i, K_0)$ for $x_i \in \{0, 1\}$ under MSK_i and a tag 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 $(\mathbf{j} \| x_i, K_0)$ under MSK_i and a tag \mathbf{j} for $x_i \in \{0, 1\}$ in E_{i-1} . In addition, we replace hardwired MSK_i in E_{i-1} with $\text{MSK}_i^* \{\mathbf{j}\}$ that is a master secret key punctured at a tag \mathbf{j} . On $(i-1)$ -bit input x_{i-1} ,
 - if $x_{i-1} = \mathbf{j}$, E_{i-1} outputs hardwired ciphertexts of $(\mathbf{j} \| x_i, K_0)$ for $x_i \in \{0, 1\}$.
 - if $x_{i-1} \neq \mathbf{j}$, it generates ciphertexts of $(x_{i-1} \| x_i, K_\beta)$ under $\text{MSK}_i^* \{\mathbf{j}\}$ and a tag x_{i-1} and outputs them, where $\beta = 1$ if $x_{i-1} < \mathbf{j}$ 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 $(\mathbf{j} \| x_i, K_1)$ 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 $(\mathbf{j} \| x_i, K_1)$ 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 $j \in \{0, \dots, 2^{i-1} - 1\}$. 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 BV15.

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 $k \in \{0, \dots, 2^{i-1}\} \setminus \{j\}$ 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 $\mathbf{k} \in \{0, 1\}^{i-1}$, it outputs ciphertexts under $\text{MSK}_i^*\{j\}$ instead of ciphertexts under MSK_i , where \mathbf{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 $\text{MSK}_i^*\{j\}$ in addition to MSK_i , K_0 , and K_1 . On $(i-1)$ -bit input \mathbf{x}_{i-1} , it runs as follows.

- If $\mathbf{x}_{i-1} < j$, it sets $\beta = 1$ and $\beta = 0$ otherwise.
- If $\mathbf{x}_{i-1} < k$ and $\mathbf{x}_{i-1} \neq j$, it outputs a ciphertext of $(\mathbf{x}_{i-1} \| x_i, K_\beta)$ under $\text{MSK}_i^*\{j\}$ and a tag \mathbf{x}_{i-1} , that is, $\text{PEnc}(\text{MSK}_i^*\{j\}, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| x_i, K_\beta))$ for $x_i \in \{0, 1\}$.
- Otherwise ($\mathbf{x}_{i-1} \geq k$ or $\mathbf{x}_{i-1} = j$), it outputs a ciphertext of $(\mathbf{x}_{i-1} \| x_i, K_\beta)$ under MSK_i and a tag \mathbf{x}_{i-1} , that is, $\text{Enc}(\text{MSK}_i, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| x_i, K_\beta))$ 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 $k \in \{0, \dots, 2^{i-1}\} \setminus \{j\}$ by three steps again based on indistinguishability of functionality under puncturing.

- (1) We hardwire ciphertexts of $(\mathbf{k} \| x_i, K_\beta)$ under MSK_i and a tag \mathbf{k} , that is, $\text{Enc}(\text{MSK}_i, \mathbf{k}, (\mathbf{k} \| x_i, K_\beta))$ 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 $(\mathbf{k} \| x_i, K_0)$ for $x_i \in \{0, 1\}$ if the input is \mathbf{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 $(\mathbf{k} \| x_i, K_\beta)$ under $\text{MSK}_i^*\{j\}$, that is $\text{PEnc}(\text{MSK}_i^*\{j\}, \mathbf{k}, (\mathbf{k} \| x_i, K_\beta))$ 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 $(\mathbf{k} \| x_i, K_1)$ for $x_i \in \{0, 1\}$. Namely, we change E_{i-1} so that on input \mathbf{k} , E_{i-1} generates ciphertexts of \mathbf{k} under $\text{MSK}_i^*\{j\}$ 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 $k \in \{0, \dots, 2^{i-1}\} \setminus \{j\}$. 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 [BV15, BV18] and the case where puncturable SKFE satisfies functionality preserving under puncturing. Our security proof incurs roughly 2^{2^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 [BV15, BV18].

2.5 Technical Overview for Achieving Collusion-Resistance

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 [AJ15, BV15, BKS18, GS16, LM16], but we cannot directly use such techniques as we see below.

Known techniques and their limitations It was shown that single-key weakly succinct PKFE implies collusion-resistant PKFE by Garg and Srinivasan [GS16] and Li and Micciancio [LM16]. Our starting point is the technique by Li and Micciancio. 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. That is, we can obtain $q_1 \cdot q_2$ -key PKFE scheme from q_1 -key and q_2 -key PKFE schemes via the PRODUCT construction. By applying the PRODUCT construction k times iteratively, we can construct a λ^k -key PKFE scheme from a λ -key PKFE scheme, where λ is the security parameter. 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 is sublinear in the size of functions and a polynomial of k , λ , and the message length (does not incur exponential blow-up). 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.⁹

The PRODUCT construction involves *the chaining of re-encryption by functional decryption keys* used in many previous works [AJ15, BV15, BKS18, GS16]. We saw an example in Section 2.1. 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. The main cause of the difficulty is the fact that we have to generate 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.

Brakerski, Komargodski, and Segev [BKS18] introduced a new re-encryption technique by functional decryption keys in the context of multi-input SKFE [GGG⁺14]. They showed that we can overcome the difficulty above by using the security notion of function privacy [BS18].

By function privacy, we can hide the information about a master-secret key embedded in a re-encryption circuit $\mathcal{E}[\text{MSK}^*]$. In the normal mode, $\mathcal{E}[\text{MSK}^*]$ outputs a ciphertext of an input under MSK^* . We embed a post-re-encrypted ciphertext ct_{post} (inner encryption) as a trapdoor into a pre-re-encrypted ciphertext

⁹Li and Micciancio proceeded with the above series of transformations via a index-based variant of PKFE, where each functional key is indexed by a number, and thus the resulting collusion-resistant scheme is also a index-based scheme. Therefore, after achieving collusion-resistance, they converted the index-based PKFE scheme into a standard PKFE scheme. For simplicity, we ignore the issue of index-based constructions in this overview.

ct_{pre} (outer encryption) in advance in the simulation for the security proof. We can switch the normal mode to the trapdoor mode by using the security of outer encryption. Embedding ct_{post} is crucial to preserve the output equivalence of \mathcal{E} . In the trapdoor mode, \mathcal{E} directly outputs the embedded ct_{post} and we can remove MSK^* from the re-encryption circuit \mathcal{E} . Thus, we can reduce the security of the resulting scheme to that of the inner encryption 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. Even if we embed ciphertexts in a functional key, we cannot avoid the issue of nested-ciphertext embedding 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 a constant number of times*. This is not sufficient for our goal since we must apply our new PRODUCT construction $\omega(1)$ times to achieve collusion-resistant SKFE.

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 [ABS15] showed that a hybrid encryption construction is useful in designing adaptively secure functional encryption from selectively secure one without any additional assumption. Brakerski et al. [BKS16] also used a hybrid encryption construction to achieve 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 generate *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 [Yao86]. A functional decryption key is a garbled circuit of f with encrypted labels by a standard secret-key encryption scheme.¹⁰ A ciphertext consists of a randomly masked message and keys of the 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 1CT

¹⁰Each pair of labels is shuffled by a random masking.

is selectively secure. In fact, this construction is a flipped variant of the single-key SKFE by Sahai and Seyalioglu [SS10].

We then modify the SKFE variant of the hybrid construction proposed by Ananth et al. [ABSV15].¹¹ We use 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 1CT), which are independent of the length of a message to be encrypted. A one-time encryption key (short and fixed length) of 1CT is encrypted by SKFE.

That is, by this hybrid construction, a real message part is shifted onto 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”. See Section 10.1 for more detailed intuition.

The third property of 1CT is also important. The size of a functional key of 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 1CT for f . A simplified description of G is below.¹² Due to the third property of 1CT, the hybrid construction preserves weak

Hard-Coded Constants: f .	// Description of (simplified) G
Input: 1CT.MSK	
1. Return $1CT.sk_f \leftarrow 1CT.KG(1CT.MSK, f)$.	

succinctness.

Moreover, from the above construction of the key generation algorithm, the number of issuable functional keys of the resulting scheme is the minimum of those of building block SKFE and 1CT. Therefore, since 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 $\omega(1)$ times via the size-shifting and construct a collusion-resistant SKFE scheme based only on a single-key (weakly) succinct SKFE scheme.¹³

Figure 5 illustrates how to construct our building blocks. An illustration of our sandwiched size-shifting procedure is described in Figure 6.

3 Preliminaries

We define some notations and cryptographic primitives.

Notations We write $x \xleftarrow{r} X$ to denote that an element x is chosen from a finite set X uniformly at random and $y \leftarrow A(x; r)$ to denote that the output of an algorithm A on an input x and a randomness r is

¹¹Their goal is to construct an adaptively secure scheme. They used *adaptively secure* single-ciphertext functional encryption that is *non-succinct* as data encapsulation mechanism.

¹²We ignore the issue of the randomness for the key generation. We use a PRF to solve this issue in the actual scheme.

¹³ 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.

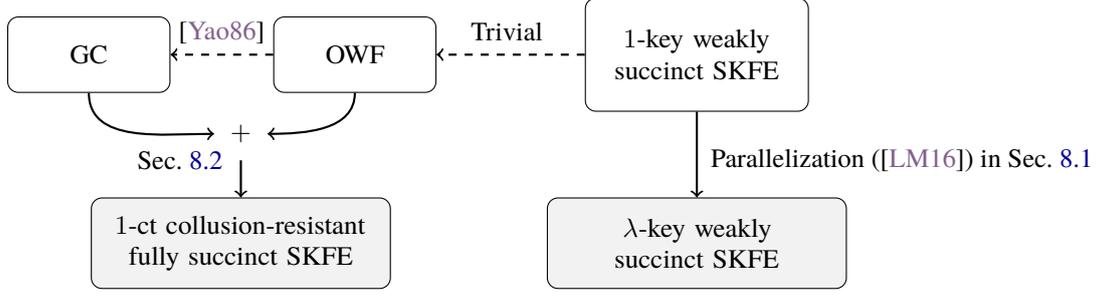


Figure 5: An illustration of our building blocks. Gray boxes denote our core schemes used in our iterated construction in Figure 6.

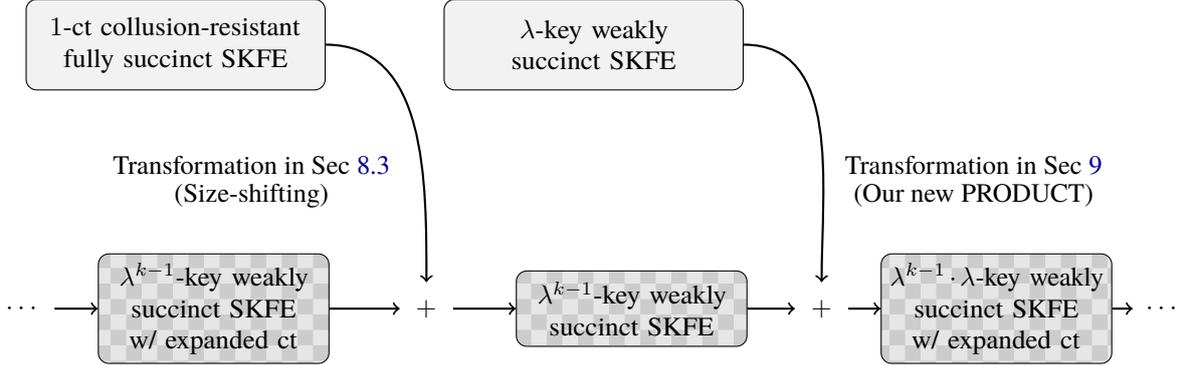


Figure 6: An illustration of our iteration technique, in which our size-shifting procedure is sandwiched. For k -th iteration, first, we apply the size-shifting procedure to a λ^{k-1} -key weakly succinct SKFE scheme with expanded ciphertexts incurred by nested-ciphertext-embedding (the result of $(k-1)$ -th iteration). Second, we apply our new PRODUCT construction to increase the number of issuable keys.

assigned to y . When there is no need to write the randomness explicitly, we omit it and simply write $y \leftarrow A(x)$. Let $y := z$ denote that y is set, defined, or substituted by z . For strings x and y , $x||y$ denotes the concatenation of x and y . Throughout this paper, λ denotes a security parameter and 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 $f(\lambda) = \text{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. PPT stands for probabilistic polynomial time. Let $[n]$ and $[x, y]$ denote the set of integers $\{1, \dots, n\}$ and $\{x, x+1, \dots, y\}$, respectively.

3.1 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 3.1 (Pseudo-Random Function). Let $\{F_K : \{0, 1\}^{\ell_1} \rightarrow \{0, 1\}^{\ell_2} \mid K \in \{0, 1\}^\lambda\}$ be a family of polynomially computable functions, where ℓ_1 and ℓ_2 are some polynomials of λ . We say that F is a pseudo-random function (PRF) family if for any PPT distinguisher \mathcal{A} , there exists $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, it holds that

$$\text{Adv}_{\mathcal{A}}^{\text{prf}}(\lambda) := \left| \Pr \left[\mathcal{A}^{F_K(\cdot)}(1^\lambda) = 1 \mid K \xleftarrow{r} \{0, 1\}^\lambda \right] - \Pr \left[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1 \mid R \xleftarrow{r} \mathcal{U} \right] \right| = \text{negl}(\lambda),$$

where \mathcal{U} is the set of all functions from $\{0, 1\}^{\ell_1}$ to $\{0, 1\}^{\ell_2}$. We further say that F is ϵ -secure, for some concrete negligible function $\epsilon(\cdot)$, if for any PPT \mathcal{A} the above advantage is smaller than $\epsilon(\lambda)^{\Omega(1)}$.

Theorem 3.2 ([GGM86]). *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.*

Definition 3.3 (Puncturable PRF). *A puncturable PRF consists of a tuple of algorithms $\text{PPRF} = (F, \text{Punc})$ where $\{F_K : \{0, 1\}^{\ell_1} \rightarrow \{0, 1\}^{\ell_2} \mid K \in \{0, 1\}^\lambda\}$ is a PRF family and satisfies the following two conditions. Note that ℓ_1 and ℓ_2 are polynomials of λ .*

Functionality preserving under puncturing: *For any polynomial-size set $S \subseteq \{0, 1\}^{\ell_1}$ and any $x \in \{0, 1\}^{\ell_1} \setminus S$, it holds that*

$$\Pr[F_K(x) = F_{K\{S\}}(x) \mid K \xleftarrow{\$} \{0, 1\}^\lambda, K\{S\} \leftarrow \text{Punc}(K, S)] = 1.$$

Pseudorandom at punctured points: *For any polynomial-size set $S \subseteq \{0, 1\}^{\ell_1}$ and any PPT distinguisher \mathcal{A} , there exists $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, it holds that*

$$|\Pr[\mathcal{A}(F_{K\{S\}}, \{F_{K\{S\}}(x_i)\}_{x_i \in S}) = 1] - \Pr[\mathcal{A}(F_{K\{S\}}, U^k) = 1]| = \text{negl}(\lambda),$$

where $K \xleftarrow{\$} \{0, 1\}^\lambda$, $K\{S\} \leftarrow \text{Punc}(K, S)$ and U denotes the uniform distribution over $\{0, 1\}^{\ell_2}$. We further say that PPRF is ϵ -secure, for some concrete negligible function $\epsilon(\cdot)$, if for any PPT \mathcal{A} the above indistinguishability gap is smaller than $\epsilon(\lambda)^{\Omega(1)}$.

Theorem 3.4 ([GGM86, BW13, BGI14, KPTZ13]). *If one-way function exist, then for any polynomials $\ell_1(\lambda)$ and $\ell_2(\lambda)$, there exists a puncturable PRF that maps ℓ_1 -bits to ℓ_2 -bits.*

Definition 3.5 (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 $\tilde{m} \in \{\perp\} \cup \mathcal{M}$. This algorithm is deterministic.

Correctness: *We require $D(K, E(K, m)) = m$ 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 $K \xleftarrow{\$} \{0, 1\}^\lambda$ and chooses the challenge bit $b \xleftarrow{\$} \{0, 1\}$. 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 $c \leftarrow E(K, m_b)$.
3. \mathcal{A} outputs $b' \in \{0, 1\}$.

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

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

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

Theorem 3.6 ([LR88]). *If there exist one-way functions, there exists CPA-secure SKE.*

Definition 3.7 (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 $(\text{Grbl}, \text{Eval})$ 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 \tilde{C} , together with $2n$ labels $\{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}$.
- The evaluation algorithm, given a garbled circuit \tilde{C} and n labels $\{L_j\}_{j \in [n]}$, outputs y .

Correctness: We require $\text{Eval}(\tilde{C}, \{L_{j,x_j}\}_{j \in [n]}) = C(x)$ for every $n \in \mathbb{N}$, $C \in \mathcal{C}_n$, and $x \in \{0,1\}^n$, where $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C)$ 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 $b \xleftarrow{r} \{0,1\}$ 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 $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C)$ and returns $(\tilde{C}, \{L_{j,x_j}\}_{j \in [n]})$ to \mathcal{A} . Otherwise, the challenger returns $(\tilde{C}, \{L_j\}_{j \in [n]}) \leftarrow \text{Sim}(1^\lambda, |C|, C(x))$.
4. \mathcal{A} outputs $b' \in \{0,1\}$.

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

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

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 $\text{Adv}_{\text{GC}, \mathcal{A}, \text{Sim}}^{\text{GC}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

Theorem 3.8 ([Yao86, BHR12, LP09]). *If there exist one-way functions, there exists secure garbling scheme for any polynomial size circuits.*

Definition 3.9 (Decomposable Randomized Encoding). A decomposable randomized encoding scheme RE for a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ consists of two polynomial-time algorithms (RE.E, RE.D).

RE.E($1^\lambda, f$) takes as inputs the security parameter 1^λ and a function f , and outputs a new function \hat{f} where $\hat{f} : \{0,1\}^n \times \{0,1\}^\rho \rightarrow \{0,1\}^\mu$ and ρ is the length of randomness used to compute \hat{f} .

RE.D(\hat{y}) takes as an input \hat{y} and outputs y .

A randomized encoding scheme satisfies the following properties. Let $s_{\hat{f}}$ (resp. s_f) denote the size of the circuit computing \hat{f} (resp. f).

Correctness: For any λ, f , and x , it holds that $\Pr[f(x) = \text{RE.D}(\hat{f}(x; r)) \mid r \xleftarrow{r} \{0,1\}^\rho, \hat{f} \leftarrow \text{RE.E}(1^\lambda, f)] = 1$.

Decomposability: Computation of \hat{f} can be decomposed into computation of μ functions. That is, $\hat{f}(x; r) = (\hat{f}_1(x; r), \dots, \hat{f}_\mu(x; r))$, where each \hat{f}_i depends on $x \in \{0,1\}^n$ and $r \in \{0,1\}^\rho$. We write $\hat{f}(x; r) = (\hat{f}_1(x; r), \dots, \hat{f}_\mu(x; r))$. Parameters ρ and μ are bounded by $s_f \cdot \text{poly}(\lambda, n)$.

Security: Let RE.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 $b \xleftarrow{r} \{0,1\}$ and sends the security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends a function f and input $x \in \{0,1\}^n$ to the challenger.

3. If $b = 0$, the challenger computes $\{\widehat{f}_i\}_{i=1}^\mu \leftarrow \text{RE.E}(1^\lambda, f)$ and $\{\widehat{f}_i(x; r)\}_{i=1}^\mu$ by choosing $r \xleftarrow{\$} \{0, 1\}^\rho$, and returns $\{\widehat{f}_i(x; r)\}_{i=1}^\mu$ to \mathcal{A} . Otherwise, the challenger returns $\{\widehat{y}\}_{i=1}^\mu \leftarrow \text{RE.Sim}(1^\lambda, 1^{|f|}, f(x))$.
4. \mathcal{A} outputs a guess $b' \in \{0, 1\}$.

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

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

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

Theorem 3.10 ([Yao86, AIK06]). *If there exist one-way function, there exists secure decomposable randomized encoding for all polynomial size functions.*

3.2 Secret-Key Functional Encryption

We review the definition of standard secret-key functional encryption (SKFE) and its variant in this section.

Definition 3.11 (Secret-key functional encryption). *An SKFE scheme SKFE is a tuple of four PPT algorithms (Setup, KG, Enc, 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 $\tilde{m} \in \{\perp\} \cup \mathcal{M}$.

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

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

Definition 3.12 (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.*

Initialization *First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \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 . Next, the challenger generates a master secret key $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses the challenge bit $b \xleftarrow{\$} \{0, 1\}$. Finally, the challenger generates ciphertexts $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, m_b^\ell)$ ($\ell \in [p]$) and sends them to \mathcal{A} .*

Key queries *\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 $sk_f \leftarrow \text{KG}(\text{MSK}, f)$, and returns sk_f to \mathcal{A} . Here, f needs to satisfy $f(m_0^\ell) = f(m_1^\ell)$ for all $\ell \in [p]$.*

Final phase *\mathcal{A} outputs $b' \in \{0, 1\}$.*

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

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

\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 $\text{Adv}_{\text{SKFE},\mathcal{A}}^{\text{sm-mp}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

We further say that an SKFE scheme is (poly, ϵ) -selective-message message private if it is (q, ϵ) -selective-message function private for any polynomial q . Note that a (poly, ϵ) -selective-message function private SKFE scheme is also said to be ϵ -secure collusion-resistant.

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

Definition 3.13 (Selective-message function privacy). Let SKFE be an SKFE scheme whose message space and function space are \mathcal{M} and \mathcal{F} , respectively. Let q be a fixed polynomial of λ . We define the selective-message function privacy game between a challenger and an adversary \mathcal{A} as follows.

Initialization First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \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 . Next, the challenger generates a master secret key $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Finally, the challenger generates ciphertexts $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, m_b^\ell)$ ($\ell \in [p]$) and sends them to \mathcal{A} .

\mathcal{A} may adaptively make key queries q times at most.

Key queries For a key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ from \mathcal{A} , the challenger generates $sk_f \leftarrow \text{KG}(\text{MSK}, f_b)$, and returns sk_f to \mathcal{A} . Here, f_0 and f_1 need to be the same size and satisfy $f_0(m_0^\ell) = f_1(m_1^\ell)$ for all $\ell \in [p]$.

Final phase \mathcal{A} outputs $b' \in \{0, 1\}$.

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

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

For a negligible function $\epsilon(\cdot)$, We say that SKFE is (q, ϵ) -selective-message function private if for any PPT \mathcal{A} , we have $\text{Adv}_{\text{SKFE},\mathcal{A}}^{\text{sm-fp}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

We say that an SKFE scheme is (poly, ϵ) -selective-message function private if it is (q, ϵ) -selective-message function private for any polynomial q . Note that a (poly, ϵ) -selective-message function private SKFE scheme is also said to be ϵ -secure collusion-resistant.¹⁴

From message privacy to function privacy If \mathcal{A} is allowed to send only a function f (not a pair of functions) that satisfies $f(m_0^\ell) = f(m_1^\ell)$ for all ℓ at each key query in Definition 3.13, then we call the security selective-message message privacy. A transformation from message private to function private SKFE is known.

Theorem 3.14 ([BS18]). *If there exists a (q, δ) -selective-message message private SKFE scheme where q is any fixed polynomial, there exists a (q, δ) -selective-message function private SKFE scheme.*

¹⁴ Collusion-resistance generally does not require function privacy. Not only function private schemes but also message private schemes are referred to as collusion-resistant if they are secure against a-priori unbounded polynomial number of functional key queries.

From selective security to adaptive security If \mathcal{A} can send the target messages $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [p]}$ adaptively in Definitions 3.12 and 3.13, then we call the security adaptive-message message/function privacy. There is a transformation from selective security to adaptive security.

Theorem 3.15 ([ABSV15, BS18]). *If there exists a (q, δ) -selective-message message private SKFE scheme where q is any fixed polynomial, there exists a (q, δ) -adaptive-message function private SKFE scheme.*

Variants of Security We can consider an weaker security notion called selective-message-function function privacy.

Definition 3.16 (Selective-message-function Function Privacy). *The selective-message-function function privacy game is the same as the selective-message function privacy game except that \mathcal{A} must submit not only messages $(m_0^1, m_1^1), \dots, (m_0^p, m_1^p)$ but also functions $(f_0^1, f_1^1), \dots, (f_0^q, f_1^q)$ to the challenger at the beginning of the game. For an SKFE scheme SKFE and adversary \mathcal{A} , the modified advantage $\text{Adv}_{\text{SKFE}, \mathcal{A}}^{\text{smf-fp}}(\lambda)$ is similarly defined as $\text{Adv}_{\text{SKFE}, \mathcal{A}}^{\text{sm-fp}}(\lambda)$. Then, SKFE is said to be selective-message-function function private if $\text{Adv}_{\text{SKFE}, \mathcal{A}}^{\text{smf-fp}}(\lambda)$ is negligible for any PPT \mathcal{A} .*

Index Based Secret-Key Functional Encryption In this paper, we increase the number of functional keys an SKFE scheme supports through the index based variant SKFE scheme introduced by Li and Micciancio [LM16]. They showed how to increase the number of functional keys a public-key functional encryption scheme supports through the index based variant syntax. We introduce the syntax of index based variant SKFE here. We call it *index based secret-key functional encryption (iSKFE)*. Index based means that in order to generate the i -th functional key, we need to feed an index i to the key generation algorithm. The index based definition is required to use the strategy similar to that of Li and Micciancio. More precisely, we need an index since we use bounded collusion-resistant schemes as building blocks and need to control how many functional keys are generated under a master secret key.

In fact, for a single-key scheme, an iSKFE scheme is also an standard SKFE scheme where the key generation algorithm does not take an index as an input by assuming that the index is always fixed to 1. In addition, if an iSKFE scheme supports super-polynomially many number of functional keys, we can easily transform it into an ordinary SKFE scheme. See Remark 3.19 for more details.

Definition 3.17 (Index based secret-key functional encryption). *An iSKFE scheme iSKFE is a four tuple $(\text{Setup}, \text{iKG}, \text{Enc}, \text{Dec})$ of PPT algorithms. Below, let \mathcal{M} , \mathcal{F} , and \mathcal{I} be the message space, function space, and index space of iSKFE, respectively.*

- *The setup algorithm Setup, given a security parameter 1^λ , outputs a master secret key MSK.*
- *The index based key generation algorithm iKG, given a master secret key MSK, a function $f \in \mathcal{F}$, and an index $i \in \mathcal{I}$, 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 $\tilde{m} \in \{\perp\} \cup \mathcal{M}$.*

Correctness We require $\text{Dec}(\text{iKG}(\text{MSK}, f, i), \text{Enc}(\text{MSK}, m)) = f(m)$ for every $m \in \mathcal{M}$, $f \in \mathcal{F}$, $i \in \mathcal{I}$, and $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$.

Next, we introduce selective-message function privacy for iSKFE. Selective-message security is known to be sufficient for constructing IO from functional encryption. As we see in Theorem 3.14, we can obtain function privacy in general. Therefore, in this paper, we start the transformation from selective-message function private single-key SKFE, and use selective-message function privacy as a standard security notion for SKFE and iSKFE.

Definition 3.18 (Selective-message function privacy). *Let iSKFE be an iSKFE scheme whose message space, function space, and index space are \mathcal{M} , \mathcal{F} , and \mathcal{I} , respectively. We let $|\mathcal{I}| = q$. We define the selective-message function privacy game between a challenger and an adversary \mathcal{A} as follows.*

Initialization *First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \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 . Next, the challenger generates a master secret key $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Finally, the challenger generates ciphertexts $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, m_b^\ell)$ ($\ell \in [p]$) and sends them to \mathcal{A} .*

\mathcal{A} may adaptively make key queries q times at most.

Key queries *For a key query $(i, f_0, f_1) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$ from \mathcal{A} , the challenger generates $sk_f \leftarrow \text{KG}(\text{MSK}, f_b, i)$, and returns sk_f to \mathcal{A} . Here, f_0 and f_1 need to be the same size and satisfy $f_0(m_0^\ell) = f_1(m_1^\ell)$ for all $\ell \in [p]$. Moreover, \mathcal{A} is not allowed to make key queries for the same index i twice.*

Final phase *\mathcal{A} outputs $b' \in \{0, 1\}$.*

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

$$\text{Adv}_{\text{iSKFE}, \mathcal{A}}^{\text{sm-fp}}(\lambda) := 2 \left| \Pr[b = b'] - \frac{1}{2} \right| = \left| \Pr[b' = 1 \mid b = 0] - \Pr[b' = 1 \mid b = 1] \right|.$$

For a negligible function $\epsilon(\cdot)$, We say that iSKFE is (q, ϵ) -selective-message function private if for any PPT \mathcal{A} , we have $\text{Adv}_{\text{iSKFE}, \mathcal{A}}^{\text{sm-fp}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

Below, we say that an iSKFE scheme is (poly, ϵ) -selective-message function private if the size of its index space q is super-polynomial.

Remark 3.19 (Transforming iSKFE into SKFE). One of the goals in this paper is to construct (poly, ϵ) -selective-message function private SKFE based only on $(1, \epsilon')$ -selective-message function private SKFE for some negligible functions ϵ and ϵ' . To achieve this task, we first construct (poly, δ) -selective-message function private iSKFE for some negligible function δ .

Note that if the size of the index space is super-polynomial, we can transform it into SKFE without compromising the security. This is done by slightly changing the key generation algorithm so that it first picks a random index from the index space then generates a functional key using the randomly chosen index in every invocation. Let S be the size of the index space of iSKFE. Then, the probability that the same index is used in different invocations of the key generation algorithm is bounded by $O(\frac{q^2}{S})$. Therefore, the resulting SKFE is $(\text{poly}, \frac{1}{S} + \delta)$ -selective-message function private. In Section 10.4, we formally show that this transformation works.

Succinctness for Functional Encryption. Next, we define the succinctness for SKFE and iSKFE.

Definition 3.20 (Succinctness). *Let s and n be the maximum size and input length of functions contained in \mathcal{F} , respectively.*

Succinct: *We say that SKFE (or iSKFE) is succinct if the size of the encryption circuit is bounded by $\text{poly}(\lambda, n, \log s)$.*

Weakly succinct: We say that SKFE (or iSKFE) is weakly succinct if the size of the encryption circuit is bounded by $s^\gamma \cdot \text{poly}(\lambda, n)$, where $\gamma < 1$ is a fixed constant. We sometimes call γ the compression factor.

Collusion-succinct: We say that SKFE (or iSKFE) is collusion succinct if the size of the encryption circuit is bounded by $\text{poly}(n, \lambda, s, \log q)$, where q is the upper bound of the number of functional decryption keys that adversaries receive in bounded-key schemes.

Weakly collusion-succinct: We say that SKFE (or iSKFE) is weakly collusion-succinct if the size of the encryption circuit is bounded by $q^\gamma \cdot \text{poly}(n, \lambda, s)$, where q is the upper bound of the number of functional decryption keys that adversaries receive and $\gamma < 1$ is a fixed constant.

In this paper, we focus on iSKFE for P/poly. Below, unless stated otherwise, let the function space of iSKFE be P/poly.

3.3 Indistinguishability Obfuscation

We review the definition of indistinguishability obfuscation (IO) and its variant.

Definition 3.21 (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 poly-size distinguisher \mathcal{D} , there exists a negligible function $\text{negl}(\cdot)$ 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

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

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 $\epsilon(\lambda)^{\Omega(1)}$.

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

Definition 3.22 (Strong exponentially-efficient indistinguishability obfuscation). Let $\gamma < 1$ be a constant. A PPT algorithm $\text{sxi}\mathcal{O}$ 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 3.21 and the following efficiency requirement:

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

We have the following theorem.

Theorem 3.23 ([BNPW20]). Assuming there exists (poly, ϵ) -selective-message function private (i.e., 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.

4 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 [BV15]. 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 a weakly succinct one by using SXIO.

We can construct SXIO based on standard (i.e., not puncturable) SKFE by Theorem 3.23. Thus, we can construct our puncturable SKFE from standard SKFE.

4.1 Syntax

Our definition of puncturable SKFE introduced below is slightly different from that proposed by Bitansky and Vaikuntanathan [BV15]. 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.

In the journal version, Bitansky and Vaikuntanathan [BV18] also proposed a notion of puncturable functional encryption, which can be seen as an abstraction of the PKFE scheme used in their construction of IO. We also briefly explain the notion of puncturable functional encryption after introducing our definition.

Definition 4.1 (Puncturable secret-key functional encryption). *A puncturable SKFE scheme pSKFE is a tuple (Setup, KG, Enc, Dec, Punc, 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 functional keys given to adversaries.*

- *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 $\tilde{m} \in \{\perp\} \cup \mathcal{M}$.*
- *The puncturing algorithm Punc, given a master secret key MSK and a tag tag , outputs a punctured master secret key $MSK^*_{\{tag\}}$*
- *The punctured encryption algorithm PEnc, given a punctured master secret key $MSK^*_{\{tag\}}$, a tag tag' , and a message m , outputs a ciphertext CT.*

Correctness: *For every $m \in \mathcal{M}$, $f \in \mathcal{F}$, $i \in [q]$, $tag \in \mathcal{T}$, and $MSK \leftarrow \text{Setup}(1^\lambda)$, we require that*

$$\text{Dec}(\text{KG}(\text{MSK}, f, i), \text{tag}, \text{Enc}(\text{MSK}, \text{tag}, m)) = f(m) .$$

4.2 Security

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

Definition 4.2 (Secure puncturable SKFE). Let $\text{pSKFE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec}, \text{Punc}, \text{PEnc})$ 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 functional keys given to adversaries. We say that pSKFE is secure puncturable SKFE if it satisfies the following properties.

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

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

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 $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses a challenge bit $b \leftarrow \{0, 1\}$. The challenger sends security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\text{tag} \in \mathcal{T}$, 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 $\text{CT} \leftarrow \text{Enc}(\text{MSK}, \text{tag}, m_b)$, $\text{sk}_{f_i} \leftarrow \text{KG}(\text{MSK}, f_i, i)$ for every $i \in [q]$, and $\text{MSK}^*\{\text{tag}\} \leftarrow \text{Punc}(\text{MSK}, \text{tag})$. Then, the challenger returns $(\text{MSK}^*\{\text{tag}\}, \text{CT}, \{\text{sk}_{f_i}\}_{i \in [q]})$ to \mathcal{A} .
4. \mathcal{A} outputs $b' \in \{0, 1\}$.

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

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

\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 $\text{Adv}_{\text{pSKFE}, \mathcal{A}}^{\text{ss}}(\lambda) = \text{negl}(\lambda)$.

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 $\text{Adv}_{\text{pSKFE}, \mathcal{A}}^{\text{ss}}(\lambda)$ 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 4.3 (Indistinguishability of functionality under puncturing). Let $\text{pSKFE} = (\text{Setup}, \text{KG}, \text{Enc}, \text{Dec}, \text{Punc}, \text{PEnc})$ 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 $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses a challenge bit $b \leftarrow \{0, 1\}$. The challenger sends security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends $m \in \mathcal{M}$ and $(\text{tag}, \text{tag}') \in \mathcal{T} \times \mathcal{T}$ such that $\text{tag} \neq \text{tag}'$ to the challenger.
3. The challenger first computes $\text{MSK}^*\{\text{tag}\} \leftarrow \text{Punc}(\text{MSK}, \text{tag})$. Then, the challenger computes $\text{CT} \leftarrow \text{Enc}(\text{MSK}, \text{tag}', m)$ if $b = 0$, and otherwise $\text{CT} \leftarrow \text{PEnc}(\text{MSK}^*\{\text{tag}\}, \text{tag}', m)$.
Then, the challenger returns $(\text{MSK}, \text{MSK}^*\{\text{tag}\}, \text{CT})$ to \mathcal{A} .
4. \mathcal{A} outputs $b' \in \{0, 1\}$.

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

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

We say that pSKFE satisfies indistinguishability of functionality under puncturing if for any PPT \mathcal{A} , there exists $\text{negl}(\cdot)$ such that for every $\lambda \in \mathbb{N}$, we have $\text{Adv}_{\text{pSKFE}, \mathcal{A}}^{\text{if}}(\lambda) = \text{negl}(\lambda)$.

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 $\text{Adv}_{\text{pSKFE}, \mathcal{A}}^{\text{if}}(\lambda)$ is smaller than $\epsilon(\lambda)^{\Omega(1)}$.

Definition 4.4 (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.

Efficiency. We can consider the same notions of succinctness for puncturable SKFE as those for SKFE Definition 3.20.

Definition 4.5. We say that puncturable SKFE is succinct (resp., weakly succinct, collusion-succinct, or weakly collusion-succinct) if the size of the encryption circuit and puncturable encryption circuit is succinct (resp., weakly succinct, collusion-succinct, or weakly collusion-succinct) in the sense of Definition 3.20.

4.3 Difference from Definition of Bitansky and Vaikuntanathan

There are three main differences between our definition of puncturable SKFE and that of Bitansky and Vaikuntanathan [BV15]. 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 a 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 5 for details.

In our construction of IO in Section 7, 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 a weakly succinct one *via a collusion-succinct scheme whose key generation algorithm is stateful* in Section 6.

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 4.3. 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 4.2 but the relaxed one. Our *non-succinct* scheme satisfies functionality preserving under puncturing.

We show that indistinguishability of functionality under puncturing suffices for constructing IO and our weakly succinct scheme satisfies the property.

Puncturable functional encryption [BV18] In the journal version, Bitansky and Vaikuntanathan also proposed a notion of puncturable functional encryption, and described their construction of IO using puncturable functional encryption.

The definition of puncturable functional encryption is significantly different from that of puncturable SKFE. Puncturable functional encryption is designed so that it also captures the mechanism based on the function-hiding technique by Brakerski and Segev [BS18] used inside their IO construction. On the other hand, the definition of puncturable SKFE does not capture the function-hiding mechanism. Our IO construction in Section 7 directly implements the function-hiding mechanism on top of puncturable SKFE similarly to the IO construction based on PKFE by BV15 [BV15].

5 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 [SS10]. 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 $GC = (\text{Grbl}, \text{Eval})$ be garbling scheme, and $\text{PPRF} = (F, \text{Punc}_F)$ be puncturable PRF. Using GC and PPRF , we construct puncturable SKFE $\text{OneKey} = (1\text{Key.Setup}, 1\text{Key.KG}, 1\text{Key.Enc}, 1\text{Key.Dec}, 1\text{Key.Punc}, 1\text{Key.PEnc})$ 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 $(f[1], \dots, f[s])$.

Construction The scheme consists of the following algorithms.

$1\text{Key.Setup}(1^\lambda)$:

- Generate $S_{j,\alpha} \xleftarrow{r} \{0, 1\}^\lambda$ for every $j \in [s]$ and $\alpha \in \{0, 1\}$.
- Return $\text{MSK} := \{S_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}$.

1Key.KG(MSK, f) :

- Parse $\{S_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}} \leftarrow \text{MSK}$ and $(f[1], \dots, f[s]) \leftarrow f$.
- Return $\text{sk}_f := (f, \{S_{j,f[j]}\}_{j \in [s]})$.

1Key.Enc(MSK, tag, m) :

- Parse $\{S_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}} \leftarrow \text{MSK}$.
- Compute $(\tilde{U}, \{L_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, U(\cdot, m))$.
- For every $j \in [s]$ and $\alpha \in \{0, 1\}$, compute $R_{j,\alpha} \leftarrow F_{S_{j,\alpha}}(\text{tag})$ and $c_{j,\alpha} := L_{j,\alpha} \oplus R_{j,\alpha}$.
- Return $\text{CT} := (\tilde{U}, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$.

1Key.Dec(sk_f , tag, CT) :

- Parse $(f, \{S_j\}_{j \in [s]}) \leftarrow \text{sk}_f$ and $(\tilde{U}, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}) \leftarrow \text{CT}$.
- For every $j \in [s]$, compute $R_j \leftarrow F_{S_j}(\text{tag})$ and $L_j := c_{j,f[j]} \oplus R_j$.
- Return $y \leftarrow \text{Eval}(\tilde{U}, \{L_j\}_{j \in [s]})$.

1Key.Punc(MSK, tag) :

- Parse $\{S_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}} \leftarrow \text{MSK}$.
- For every $j \in [s]$ and $\alpha \in \{0, 1\}$, compute $S_{j,\alpha}^* \{\text{tag}\} \leftarrow \text{Punc}_F(S_{j,\alpha}, \text{tag})$.
- Return $\text{MSK}^* \{\text{tag}\} := \{S_{j,\alpha}^* \{\text{tag}\}\}_{j \in [s], \alpha \in \{0,1\}}$.

1Key.PEnc(MSK*, tag', m)

- Parse $\{S_{j,\alpha}^*\}_{j \in [s], \alpha \in \{0,1\}} \leftarrow \text{MSK}^*$.
- Compute $(\tilde{U}, \{L_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, U(\cdot, m))$.
- For every $j \in [s]$ and $\alpha \in \{0, 1\}$, compute $R_{j,\alpha} \leftarrow F_{S_{j,\alpha}^*}(\text{tag}')$ and $c_{j,\alpha} := L_{j,\alpha} \oplus R_{j,\alpha}$.
- Return $\text{CT} := (\tilde{U}, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$.

Then, we have the following theorem.

Theorem 5.1. *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.*

Proof of Theorem 5.1 The correctness follows from those of GC and PPRF. We first prove the functionality preserving under puncturing of OneKey. Then, we show that OneKey satisfies semantic security at punctured tag.

Functionality preserving under puncturing We have $1\text{Key.PEnc}(\text{MSK}^* \{\text{tag}\}, \text{tag}', m; r) = 1\text{Key.Enc}(\text{MSK}, \text{tag}', m; r)$ for every $m \in \mathcal{M}$, $(\text{tag}, \text{tag}') \in \mathcal{T} \times \mathcal{T}$ such that $\text{tag} \neq \text{tag}'$, randomness r , $\text{MSK} \leftarrow 1\text{Key.Setup}(1^\lambda)$, and $\text{MSK}^* \{\text{tag}\} \leftarrow 1\text{Key.Punc}(\text{MSK}, \text{tag})$, since the underlying PPRF satisfies functionality preserving under puncturing property. This implies that OneKey satisfies functionality preserving under puncturing property.

Semantic security at punctured tag Let \mathcal{A} be a valid adversary that attacks the semantic security at punctured tag of OneKey. We proceed the proof via a sequence of games. Below, for every $\ell \in \{0, 1, 2, 3\}$, let SUC_ℓ be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game ℓ .

Game 0: This is the original security game regarding OneKey. Then, we have $\text{Adv}_{\text{pSKFE}, \mathcal{A}}^{\text{ss}}(\lambda) = 2|\Pr[\text{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger generates $S_{j,\alpha} \xleftarrow{r} \{0,1\}^\lambda$ for every $j \in [s]$ and $\alpha \in \{0,1\}$, and sets $\text{MSK} \leftarrow \{S_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}$. The challenger also chooses a challenge bit $b \xleftarrow{r} \{0,1\}$. The challenger sends security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\text{tag} \in \mathcal{T}$, and a function f to the challenger.
3. The challenger computes $(\tilde{U}, \{L_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, U(\cdot, m_b))$ and $R_{j,\alpha} \leftarrow \text{FS}_{j,\alpha}(\text{tag})$ and $c_{j,\alpha} := L_{j,\alpha} \oplus R_{j,\alpha}$ for every $j \in [s]$ and $\alpha \in \{0,1\}$, and sets $\text{CT} := (\tilde{U}, \{c_{j,\alpha}\}_{j \in [s], \alpha \in \{0,1\}})$. Next, the challenger sets $\text{sk}_f \leftarrow (f, \{S_{j,f[j]}\}_{j \in [s]})$. Then, the challenger computes $S_{j,\alpha}^* \{\text{tag}\} \leftarrow \text{Punc}_F(S_{j,\alpha}, \text{tag})$ for every $j \in [s]$ and $\alpha \in \{0,1\}$, and sets $\text{MSK}^* \{\text{tag}\} \leftarrow \{S_{j,\alpha}^* \{\text{tag}\}\}_{j \in [s], \alpha \in \{0,1\}}$. The challenger returns $(\text{MSK}^* \{\text{tag}\}, \text{CT}, \text{sk}_f)$ to \mathcal{A} .
4. \mathcal{A} outputs $b' \in \{0,1\}$.

Game 1: Same as Game 1 except that the challenger generates $\{R_{j,1-f[j]}\}_{j \in [n]}$ as truly random strings.

From the pseudorandomness of punctured point of PPRF, we see that $|\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| \leq \delta^{\Omega(1)}$.

Game 2: Same as Game 2 except that for every $j \in [n]$, the challenger generates $c_{j,1-f[j]} \leftarrow R_{j,f[j]}$.

In Game 1, $c_{j,1-f[j]}$ is generated as $c_{j,1-f[j]} := L_{j,1-f[j]} \oplus R_{j,1-f[j]}$ for every $j \in [n]$. However, in Game 1, $R_{j,1-f[j]}$ is generated as a truly random string for every $j \in [n]$, and thus the distribution of $c_{j,1-f[j]}$ is uniformly random. Therefore, In Game 1 and 2, the distribution of $c_{j,1-f[j]}$ for every $j \in [n]$ is the same and we have $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| = 0$.

Game 3: Same as Game 2 except that the challenger computes $(\tilde{U}, \{L_j\}_{j \in [n]}) \leftarrow \text{Sim}(1^\lambda, |U(\cdot, m_b)|, y)$ and $c_{j,f[j]} := R_{j,f[j]} \oplus L_j$, where $y = f(m_0) = f(m_1)$.

In both Game 2 and 3, \mathcal{A} is not given any information of labels $\{L_{j,1-f[j]}\}_{j \in [n]}$. Therefore, we can use the security guarantee of GC, and obtain $|\Pr[\text{SUC}_2] - \Pr[\text{SUC}_3]| \leq \delta^{\Omega(1)}$.

In Game 3, the choice of the challenge bit b is information theoretically hidden from the view of \mathcal{A} , and thus we have $|\Pr[\text{SUC}_3] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned} \frac{1}{2} \text{Adv}_{\text{OneKey}, \mathcal{A}}^{\text{ss}}(\lambda) &= |\Pr[\text{SUC}_0] - \frac{1}{2}| \\ &\leq |\Pr[\text{SUC}_0] - \Pr[\text{SUC}_3]| \\ &\leq \sum_{\ell=0}^2 |\Pr[\text{SUC}_\ell] - \Pr[\text{SUC}_{\ell+1}]|. \end{aligned} \quad (1)$$

From the above argument, each term of the right side of inequality 1 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\text{Adv}_{\text{OneKey}, \mathcal{A}}^{\text{ss}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of \mathcal{A} is arbitrary, OneKey satisfies δ -semantic security at punctured tag. \square (**Theorem 5.1**)

6 From Non-Succinct to Weakly Succinct Puncturable SKFE

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. [LPST16] (and extended by Bitansky et al. [BNPW20] and Kitagawa et al. [KNT21]), 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.

Below, we start with the first step.

6.1 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 $\text{OneKey} = (1\text{Key.Setup}, 1\text{Key.KG}, 1\text{Key.Enc}, 1\text{Key.Dec}, 1\text{Key.Punc}, 1\text{Key.PEnc})$ be puncturable SKFE that we constructed in Section 5. Let sxiO be SXIO and $\text{PPRF} = (F, \text{Punc}_F)$ puncturable PRF. Using OneKey , sxiO , and PPRF , we construct puncturable SKFE $\text{CollSuc} = (\text{CS.Setup}, \text{CS.KG}, \text{CS.Enc}, \text{CS.Dec}, \text{CS.Punc}, \text{CS.PEnc})$ as follows. We again note that q is a fixed polynomial of λ . Let the tag space of CollSuc be \mathcal{T} . Then, the tag space of OneKey is also \mathcal{T} .

Construction The scheme consists of the following algorithms.

$\text{CS.Setup}(1^\lambda)$:

- Generate $S \xleftarrow{r} \{0, 1\}^\lambda$ and return $\text{MSK} := S$.

$\text{CS.KG}(\text{MSK}, f, i)$:

- Parse $S \leftarrow \text{MSK}$.
- Compute $r_{\text{Setup}}^i \leftarrow F_S(i)$ and $\text{MSK}_i \leftarrow 1\text{Key.Setup}(1^\lambda; r_{\text{Setup}}^i)$.
- Compute $1\text{Key.sk}_f \leftarrow 1\text{Key.KG}(\text{MSK}_i, f)$ and return $\text{sk}_f := (i, 1\text{Key.sk}_f)$.

$\text{CS.Enc}(\text{MSK}, \text{tag}, m)$:

- Parse $S \leftarrow \text{MSK}$.
- Generate $S_{\text{Enc}} \xleftarrow{r} \{0, 1\}^\lambda$ and return $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}[S, S_{\text{Enc}}, \text{tag}, m])$. The circuit $E_{1\text{Key}}$ is defined in Figure 7.

$\text{CS.Dec}(\text{sk}_f, \text{tag}, \text{CT})$:

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

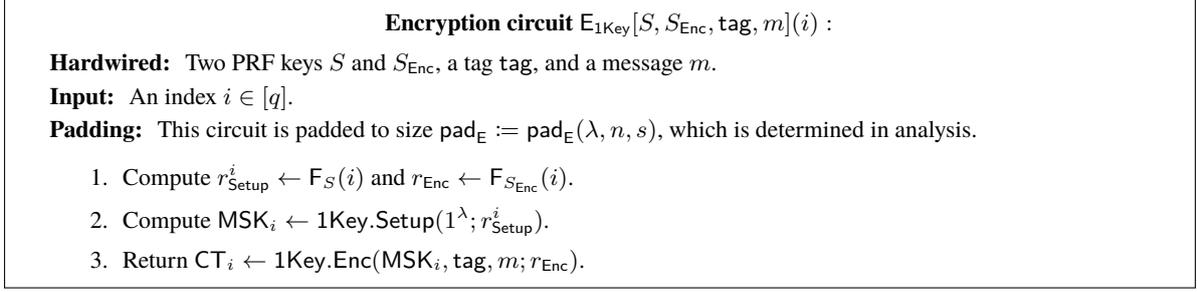


Figure 7: The description of $E_{1\text{Key}}$.

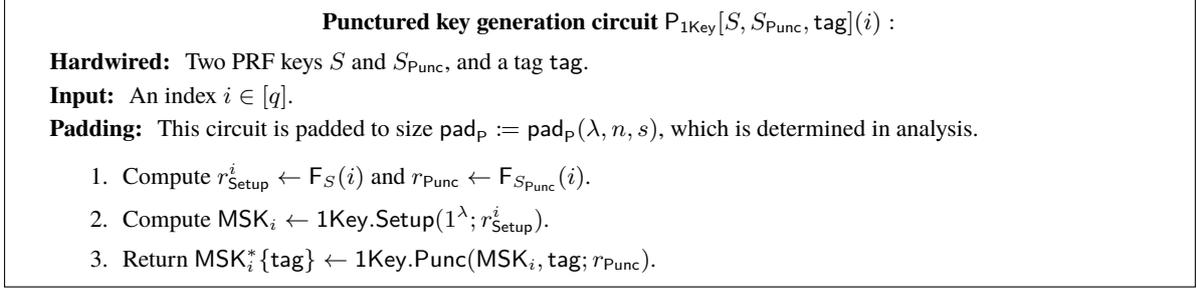


Figure 8: The description of $P_{1\text{Key}}$.

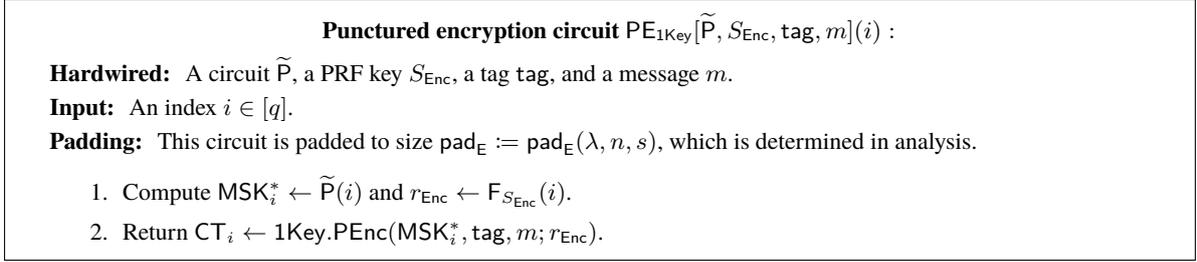


Figure 9: The description of $PE_{1\text{Key}}$.

$\text{CS.Punc}(\text{MSK}, \text{tag}) :$

- Parse $S \leftarrow \text{MSK}$.
- Generate $S_{\text{Punc}} \xleftarrow{r} \{0, 1\}^\lambda$ and compute $\tilde{P} \leftarrow \text{sxiO}(P_{1\text{Key}}[S, S_{\text{Punc}}, \text{tag}])$. The circuit $P_{1\text{Key}}$ is defined in Figure 8.
- Return $\text{MSK}^* \{\text{tag}\} := \tilde{P}$.

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

- Parse $\tilde{P} \leftarrow \text{MSK}^*$.
- Generate $S_{\text{Enc}} \xleftarrow{r} \{0, 1\}^\lambda$ and return $\text{CT} \leftarrow \text{sxiO}(PE_{1\text{Key}}[\tilde{P}, S_{\text{Enc}}, \text{tag}', m])$. The circuit $PE_{1\text{Key}}$ is defined in Figure 9.

Then, we have the following theorem.

Theorem 6.1. *Let $\delta(\cdot)$ be some negligible function. Let OneKey be δ -secure single-key puncturable SKFE constructed in Section 5. Let sxiO be δ -secure γ -compressing SXIO, where γ is a sufficiently small constant*

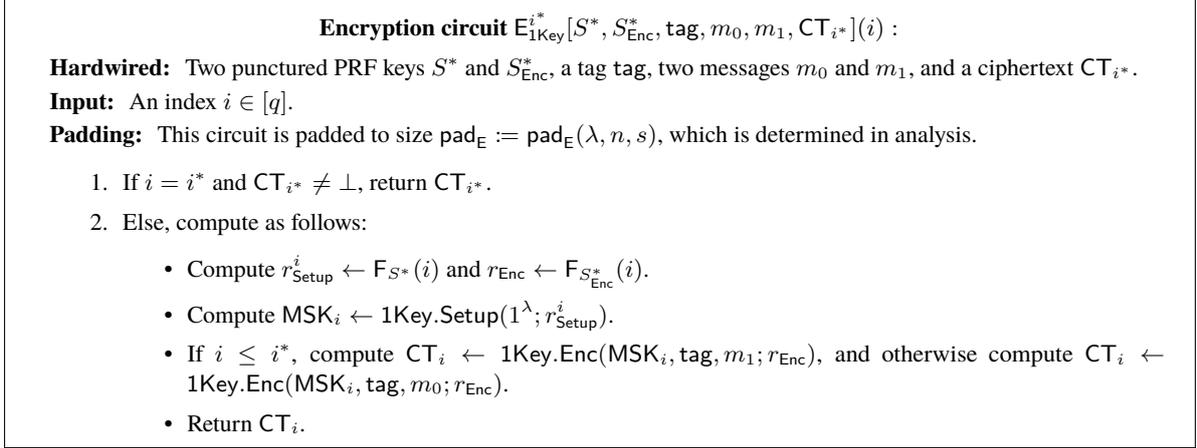


Figure 10: The description of $E_{1\text{Key}}^{i^*}$. The circuit is defined for every $i^* \in [q]$.

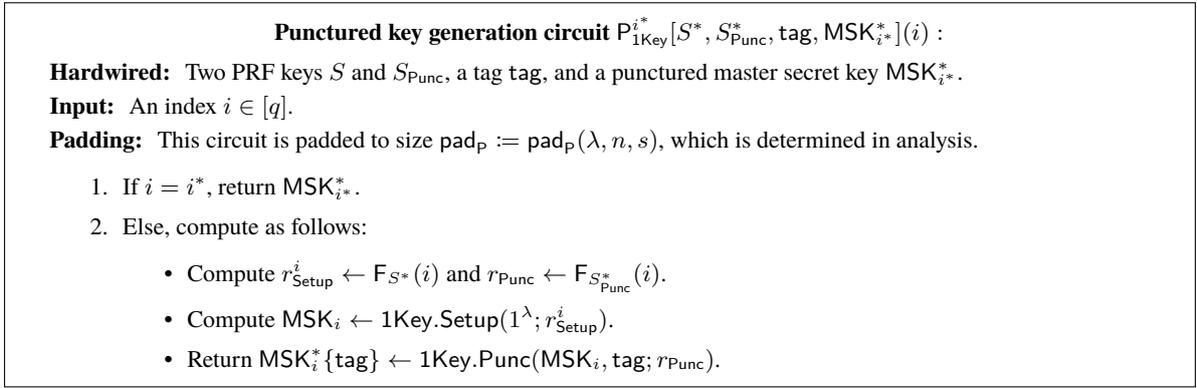


Figure 11: The description of $P_{1\text{Key}}^{i^*}$. The circuit is defined for every $i^* \in [q]$.

such that $\gamma < 1$. Let PPRF be δ -secure puncturable PRF. Then, CollSuc is (δ, δ) -secure puncturable SKFE with indistinguishability of functionality that is weakly collusion-succinct with compression factor $\hat{\gamma}$, which is a constant smaller than 1.

The concrete value of $\hat{\gamma}$ is determined in the efficiency analysis in the following proof of Theorem 6.1. As we will see, we can make $\hat{\gamma}$ smaller than 1 by using SXIO with sufficiently small compression factor as a the building block. Such SXIO is constructed from collusion-resistant SKFE [BNPW20].

Proof of Theorem 6.1 We first determine the size of padding parameters. Then, we analyze the efficiency. Finally, we complete the security proof.

Below, let s and n be the upper bound of size and input length of functions supported by CollSuc.

Padding Parameter In order to complete this proof, we ensure that the encryption circuits $E_{1\text{Key}}$, $PE_{1\text{Key}}$, and $E_{1\text{Key}}^{i^*}$ for every $i^* \in [q]$ are indistinguishable when we obfuscate them by SXIO. Moreover, the obfuscated $P_{1\text{Key}}$ and $P_{1\text{Key}}^{i^*}$ also need to be indistinguishable. For this reason, we need appropriate size padding for these circuits. Below, we first analyze the size of padding for $P_{1\text{Key}}$ and $P_{1\text{Key}}^{i^*}$ because the description of $PE_{1\text{Key}}$ includes obfuscated $P_{1\text{Key}}$.

To guarantee the indistinguishability of $P_{1\text{Key}}$ and $P_{1\text{Key}}^{i^*}$ when we obfuscate them, we need to set

$$\text{pad}_P := \max(|P_{1\text{Key}}|, |P_{1\text{Key}}^{i^*}|) .$$

Both $P_{1\text{Key}}$ and $P_{1\text{Key}}^{i^*}$ includes two PRF evaluation over the domain $[q]$, and the key generation and puncturing procedure of OneKey. Since OneKey is a single-key scheme, and q is determined independently of OneKey, the running time of each algorithm of OneKey is independent of q . Therefore, we have

$$\begin{aligned} \text{pad}_P &\leq \text{poly}(\lambda, \log q) + \text{poly}(\lambda, n, s) \\ &\leq \text{poly}_P(\lambda, n, s, \log q) , \end{aligned}$$

where poly_P is some fixed polynomial.

Then, we move on to the analysis of the padding parameter for encryption algorithms.

We need to set pad_E as

$$\text{pad}_E := \max(|E_{1\text{Key}}|, |PE_{1\text{Key}}|, |E_{1\text{Key}}^{i^*}|) .$$

$E_{1\text{Key}}$ and $E_{1\text{Key}}^{i^*}$ for every $i^* \in [q]$ consists of two PRF evaluation over the domain $[q]$, and the key generation and encryption procedure of OneKey. Therefore, we have

$$\max(|E_{1\text{Key}}|, |E_{1\text{Key}}^{i^*}|) \leq \text{poly}(\lambda, n, s, \log q) . \quad (2)$$

In addition, $PE_{1\text{Key}}$ includes one PRF evaluation over the domain $[q]$, the execution of \tilde{P} that is obfuscated P by $\text{sxi}\mathcal{O}$, and the punctured encryption procedure of OneKey. Then, since the non-trivial efficiency of $\text{sxi}\mathcal{O}$, when we obfuscate a circuit C with input space $[N]$ by $\text{sxi}\mathcal{O}$, we can bound the size of obfuscated C by $N^\gamma \cdot |C|^c \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda)$, where $\gamma < 1$ and c are constants. Thus, we have

$$\begin{aligned} |\tilde{P}| &\leq q^\gamma \cdot |P|^c \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda) \\ &= q^\gamma \cdot |\text{poly}_P(\lambda, n, s, \log q)|^c \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda) \\ &\leq q^{\gamma_1} \cdot \text{poly}(\lambda, n, s) , \end{aligned}$$

where γ_1 is an arbitrary constant such that $\gamma < \gamma_1 < 1$. Hence, we obtain

$$\begin{aligned} |PE_{1\text{Key}}| &\leq \text{poly}(\lambda, \log q) + q^{\gamma_1} \cdot \text{poly}(\lambda, n, s) + \text{poly}(\lambda, n, s) \\ &\leq q^{\gamma_1} \cdot \text{poly}(\lambda, n, s) . \end{aligned} \quad (3)$$

Therefore, from inequalities 2 and 3, we have

$$\text{pad}_E \leq q^{\gamma_1} \cdot \text{poly}_E(\lambda, n, s) , \quad (4)$$

where poly_E is some fixed polynomial.

Efficiency To simplify the efficiency analysis, we assume that we use two different SXIO $\text{sxi}\mathcal{O}$ and $\text{sxi}\mathcal{O}'$. We use $\text{sxi}\mathcal{O}$ to obfuscate $P_{1\text{Key}}$. We use $\text{sxi}\mathcal{O}'$ to obfuscate $E_{1\text{Key}}$ and $PE_{1\text{Key}}$.

We assume that when we obfuscate a circuit C with input space $[N]$ by $\text{sxi}\mathcal{O}$ and $\text{sxi}\mathcal{O}'$, we can bound the size of $\text{sxi}\mathcal{O}(C)$ and $\text{sxi}\mathcal{O}'(C)$ by

$$N^\gamma \cdot |C|^c \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda) \quad \text{and} \quad N^{\gamma'} \cdot |C|^{c'} \cdot \text{poly}_{\text{sxi}\mathcal{O}'}(\lambda) ,$$

respectively, where γ and γ' are constants strictly smaller than 1, and c and c' are constants.

Then, from inequality 4, we can bound the running time of both CS.Enc and CS.PEnc by

$$\begin{aligned} q^{\gamma'} \cdot (\text{pad}_E)^{c'} \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda) &\leq q^{\gamma'} \cdot (q^{\gamma_1} \cdot \text{poly}_E(\lambda, n, s))^{c'} \cdot \text{poly}_{\text{sxi}\mathcal{O}}(\lambda) \\ &\leq q^{\gamma' + c'\gamma_1} \cdot \text{poly}(\lambda, n, s) , \end{aligned}$$

where γ_1 is an arbitrary constant such that $\gamma < \gamma_1 < 1$.

Therefore, if we have $\gamma' + c'\gamma_1 < 1$, we can conclude that CollSuc is weakly collusion-succinct. From Theorem 3.23, using a collusion-resistant SKFE scheme, we can construct SXIO with arbitrary constant compression factor. Thus, we can use SXIO with compression factor smaller than $\frac{1-\gamma'}{c'}$ as $\text{sxi}\mathcal{O}$, and ensure that $\hat{\gamma} := \gamma' + c'\gamma_1 < 1$ in our construction by assuming a collusion-resistant SKFE scheme.

This completes the efficiency analysis.

Indistinguishability of functionality under puncturing A ciphertext output by the standard encryption algorithm is an obfuscated circuit of $E_{1\text{Key}}$. A ciphertext output by the punctured encryption algorithm is an obfuscated circuit of $PE_{1\text{Key}}$. Thus, if we prove that $E_{1\text{Key}}$ and $PE_{1\text{Key}}$ are functionally equivalent, δ -indistinguishability of functionality under puncturing of CollSuc holds due to the δ -security of $\text{sxi}\mathcal{O}$.

Note that \tilde{P} in $PE_{1\text{Key}}$ has the exactly same functionality as $P_{1\text{Key}}$ due to the functionality preserving property of $\text{sxi}\mathcal{O}$. Thus, on input $i \in [q]$, $E_{1\text{Key}}$ and $PE_{1\text{Key}}$ basically compute the followings:

1. Compute $r_{\text{Setup}}^i \leftarrow F_S(i)$ and $r_{\text{Enc}} \leftarrow F_{S_{\text{Enc}}}(i)$.
2. Compute $\text{MSK}_i \leftarrow 1\text{Key.Setup}(1^\lambda; r_{\text{Setup}}^i)$.
3. $E_{1\text{Key}}$ and $PE_{1\text{Key}}$ respectively computes CT_i as follows:
 - $E_{1\text{Key}}$ computes $\text{CT}_i \leftarrow 1\text{Key.Enc}(\text{MSK}_i, \text{tag}', m; r_{\text{Enc}})$
 - $PE_{1\text{Key}}$ computes $\text{CT}_i \leftarrow 1\text{Key.PEnc}(\text{MSK}_i^* \{\text{tag}\}, \text{tag}', m; r_{\text{Enc}})$ by using $\text{MSK}_i^* \{\text{tag}\} \leftarrow 1\text{Key.Punc}(\text{MSK}_i, \text{tag}; r_{\text{Punc}})$ and $r_{\text{Punc}} \leftarrow F_{S_{\text{Punc}}}(i)$.
4. Return CT_i .

Recall that OneKey satisfies functionality preserving under puncturing property defined in Definition 4.2. Thus, both $E_{1\text{Key}}$ and $PE_{1\text{Key}}$ compute the same CT_i as long as $\text{tag}' \neq \text{tag}$ holds and the same S_{Enc} is used in both circuits.

Thus, we can conclude that CollSuc satisfies δ -indistinguishability of functionality under puncturing by the δ -security of $\text{sxi}\mathcal{O}$.

Semantic security at punctured tag Let \mathcal{A} be a valid adversary that attacks the semantic security at punctured tag of CollSuc. We proof it via a sequence of games. Let SUC_j denote the event that \mathcal{A} succeeds in guessing the challenge bit b in Game j .

Game 0 This is the punctured semantic security game regarding CollSuc. Then, we have $\text{Adv}_{\text{CollSuc}, \mathcal{A}}^{\text{SS}}(\lambda) = 2|\Pr[\text{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger generates $S \xleftarrow{r} \{0, 1\}^\lambda$ and sets $\text{MSK} := S$. The challenger also chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. The challenger sends security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\text{tag} \in \mathcal{T}$, and $\{f_i\}_{i \in [q]}$ to the challenger.

3. The challenger generates $S_{\text{Enc}} \xleftarrow{r} \{0, 1\}^\lambda$ and computes $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}[S, S_{\text{Enc}}, \text{tag}, m])$.
Next, for every $i \in [q]$, the challenger computes as follows. The challenger computes $r_{\text{Setup}}^i \leftarrow F_S(i)$, $\text{MSK}_i \leftarrow 1\text{Key.Setup}(1^\lambda; r_{\text{Setup}}^i)$, and $1\text{Key.sk}_{f_i} \leftarrow 1\text{Key.KG}(\text{MSK}_i, f_i)$.
The challenger sets $\text{sk}_{f_i} \leftarrow (i, 1\text{Key.sk}_{f_i})$.
Then, the challenger generates $S_{\text{Punc}} \xleftarrow{r} \{0, 1\}^\lambda$ and computes $\tilde{\text{P}} \leftarrow \text{sxiO}(P_{1\text{Key}}[S, S_{\text{Punc}}, \text{tag}])$.
Then, the challenger sets $\text{MSK}^*\{\text{tag}\} := \tilde{\text{P}}$.
The challenger returns $(\text{MSK}^*\{\text{tag}\}, \text{CT}, \{\text{sk}_{f_i}\}_{i \in [q]})$ to \mathcal{A} .
4. \mathcal{A} outputs $b' \in \{0, 1\}$.

Then, for every $i^* \in [q]$, we define the following games. Let $\text{SUC}_{(\ell, i^*)}$ denote the event that \mathcal{A} succeeds in guessing the challenge bit b in Game (ℓ, i^*) for every $\ell \in \{1, \dots, 6\}$ and $i^* \in [q]$.

Game $(6, 0)$ We define Game $(6, 0)$ as the same game as Game 0.

Game $(1, i^*)$ Same as Game $(6, i^* - 1)$ except the followings. The challenger generates $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}^{i^*}[S^*\{i^*\}, S_{\text{Enc}}^*\{i^*\}, \text{tag}, m_b, m_1, \text{CT}_{i^*}])$, where $\text{CT}_{i^*} \leftarrow 1\text{Key.Enc}(\text{MSK}_{i^*}, \text{tag}, m_b; r_{\text{Enc}}^{i^*})$, $\text{MSK}_{i^*} \leftarrow 1\text{Key.Setup}(1^\lambda; r_{\text{Setup}}^{i^*})$, $r_{\text{Setup}}^{i^*} \leftarrow F_S(i^*)$, and $r_{\text{Enc}}^{i^*} \leftarrow F_{S_{\text{Enc}}}(i^*)$.

The only difference between Game $(6, i^* - 1)$ and $(1, i^*)$ is how CT is generated. In Game $(6, i^* - 1)$, CT is generated by $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}^{i^*-1}[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp])$. However, we see that $E_{1\text{Key}}^{i^*-1}[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp]$ and $E_{1\text{Key}}^{i^*}[S^*\{i^*\}, S_{\text{Enc}}^*\{i^*\}, \text{tag}, m_b, m_1, \text{CT}_{i^*}]$ have exactly the same functionality. Therefore, by the indistinguishability guarantee of sxiO , we have $|\Pr[\text{SUC}_{(6, i^*-1)}] - \Pr[\text{SUC}_{(1, i^*)}]| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

Game $(2, i^*)$ Same as Game $(1, i^*)$ except the followings. The challenger generates $\tilde{\text{P}} \leftarrow \text{sxiO}(P_{1\text{Key}}^{i^*}[S^*\{i^*\}, S_{\text{Punc}}^*\{i^*\}, \text{tag}, \text{MSK}_{i^*}^*\{\text{tag}\}])$, where $\text{MSK}_{i^*}^*\{\text{tag}\} \leftarrow 1\text{Key.Punc}(\text{MSK}_{i^*}, \text{tag}; r_{\text{Punc}}^{i^*})$.

Similarly to the analysis between Game $(6, i^* - 1)$ and $(1, i^*)$, due to the indistinguishability guarantee of sxiO and the fact that $P_{1\text{Key}}[S, S_{\text{Punc}}, \text{tag}]$ and $P_{1\text{Key}}^{i^*}[S^*\{i^*\}, S_{\text{Punc}}^*\{i^*\}, \text{tag}, \text{MSK}_{i^*}^*\{\text{tag}\}]$ have the same functionality, we have $|\Pr[\text{SUC}_{(1, i^*)}] - \Pr[\text{SUC}_{(2, i^*)}]| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

Game $(3, i^*)$ Same as Game $(2, i^*)$ except that the challenger generates $r_{\text{Setup}}^{i^*}$, $r_{\text{Enc}}^{i^*}$, and $r_{\text{Punc}}^{i^*}$ as truly random strings.

From the pseudorandomness of PPRF, $|\Pr[\text{SUC}_{(2, i^*)}] - \Pr[\text{SUC}_{(3, i^*)}]| \leq \delta^{\Omega(1)}$ holds for every $i^* \in [q]$.

Game $(4, i^*)$ Same as Game $(3, i^*)$ except that the challenger generates $\text{CT}_{i^*} \leftarrow 1\text{Key.Enc}(\text{MSK}_{i^*}, \text{tag}, m_1)$.

In both Game $(3, i^*)$ and $(4, i^*)$, all of MSK_{i^*} , $\text{MSK}_{i^*}^*\{\text{tag}\}$, and CT_{i^*} are generated under truly random strings. In addition, since \mathcal{A} is a valid adversary, it holds that $f_{i^*}(m_0) = f_{i^*}(m_1)$. Therefore, from the semantic security at punctured tag of OneKey, we obtain $|\Pr[\text{SUC}_{(3, i^*)}] - \Pr[\text{SUC}_{(4, i^*)}]| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

Game $(5, i^*)$ Same as Game $(4, i^*)$ except that the challenger generates $r_{\text{Setup}}^{i^*}$, $r_{\text{Enc}}^{i^*}$, and $r_{\text{Punc}}^{i^*}$ using PPRF.

From the pseudorandomness of PPRF, $|\Pr[\text{SUC}_{(4, i^*)}] - \Pr[\text{SUC}_{(5, i^*)}]| \leq \delta^{\Omega(1)}$ holds for every $i^* \in [q]$.

Game $(6, i^*)$ Same as Game $(5, i^*)$ except that the challenger generates $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}^{i^*}[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp])$ and $\tilde{\text{P}} \leftarrow \text{sxiO}(P_{1\text{Key}}[S, S_{\text{Punc}}, \text{tag}])$.

Similarly to the analysis between Game $(6, i^* - 1)$ and $(1, i^*)$, due to the indistinguishability guarantee of sxiO and the fact that $E_{1\text{Key}}^{i^*}[S^*\{i^*\}, S_{\text{Enc}}^*\{i^*\}, \text{tag}, m_b, m_1, \text{CT}_{i^*}]$ and $E_{1\text{Key}}^{i^*}[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp]$ have exactly the same functionality, we have $|\Pr[\text{SUC}_{(5, i^*)}] - \Pr[\text{SUC}_{(6, i^*)}]| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

We define one additional game.

Game 7 Same as Game $(6, q)$ except the followings. The challenger generates $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}[S, S_{\text{Enc}}, \text{tag}, m_1])$.

In Game $(6, q)$, CT is generated by $\text{CT} \leftarrow \text{sxiO}(E_{1\text{Key}}^q[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp])$. $E_{1\text{Key}}^q[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp]$ always ignores m_b and outputs a ciphertext of m_1 . Therefore, $E_{1\text{Key}}^q[S, S_{\text{Enc}}, \text{tag}, m_b, m_1, \perp]$ and $E_{1\text{Key}}[S, S_{\text{Enc}}, \text{tag}, m_1]$ have the same functionality. Therefore, from the indistinguishability guarantee of sxiO , we have $|\Pr[\text{SUC}_{(6, q)}] - \Pr[\text{SUC}_7]| \leq \delta^{\Omega(1)}$.

In Game 7, the choice of the challenge bit b is information theoretically hidden from the view of \mathcal{A} , and thus we have $|\Pr[\text{SUC}_7] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned}
& \frac{1}{2} \text{Adv}_{\text{CollSuc}, \mathcal{A}}^{\text{ss}}(\lambda) \\
&= |\Pr[\text{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\text{SUC}_0] - \Pr[\text{SUC}_7]| \\
&\leq |\Pr[\text{SUC}_0] - \Pr[\text{SUC}_{(1,1)}]| + \sum_{i^* \in [q]} \sum_{\ell=1}^5 |\Pr[\text{SUC}_{(\ell, i^*)}] - \Pr[\text{SUC}_{(\ell+1, i^*)}]| \\
&\quad + |\Pr[\text{SUC}_{(6, q)}] - \Pr[\text{SUC}_7]|. \tag{5}
\end{aligned}$$

From the above argument, each term of the right side of inequality 5 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\text{Adv}_{\text{CollSuc}, \mathcal{A}}^{\text{ss}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of \mathcal{A} is arbitrary, CollSuc satisfies δ -semantic security at punctured tag. \square (**Theorem 6.1**)

6.2 From Collusion-Succinct to Weakly Succinct

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

This transformation is based on those proposed by Bitansky and Vaikuntanathan [BV15] and Ananth et al. [AJS15], 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 $\text{WeakSuc} = (\text{WS.Setup}, \text{WS.KG}, \text{WS.Enc}, \text{WS.Dec}, \text{WS.Punc}, \text{WS.PEnc})$. Let s and n be the maximum size and input length of functions supported by WeakSuc . Let $\text{RE} = \text{RE}(\text{E}, \text{D})$ be decomposable randomized encoding. 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 $s \cdot \text{poly}_{\text{RE}}(\lambda, n)$, where $\text{poly}_{\text{RE}}(\lambda, n)$ is a fixed polynomial. We also suppose that the randomness space of RE is $\{0, 1\}^\rho$, where ρ is a polynomial bounded by $s \cdot \text{poly}_{\text{RE}}(\lambda, n)$. Let $\text{CollSuc} = (\text{CS.Setup}, \text{CS.KG}, \text{CS.Enc}, \text{CS.Dec}, \text{CS.Punc}, \text{CS.PEnc})$ be puncturable SKFE whose index space and tag space are $[\mu]$ and \mathcal{T} , respectively. Let $\text{SKE} = (\text{E}, \text{D})$ be SKE and PRF PRF. In the scheme,

Decomposable Randomized Encoding Circuit $\text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}](b, m, S_{\text{encd}}, K)$

Hardwired: A decomposed function \widehat{f}_i , a string t , and a ciphertext CT_i^{ske} .

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

1. If $b = 1$, return $e_i \leftarrow \text{D}(K, \text{CT}_i^{\text{ske}})$.
2. Else, compute $r \leftarrow \text{PRF}_{S_{\text{encd}}}(t)$.
3. Return $e_i \leftarrow \widehat{f}_i(m; r)$.

Figure 12: The description of En_{dre} .

we use $\text{PRF} : \{0, 1\}^\lambda \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\rho$. Using CollSuc , RE , SKE , and PRF , we construct WeakSuc as follows. The tag space of WeakSuc is \mathcal{T} .

$\text{WS.Setup}(1^\lambda)$:

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

$\text{WS.KG}(\text{MSK}, f)$:

- Generate $K \xleftarrow{r} \{0, 1\}^\lambda$ and $t \xleftarrow{r} \{0, 1\}^\lambda$.
- Compute $(\widehat{f}_1, \dots, \widehat{f}_\mu) \leftarrow \text{RE.E}(1^\lambda, f)$.
- Generate $\text{CT}_i^{\text{ske}} \leftarrow \text{E}(K, 0^{|\widehat{f}_i(\cdot)|})$, and compute $\text{sk}_{\text{En}_i} \leftarrow \text{CS.KG}(\text{MSK}, \text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}], i)$ for every $i \in [\mu]$. En_{dre} defined in Figure 12.
- Return $\text{sk}_f := (\text{sk}_{\text{En}_1}, \dots, \text{sk}_{\text{En}_\mu})$.

$\text{WS.Enc}(\text{MSK}, \text{tag}, m)$:

- Generate $S_{\text{encd}} \xleftarrow{r} \{0, 1\}^\lambda$.
- Return $\text{CT} \leftarrow \text{CS.Enc}(\text{MSK}, \text{tag}, (0, m, S_{\text{encd}}, \perp))$.

$\text{WS.Dec}(\text{sk}_f, \text{tag}, \text{CT})$:

- Parse $(\text{sk}_{\text{En}_1}, \dots, \text{sk}_{\text{En}_\mu}) \leftarrow \text{sk}_f$.
- For every $i \in [\mu]$, compute $e_i \leftarrow \text{CS.Dec}(\text{sk}_{\text{En}_i}, \text{tag}, \text{CT})$.
- Decode the encoding, that is, compute $y \leftarrow \text{RE.D}(e_1, \dots, e_\mu)$.
- Return y .

$\text{WS.Punc}(\text{MSK}, \text{tag})$:

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

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

- Generate $S_{\text{encd}} \xleftarrow{r} \{0, 1\}^\lambda$.
- Return $\text{CT} \leftarrow \text{CS.PEnc}(\text{MSK}^*, \text{tag}', (0, m, S_{\text{encd}}, \perp))$.

Theorem 6.2. *Let $\delta(\cdot)$ be negligible function. Let CollSuc be (δ, δ) -secure puncturable SKFE with indistinguishability of functionality that supports μ functional keys and is weakly collusion-succinct with compression factor γ , where $\gamma < 1$ is a constant. 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 $\gamma < \gamma' < 1$.*

Proof of Theorem 6.2 We start with analyzing the weak succinctness of WeakSuc, and then move on to the security proof.

Efficiency Let En_{dre}^i denote the circuit $\text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}]$. In order to issue one functional key, the construction needs to issue $1 \cdot \mu$ keys of CollSuc and it holds $\mu \leq s \cdot \text{poly}_{\text{RE}}(\lambda, n)$ since we consider functions of size s and n -bit input. Thus, we choose μ as the number of functional keys of CollSuc given to adversaries. The size of En_{dre}^i is bounded by $\text{poly}_{\text{En}}(\lambda, n, \log s)$ since the size of \widehat{f}_i and CT_i^{ske} are independent of s from the decomposability of RE, t is a λ -bit string, and the running time of the PRF evaluation in En_{dre}^i is logarithmic in s , where poly_{En} is a polynomial. Since CollSuc is collusion-succinct, the encryption time of WeakSuc is bounded by

$$\begin{aligned} \mu^\gamma \cdot \text{poly}(\lambda, n, |\text{En}_{\text{dre}}^i|) &\leq (s \cdot \text{poly}_{\text{RE}}(\lambda, n))^\gamma \cdot \text{poly}(\lambda, n, \text{poly}_{\text{En}}(\lambda, n, \log s)) \\ &\leq s^{\gamma'} \cdot \text{poly}(\lambda, n) , \end{aligned}$$

where γ and γ' are constants such that $0 < \gamma < \gamma' < 1$. This implies that WeakSuc is weakly succinct.

Indistinguishability of functionality under puncturing WS.Enc and WS.PEnc just outputs a ciphertext output by CS.Enc and CS.PEnc, respectively. Therefore, we can see that if CollSuc satisfies δ -indistinguishability of functionality under puncturing, then so does WeakSuc.

Semantic security at punctured tag Let \mathcal{A} be an adversary that attacks the semantic security at punctured tag of WeakSuc. We prove it via sequence of games. Below, for every $\ell \in \{0, \dots, 4\}$, let SUC_ℓ be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game ℓ .

Game 0: This is the punctured semantic security game regarding WeakSuc. Then, we have $\text{Adv}_{\text{WeakSuc}, \mathcal{A}}^{\text{SS}}(\lambda) = 2|\Pr[\text{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger generates $\text{MSK} \leftarrow \text{CS.Setup}(1^\lambda)$. The challenger also chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. The challenger sends security parameter 1^λ to \mathcal{A} .
2. \mathcal{A} sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\text{tag} \in \mathcal{T}$, and a function f to the challenger.
3. The challenger generate $S_{\text{encd}} \xleftarrow{r} \{0, 1\}^\lambda$ and computes $\text{CT} \leftarrow \text{CS.Enc}(\text{MSK}, \text{tag}, (0, m_b, S_{\text{encd}}, \perp))$. Next, the challenger generates $K \xleftarrow{r} \{0, 1\}^\lambda$ and $t \xleftarrow{r} \{0, 1\}^\lambda$, and computes $\widehat{f} \leftarrow \text{RE.E}(1^\lambda, f)$ and decomposed encodings $\widehat{f}_1 \cdots \widehat{f}_\mu$. Then, the challenger generates $\text{CT}_i^{\text{ske}} \leftarrow \text{E}(K, 0^{|\widehat{f}_i(\cdot, \cdot)|})$, and computes $\text{sk}_{\text{En}_i} \leftarrow \text{CS.KG}(\text{MSK}, \text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}], i)$ for every $i \in [\mu]$. Moreover, the challenger sets $\text{sk}_f \leftarrow (\text{sk}_{\text{En}_1}, \dots, \text{sk}_{\text{En}_\mu})$. Then, the challenger computes $\text{MSK}^*\{\text{tag}\} \leftarrow \text{CS.Punc}(\text{MSK}, \text{tag})$. The challenger returns $(\text{MSK}^*\{\text{tag}\}, \text{CT}, \text{sk}_f)$ to \mathcal{A} .
4. \mathcal{A} outputs $b' \in \{0, 1\}$.

Game 1 Same as Game 0 except that the challenger generates $\text{CT}_i^{\text{ske}} \leftarrow \text{E}(K, e_i)$ for every $i \in [\mu]$, where $e_i \leftarrow \widehat{f}_i(m_b; r)$.

In Game 0 and 1, \mathcal{A} is not given any information of secret key K of SKE. Therefore, from the security guarantee of SKE, we have $|\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| \leq \delta^{\Omega(1)}$.

Game 2 Same as Game 1 except that the challenger generates $\text{CT} \leftarrow \text{CS.Enc}(\text{MSK}, \text{tag}, (1, \perp, \perp, K))$.

We can see that for every $i \in [\mu]$, we have

$$\begin{aligned} \text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}](0, m_b, S_{\text{encd}}, \perp) &= \widehat{f}_i(m_b; r) \\ &= \text{En}_{\text{dre}}[\widehat{f}_i, t, \text{CT}_i^{\text{ske}}](1, \perp, \perp, K). \end{aligned}$$

Therefore, from the semantic security at punctured tag of CollSuc, it holds that $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \leq \delta^{\Omega(1)}$.

Game 3 Same as Game 2 except that the challenger generates r as a truly random string.

From the pseudorandomness of PRF, we have $|\Pr[\text{SUC}_2] - \Pr[\text{SUC}_3]| \leq \delta^{\Omega(1)}$.

Game 4 Same as Game 3 except that the challenger generates $\{e_i\}_{i \in [\mu]} \leftarrow \text{RE.Sim}(1^\lambda, s, y)$, where RE.Sim is a simulator for RE and $y = f(m_0) = f(m_1)$.

In Game 3 and 4, for every $i \in [\mu]$, e_i hardwired into En_{dre} after encrypted is generated with a truly random string. Therefore, from the security guarantee of RE, we have $|\Pr[\text{SUC}_3] - \Pr[\text{SUC}_4]| \leq \delta^{\Omega(1)}$.

In Game 4, the choice of the challenge bit b is information theoretically hidden from the view of \mathcal{A} , and thus we have $|\Pr[\text{SUC}_4] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned} \frac{1}{2} \text{Adv}_{\text{WeakSuc}, \mathcal{A}}^{\text{ss}}(\lambda) &= |\Pr[\text{SUC}_0] - \frac{1}{2}| \\ &\leq |\Pr[\text{SUC}_0] - \Pr[\text{SUC}_4]| \\ &\leq \sum_{\ell=0}^3 |\Pr[\text{SUC}_\ell] - \Pr[\text{SUC}_{\ell+1}]|. \end{aligned} \quad (6)$$

From the above argument, each term of the right side of inequality 6 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\text{Adv}_{\text{WeakSuc}, \mathcal{A}}^{\text{ss}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of \mathcal{A} is arbitrary, WeakSuc satisfies δ -semantic security at punctured tag. \square (**Theorem 6.2**)

7 Indistinguishability Obfuscation from Puncturable SKFE

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

Theorem 7.1. *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.*

In addition, by combining Theorems 3.23, 5.1, 6.1 and 6.2, we also obtain the following theorem.

Theorem 7.2. *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 7.2, we also use δ -secure PRF, puncturable PRF, plain SKE, garbling scheme, and decomposable randomized encoding as building blocks. From Theorems 3.2, 3.4, 3.6, 3.8 and 3.10, all of these primitives are implied by δ -secure one-way functions thus implied by δ -secure collusion-resistant SKFE for all circuits.

By combining Theorems 7.1 and 7.2, we obtain the following main theorem.

Theorem 7.3. *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 7.4 (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 [KS20]. 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.

To prove Theorem 7.1, we first give the construction of IO based on puncturable SKFE. Then, we analyze its security and efficiency.

7.1 Construction

Our construction of IO is almost the same as that of Bitansky and Vaikuntanathan [BV15]. The notable difference is that we use the relaxed variant of puncturable SKFE in Definition 4.4 instead of PKFE or their puncturable SKFE. Thus, the security analysis of our IO is different from and more complex than that of Bitansky and Vaikuntanathan.

Let pSKFE = (Setup, KG, Enc, Dec, Punc, PEnc) be a single-key weakly succinct puncturable SKFE scheme. Let SKE = (E, D) be an SKE scheme and PPRF = (F, Punc_F) a puncturable PRF. Below, let $\tilde{\lambda}$ denote the security parameter given to these building block schemes. Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$, where $\epsilon < 1$ is a constant. We assume that pSKFE is a (δ, δ) -secure puncturable SKFE with indistinguishability of functionality under puncturing. In addition, we assume that SKE and PPRF are δ -secure. Note that the existence of such SKE and PPRF are implied by that of pSKFE. Using pSKFE, SKE, and PPRF, we construct an indistinguishability obfuscation $i\mathcal{O}$ as follows.

Given a circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and a security parameter λ , the obfuscator $i\mathcal{O}$ first sets the security parameter $\tilde{\lambda}$ for building block schemes as $\tilde{\lambda} = \omega((n^2 + \log \lambda)^{1/\epsilon})$. $i\mathcal{O}$ uses pSKFE whose tag space and message space is $\{0, 1\}^n$ and $\{0, 1\}^n \times \{0, 1\}^{\tilde{\lambda}} \times \{0, 1\}$, respectively. $i\mathcal{O}$ also uses PPRF whose domain is $\{0, 1\}^n$. When a shorter string than expected is used as an input to these schemes, we always consider that it is fed after padded to the appropriate length. $i\mathcal{O}$ invokes the following recursive obfuscation procedure $ri\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ in order to obfuscate C .

$ri\mathcal{O}(1^{\tilde{\lambda}}, i, C_i) :$

- If $i = 1$, return $\tilde{C}_i \leftarrow (C_i(0), C_i(1))$. Note that the input domain of C_i is $\{0, 1\}^i$.
- Else, runs as follows:
 - Generate $K_{i,0}, K_{i,1} \xleftarrow{r} \{0, 1\}^{\tilde{\lambda}}$ and compute $\text{CT}_{i,0}^{\text{ske}} \leftarrow \text{E}(K_{i,0}, C_i)$ and $\text{CT}_{i,1}^{\text{ske}} \leftarrow \text{E}(K_{i,1}, C_i)$.
 - Generate $\text{MSK}_i \leftarrow \text{Setup}(1^{\tilde{\lambda}})$ and compute $\text{sk}_{\text{Ev}_i} \leftarrow \text{KG}(\text{MSK}_i, \text{Ev}_i[\text{CT}_{i,0}^{\text{ske}}, \text{CT}_{i,1}^{\text{ske}}])$. The circuit Ev_i is defined in Figure 13.

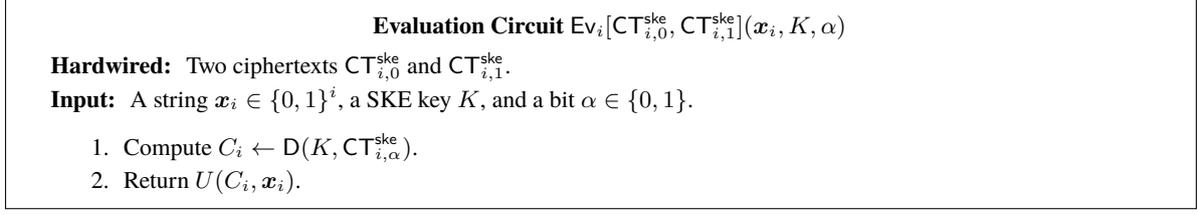


Figure 13: The description of Ev_i for every $i \in \{2, \dots, n\}$. In the description, $U(\cdot, \cdot)$ is a universal circuit.

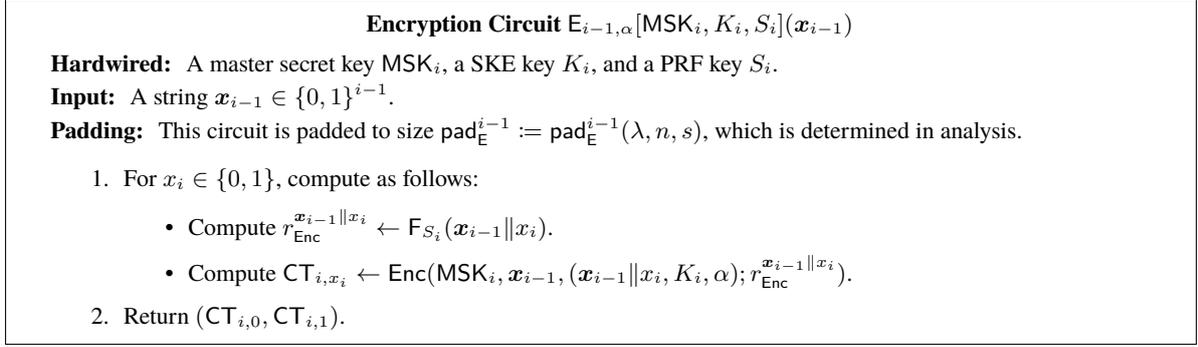


Figure 14: The description of $\text{E}_{i-1,\alpha}$ for every $i \in \{3, \dots, n\}$ and $\alpha \in \{0, 1\}$.

- Generate $S_i \xleftarrow{r} \{0, 1\}^{\tilde{\lambda}}$ and compute $\tilde{\text{E}}_{i-1} \leftarrow \text{riO}(1^{\tilde{\lambda}}, i-1, \text{E}_{i-1,0}[\text{MSK}_i, K_{i,0}, S_i])$.
The circuit $\text{E}_{i-1,0}$ is defined in Figure 21.
- Return $\tilde{C}_i := (\text{sk}_{\text{Ev}_i}, \tilde{\text{E}}_{i-1})$.

The corresponding recursive evaluation procedure is as follows.

$\text{rEval}(i, \tilde{C}_i, \mathbf{x}_i)$:

- If $i = 1$, parse $(\text{CT}_{1,0}, \text{CT}_{1,1}) \leftarrow \tilde{C}_i$ and return CT_{1,x_i} .
- Else, runs as follows:
 - Parse $(\text{sk}_{\text{Ev}_i}, \tilde{\text{E}}_{i-1}) \leftarrow \tilde{C}_i$ and $\mathbf{x}_{i-1} \| x_i \leftarrow \mathbf{x}_i$.
 - Compute $(\text{CT}_{i,0}, \text{CT}_{i,1}) \leftarrow \text{rEval}(i-1, \tilde{\text{E}}_{i-1}, \mathbf{x}_{i-1})$.
 - Return $y \leftarrow \text{Dec}(\text{sk}_{\text{Ev}_i}, \mathbf{x}_{i-1}, \text{CT}_{i,x_i})$.

By using these recursive procedures, our obfuscator and its evaluation algorithm are described as follows.

$i\text{O}(1^\lambda, C)$: To obfuscate $C : \{0, 1\}^n \rightarrow \{0, 1\}^m$, it does the following.

- Set $\tilde{\lambda} = \omega((n^2 + \log \lambda)^{1/\epsilon})$.
- Output $\text{riO}(1^{\tilde{\lambda}}, n, C)$.

$\text{Eval}(\tilde{C}, \mathbf{x})$: To compute $\tilde{C}(\mathbf{x}) = i\text{O}(1^\lambda, C)(\mathbf{x})$ where $\mathbf{x} \in \{0, 1\}^n$, it does the following.

- Output $\text{rEval}(n, \tilde{C}, \mathbf{x})$.

Correctness We prove that our obfuscator satisfies correctness by induction on $i \in [n]$ (input length). Our goal is to prove that for any C_i and $\tilde{E}_i \leftarrow ri\mathcal{O}(1^{\tilde{\lambda}}, i, C_i)$, it holds that

$$\forall \mathbf{x}_i \in \{0, 1\}^i \quad r\text{Eval}(i, \tilde{E}_i, \mathbf{x}_i) = C_i(\mathbf{x}_i). \quad (7)$$

For the base case $i = 1$, it holds $\tilde{E}_1 = ri\mathcal{O}(1^{\tilde{\lambda}}, 1, C_1) = (C_1(0), C_1(1))$ since $i = 1$. Thus, Equation (7) holds for $i = 1$ since $r\text{Eval}(1, \tilde{E}_1, \mathbf{x}_1) = C_1(\mathbf{x}_1)$ by the description of $r\text{Eval}$.

We assume that Equation (7) holds for $i - 1$. By the description of \tilde{E} , we obtain

$$\begin{aligned} \tilde{E}_i &= ri\mathcal{O}(1^{\tilde{\lambda}}, i, C_i) = (\text{sk}_{\text{Ev}_i}, \tilde{E}_{i-1}) \\ &= (\text{sk}_{\text{Ev}_i}, ri\mathcal{O}(1^{\tilde{\lambda}}, i-1, E_{i-1,0}[\text{MSK}_i, K_{i,0}, S_i])), \end{aligned}$$

where $\text{sk}_{\text{Ev}_i} \leftarrow \text{KG}(\text{MSK}_i, \text{Ev}_i[\text{CT}_{i,0}^{\text{ske}}, \text{CT}_{i,1}^{\text{ske}}])$. The circuit $E_{i-1,0}(\mathbf{x}_{i-1})$ computes $\text{CT}_{i,b} \leftarrow \text{Enc}(\text{MSK}_i, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| b, K_i, 0))$ for both $b = 0, 1$. Thus, by the assumption that Equation (7) holds for $i - 1$, it holds that

$$\begin{aligned} &r\text{Eval}(i-1, \tilde{E}_{i-1}, \mathbf{x}_{i-1}) \\ &= (\text{Enc}(\text{MSK}_i, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| 0, K_i, 0)), \text{Enc}(\text{MSK}_i, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| 1, K_i, 0))). \end{aligned}$$

Therefore, we obtain

$$\begin{aligned} r\text{Eval}(i, \tilde{E}_i, \mathbf{x}_i) &= \text{Dec}(\text{sk}_{\text{Ev}_i}, \mathbf{x}_{i-1}, \text{Enc}(\text{MSK}_i, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| \mathbf{x}_i, K_i, 0))) \\ &= \text{Ev}_i[\text{CT}_{i,0}^{\text{ske}}, \text{CT}_{i,1}^{\text{ske}}](\mathbf{x}_{i-1} \| \mathbf{x}_i, K_i, 0) \\ &= C_i(\mathbf{x}_i) \end{aligned}$$

by the correctness of pSKFE and the description of Ev_i . This completes the proof of correctness.

Remark 7.5 (On the parameter setting of $\tilde{\lambda}$). In the construction we set the security parameter $\tilde{\lambda}$ for building blocks as $\tilde{\lambda} = \omega((n^2 + \log \lambda)^{1/\epsilon})$. In fact, this setting is the same as that of Bitansky and Vaikuntanathan [BV15]. However, the security loss is different between this work and the work by Bitansky and Vaikuntanathan. In our construction, $2^{O(n^2)}$ security loss occurs while the construction of Bitansky and Vaikuntanathan incurs $2^{O(n^2/2)}$ loss. The difference occurs due to our additional exponential hybrids that we need to complete the security proof while the building block puncturable SKFE scheme satisfies only indistinguishability of functionality under puncturing property. For the detailed security analysis, see Section 7.2.

Note that the size of padding for the encryption circuit $E_{i-1,\alpha}$ is determined in the security analysis of our indistinguishability obfuscator $i\mathcal{O}$. We need to know the size of padding in order to analyze the efficiency of $i\mathcal{O}$. Therefore, we first analyze the security of $i\mathcal{O}$ in Section 7.2. Then, we analyze the efficiency of $i\mathcal{O}$ in Section 7.3. We complete the proof of Theorem 7.1 by completing the analysis of security and efficiency.

7.2 Security Analysis

Our goal is to prove that for any PPT distinguisher \mathcal{D} and circuits C_0 and C_1 of the same functionality, we have

$$\begin{aligned} &\left| \Pr[\mathcal{D}(i\mathcal{O}(1^\lambda, C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(1^\lambda, C_1)) = 1] \right| \\ &= \left| \Pr[\mathcal{D}(ri\mathcal{O}(1^{\tilde{\lambda}}, n, C_0)) = 1] - \Pr[\mathcal{D}(ri\mathcal{O}(1^{\tilde{\lambda}}, n, C_1)) = 1] \right| = \text{negl}(\lambda). \end{aligned}$$

In order to prove this, for every $i \in [n]$, we define

$$\delta_i := \max_{C_{i,0}, C_{i,1}} \left| \Pr \left[\mathcal{D}_i(\text{riO}(1^\lambda, i, C_{i,0})) = 1 \right] - \Pr \left[\mathcal{D}_i(\text{riO}(1^\lambda, i, C_{i,1})) = 1 \right] \right| ,$$

where \mathcal{D}_i is a PPT distinguisher and $C_{i,0}$ and $C_{i,1}$ are pair of any circuits with i -bit input that are the same functionality. Then, our goal is restated to show that $\delta_n \leq 2^{-\omega(\log \lambda)}$ holds.

Note that we have $\delta_1 = 0$. This is because circuits with 1-bit input $C_{1,0}$ and $C_{1,1}$ of the same functionality are both obfuscated to the same truth table. Our goal is to prove the following lemma.

Lemma 7.6. *Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Assuming that SKE and PPRF are δ -secure and pSKFE is a (δ, δ) -secure puncturable SKFE with indistinguishability of functionality. It holds that*

$$\delta_i \leq 2^{2(i-1)} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \quad (8)$$

for every $i \in \{2, \dots, n\}$.

By this lemma, we can estimate δ_n as

$$\begin{aligned} \delta_n &\leq 2^{2(n-1)} \cdot O(\delta_{n-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \\ &\leq 2^{2(n-1)} \cdot O(\delta_{n-1}) + 2^{2(n-1)} \cdot O(2^{-\Omega(\tilde{\lambda}^\epsilon)}) \\ &\leq \dots \leq \left(\sum_{i=1}^n \prod_{j=1}^i 2^{2(n-j)} \right) \cdot O(2^{-\Omega(\tilde{\lambda}^\epsilon)}) \leq n \cdot 2^{n^2} \cdot O(2^{-\omega(n^2 + \log \lambda)}) \leq 2^{-\omega(\log \lambda)} . \end{aligned}$$

This inequality shows that we complete the proof of Theorem 7.3.

Therefore, if we prove that inequality 8 holds for every $i \in \{2, \dots, n\}$, that is Lemma 7.6, we can conclude that our $i\mathcal{O}$ is a secure indistinguishability obfuscator. In the rest of this section, we prove that inequality 8 holds for every $i \in \{2, \dots, n\}$.

Let $i \in \{2, \dots, n\}$. Let \mathcal{D}_i be any PPT distinguisher again. In addition, let $C_{i,0}$ and $C_{i,1}$ be circuits with i -bit input of the same functionality that maximize the value of δ_i . First, we consider the following sequence of hybrid experiments.

\mathcal{H}_0 : In this experiment, \mathcal{D}_i is given an obfuscation of the circuit $C_{i,0}$, that is $\text{riO}(1^\lambda, i, C_{i,0})$.

\mathcal{H}_1 : Same as \mathcal{H}_0 except that $\text{CT}_{i,1}^{\text{ske}}$ is generated as $\text{CT}_{i,1}^{\text{ske}} \leftarrow \text{E}(K_{i,1}, C_{i,1})$. Note that in \mathcal{H}_0 , $\text{CT}_{i,1}^{\text{ske}}$ is generated as $\text{CT}_{i,1}^{\text{ske}} \leftarrow \text{E}(K_{i,1}, C_{i,0})$.

\mathcal{H}_2 : Same as \mathcal{H}_1 except that $\tilde{\text{E}}_{i-1}$ is generated as $\tilde{\text{E}}_{i-1} \leftarrow \text{riO}(1^\lambda, i-1, \text{E}_{i-1,1}[\text{MSK}_i, K_{i,1}, S_i])$. Note that in \mathcal{H}_1 , $\tilde{\text{E}}_{i-1}$ is generated as $\tilde{\text{E}}_{i-1} \leftarrow \text{riO}(1^\lambda, i-1, \text{E}_{i-1,0}[\text{MSK}_i, K_{i,0}, S_i])$.

\mathcal{H}_3 : Same as \mathcal{H}_2 except that $\text{CT}_{i,0}^{\text{ske}}$ is generated as $\text{CT}_{i,0}^{\text{ske}} \leftarrow \text{E}(K_{i,0}, C_{i,1})$. Note that in \mathcal{H}_2 , $\text{CT}_{i,0}^{\text{ske}}$ is generated as $\text{CT}_{i,0}^{\text{ske}} \leftarrow \text{E}(K_{i,1}, C_{i,0})$.

\mathcal{H}_4 : Same as \mathcal{H}_3 except that $\tilde{\text{E}}_{i-1}$ is generated as $\tilde{\text{E}}_{i-1} \leftarrow \text{riO}(1^\lambda, i-1, \text{E}_{i-1,0}[\text{MSK}_i, K_{i,0}, S_i])$. Note that in this experiment, the distribution of the input to \mathcal{D}_i is exactly the same as an obfuscation of the circuit $C_{i,1}$, that is $\text{riO}(1^\lambda, i, C_{i,1})$.

For an experiment \mathcal{H} , we let $\mathcal{D}_i(\mathcal{H})$ denote the event that \mathcal{D}_i outputs 1 in \mathcal{H} . Then, we can estimate δ_i as

$$\delta_i \leq \sum_{\ell=0}^3 |\Pr[\mathcal{D}_i(\mathcal{H}_\ell)] - \Pr[\mathcal{D}_i(\mathcal{H}_{\ell+1})]| . \quad (9)$$

In the following, by estimating each term of the right hand side of inequality 9, we prove that inequality 8 holds for every $i \in \{2, \dots, n\}$. We give relations of hybrid experiments in Figure 15 and 19 in order to see easily the dependence of hybrid experiments.

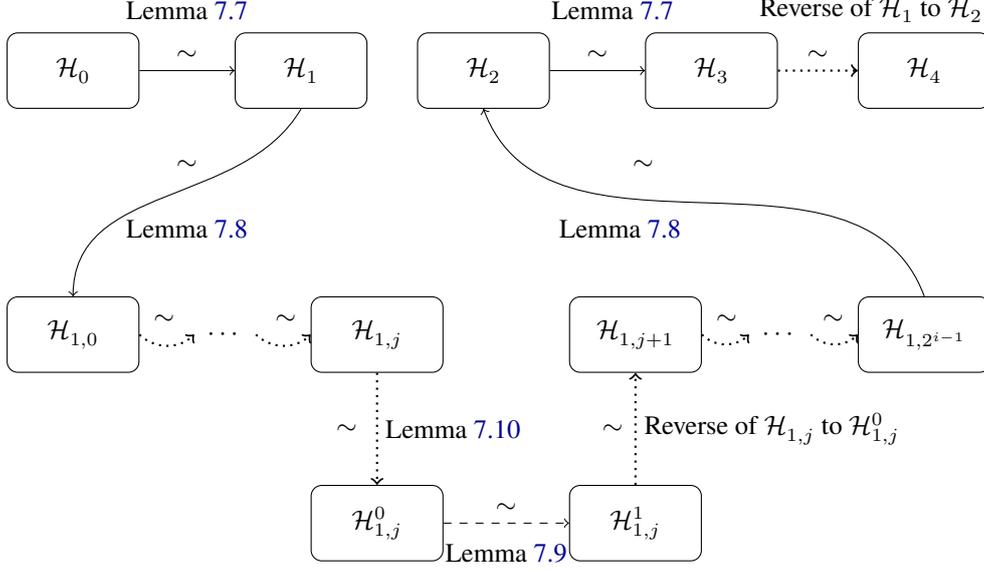


Figure 15: Relations of the hybrid experiments from \mathcal{H}_0 to \mathcal{H}_4 for the security of $ri\mathcal{O}$. Solid lines denote that the indistinguishability is proven by one step. Dashed lines denote that we use a few hybrid experiments to prove the indistinguishability. Dotted lines denote that we use many hybrid experiments to prove the indistinguishability (Figure 19 illustrates those of hybrid experiments for Lemma 7.10).

From \mathcal{H}_0 to \mathcal{H}_1 and From \mathcal{H}_2 to \mathcal{H}_3

First, we estimate $|\Pr[\mathcal{D}_i(\mathcal{H}_0)] - \Pr[\mathcal{D}_i(\mathcal{H}_1)]|$ and $|\Pr[\mathcal{D}_i(\mathcal{H}_2)] - \Pr[\mathcal{D}_i(\mathcal{H}_3)]|$. In fact, we can easily bound these values by the security of SKE. Formally, we have the following lemma.

Lemma 7.7. *Let SKE be δ -secure, where $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Then, it holds that $|\Pr[\mathcal{D}_i(\mathcal{H}_0)] - \Pr[\mathcal{D}_i(\mathcal{H}_1)]| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ and $|\Pr[\mathcal{D}_i(\mathcal{H}_2)] - \Pr[\mathcal{D}_i(\mathcal{H}_3)]| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$.*

The proof of this lemma is straightforward and thus we omit it.

From \mathcal{H}_1 to \mathcal{H}_2 and From \mathcal{H}_3 to \mathcal{H}_4

Next, we estimate $|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]|$ and $|\Pr[\mathcal{D}_i(\mathcal{H}_3)] - \Pr[\mathcal{D}_i(\mathcal{H}_4)]|$. Since the difference between \mathcal{H}_1 and \mathcal{H}_2 , and that of \mathcal{H}_3 and \mathcal{H}_4 are almost symmetric, we focus on estimating $|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]|$ here. We can apply the following arguments for the estimation of $|\Pr[\mathcal{D}_i(\mathcal{H}_3)] - \Pr[\mathcal{D}_i(\mathcal{H}_4)]|$.

In order to accomplish the estimation of $|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]|$, we first introduce intermediate hybrid experiments $\mathcal{H}_{1,j}$ between \mathcal{H}_1 and \mathcal{H}_2 , where $j \in \{0, \dots, 2^{i-1}\}$. In the following, let $\mathbf{j} \in \{0, 1\}^{i-1} \cup \{1\|0^{i-1}\}$ be the binary representation of j .

$\mathcal{H}_{1,j}$: In this experiment, $\tilde{\mathbf{E}}_{i-1}$ is computed as $\tilde{\mathbf{E}}_{i-1} \leftarrow ri\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathbf{E}_{i-1}^{\mathbf{j}}[\text{MSK}_i, K_{i,0}, K_{i,1}, S_i])$. The circuit $\mathbf{E}_{i-1}^{\mathbf{j}}$ is defined in Figure 16.

Encryption Circuit $E_{i-1}^j[\text{MSK}_i, K_{i,0}, K_{i,1}, S_i](\mathbf{x}_{i-1})$

Hardwired: A master secret key MSK_i , two SKE keys $K_{i,0}$ and $K_{i,1}$, and a PRF key S_i .

Input: A string $\mathbf{x}_{i-1} \in \{0, 1\}^{i-1}$.

Padding: This circuit is padded to size $\text{pad}_E^{i-1} := \text{pad}_E^{i-1}(\lambda, n, s)$, which is determined in analysis.

1. If $\mathbf{x}_{i-1} < \mathbf{j}$, set $\alpha \leftarrow 1$, and otherwise, set $\alpha \leftarrow 0$.
2. For $x_i \in \{0, 1\}$, compute as follows:
 - Compute $r_{\text{Enc}}^{\mathbf{x}_{i-1} \| x_i} \leftarrow F_{S_i}(\mathbf{x}_{i-1} \| x_i)$.
 - Compute $\text{CT}_{i,x_i} \leftarrow \text{Enc}(\text{MSK}, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1} \| x_i, K_{i,\alpha}, \alpha); r_{\text{Enc}}^{\mathbf{x}_{i-1} \| x_i})$.
3. Return $(\text{CT}_{i,0}, \text{CT}_{i,1})$.

Figure 16: The description of E_{i-1}^j . The red underline is the difference from $E_{i-1,\alpha}$.

Then, we have

$$\begin{aligned} |\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]| &\leq \left| \Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})] \right| \\ &\quad + \sum_{j=1}^{2^{i-1}-1} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})] \right| \\ &\quad + \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)] \right|. \end{aligned}$$

If we use a PKFE scheme instead of our puncturable SKFE scheme, we can directly prove the indistinguishability between $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j+1}$. However, to estimate each term of the right hand side of the above inequality, we need introduce the following additional hybrid experiments for every $j \in \{1, \dots, 2^{i-1}\}$ since we use a puncturable SKFE scheme.

$\mathcal{H}_{1,j}^0$: \tilde{E}_{i-1} is computed as $\tilde{E}_{i-1} \leftarrow \text{riO}(1^\lambda, i-1, \text{PE}_{i-1}^j[\text{MSK}_i^* \{j\}, K_{i,0}, K_{i,1}, S_i \{j \| 0, j \| 1\}, u_{i,0}, u_{i,1}])$ in this experiment, where $u_{i,b} \leftarrow \text{Enc}(\text{MSK}_i, j, (j \| b, K_{i,0}, 0); r_{\text{Enc}}^{j \| b})$ and $r_{\text{Enc}}^{j \| b} \leftarrow F_{S_i}(j \| b)$ for every $b \in \{0, 1\}$. The circuit PE_{i-1}^j is defined in Figure 17. The other part of this experiment is same as $\mathcal{H}_{1,j}$.

$\mathcal{H}_{1,j}^1$: Same as $\mathcal{H}_{1,j}^0$ except that $u_{i,b}$ is computed by $u_{i,b} \leftarrow \text{Enc}(\text{MSK}, j, (j \| b, K_{i,1}, 1); r_{\text{Enc}}^{j \| b})$ and $r_{\text{Enc}}^{j \| b} \leftarrow F_{S_i}(j \| b)$ for every $b \in \{0, 1\}$.

Then, we can estimate $|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]|$ in more detail and obtain

$$\begin{aligned} |\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]| &\leq \left| \Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})] \right| \\ &\quad + \sum_{j=0}^{2^{i-1}-1} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right| \\ &\quad + \sum_{j=0}^{2^{i-1}-1} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right| \\ &\quad + \sum_{j=0}^{2^{i-1}-1} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})] \right| \\ &\quad + \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)] \right|. \end{aligned} \tag{10}$$

In the rest of this section, we estimate each term of the right side of inequality 10 by Lemma 7.8, 7.9, and 7.10.

<p>Punctured Encryption Circuit $\text{PE}_{i-1}^j[\text{MSK}_i^*\{j\}, K_{i,0}, K_{i,1}, S_i^*\{j\ 0, j\ 1\}, u_{i,0}, u_{i,1}](\mathbf{x}_{i-1})$</p> <p>Hardwired: A punctured master secret key $\text{MSK}_i^*\{j\}$, two SKE keys $K_{i,0}$ and $K_{i,1}$, a punctured PRF key $S_i^*\{j\ 0, j\ 1\}$, and two ciphertexts $u_{i,0}$ and $u_{i,1}$.</p> <p>Input: A string $\mathbf{x}_{i-1} \in \{0, 1\}^{i-1}$.</p> <p>Padding: This circuit is padded to size $\text{pad}_{\mathbb{E}}^{i-1} := \text{pad}_{\mathbb{E}}^{i-1}(\lambda, n, s)$, which is determined in analysis.</p> <ol style="list-style-type: none"> 1. If $\mathbf{x}_{i-1} = \mathbf{j}$, return $(u_{i,0}, u_{i,1})$. 2. Else, set $\alpha := 1$ if $\mathbf{x}_{i-1} < \mathbf{j}$ and $\alpha := 0$ otherwise. 3. For $x_i \in \{0, 1\}$, compute as follows: <ul style="list-style-type: none"> • Compute $r_{\text{Enc}}^{\mathbf{x}_{i-1}\ x_i} \leftarrow F_{S_i^*\{j\ 0, j\ 1\}}(\mathbf{x}_{i-1}\ x_i)$. • Compute $\text{CT}_{i,x_i} \leftarrow \text{PEnc}(\text{MSK}_i^*\{j\}, \mathbf{x}_{i-1}, (\mathbf{x}_{i-1}\ x_i, K_{i,\alpha}, \alpha); r_{\text{Enc}}^{\mathbf{x}_{i-1}\ x_i})$. 4. Return $(\text{CT}_{i,0}, \text{CT}_{i,1})$.

Figure 17: The description of PE_{i-1}^j . Red underlines are differences from E_{i-1}^j .

From \mathcal{H}_1 to $\mathcal{H}_{1,0}$ and from $\mathcal{H}_{1,2^{i-1}}$ to \mathcal{H}_2 We have the following lemma.

Lemma 7.8. $\left| \Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})] \right| \leq \delta_{i-1}$ and $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)] \right| \leq \delta_{i-1}$ hold.

Proof of Lemma 7.8 We focus on proving $\left| \Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})] \right| \leq \delta_{i-1}$ here.

The only difference between \mathcal{H}_1 and $\mathcal{H}_{1,0}$ is how $\tilde{\mathbb{E}}_{i-1}$ is generated. In \mathcal{H}_1 , $\tilde{\mathbb{E}}_{i-1}$ is generated by $\tilde{\mathbb{E}}_{i-1} \leftarrow \text{riO}(1^{\tilde{\lambda}}, i-1, \mathbb{E}_{i-1,0}[\text{MSK}_i, K_{i,0}, S_i])$. On the other hand, in $\mathcal{H}_{1,0}$, $\tilde{\mathbb{E}}_{i-1}$ is generated by $\tilde{\mathbb{E}}_{i-1} \leftarrow \text{riO}(1^{\tilde{\lambda}}, i-1, \mathbb{E}_{i-1}^0[\text{MSK}_i, K_{i,0}, K_{i,1}, S_i])$. We can see that circuits obfuscated in each experiment have the same functionality. Moreover, both circuits are padded to the same size $\text{pad}_{\mathbb{E}}^{i-1}$. Therefore, we have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})] \right| \leq \delta_{i-1}$.

By analyzing similarly, we also obtain $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)] \right| \leq \delta_{i-1}$. \square (**Lemma 7.8**)

From $\mathcal{H}_{1,j}^0$ to $\mathcal{H}_{1,j}^1$ Next, we estimate $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right|$ for every $j \in \{0, \dots, 2^{i-1} - 1\}$.

We can estimate $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right|$ by the semantic security at punctured tag of pSKFE and the security of PPRF. Formally, we have the following lemma.

Lemma 7.9. Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Let pSKFE satisfy δ -semantic security at punctured tag. Let PPRF be δ -secure. Then, $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ holds for every $i \in \{0, \dots, 2^{i-1} - 1\}$.

Proof of Lemma 7.9 In order to use the semantic security at punctured tag of pSKFE, we change the randomness $r_{\text{Enc}}^{j\|b}$ used in $u_{i,b}$ into truly random for every $b \in \{0, 1\}$. Thus, we introduce the following intermediate hybrid experiments $\mathcal{H}_{1,j}^{0,\text{rnd}}$ and $\mathcal{H}_{1,j}^{1,\text{rnd}}$ between $\mathcal{H}_{1,j}^0$ and $\mathcal{H}_{1,j}^1$.

$\mathcal{H}_{1,j}^{0,\text{rnd}}$: Same as $\mathcal{H}_{1,j}^0$ except that $r_{\text{Enc}}^{j\|b}$ is generated as a truly random string for every $b \in \{0, 1\}$.

By the pseudorandomness of PPRF, we have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^{0,\text{rnd}})] \right| \leq 2^{-\tilde{\lambda}^\epsilon}$.

$\mathcal{H}_{1,j}^{1,\text{rnd}}$: Same as $\mathcal{H}_{1,j}^1$ except that $r_{\text{Enc}}^{j\|b}$ is generated as a truly random string for every $b \in \{0, 1\}$.

Note that the only difference between $\mathcal{H}_{1,j}^{0,\text{rnd}}$ and $\mathcal{H}_{1,j}^{1,\text{rnd}}$ is how $u_{i,b}$ is generated for every $b \in \{0, 1\}$. In $\mathcal{H}_{1,j}^{0,\text{rnd}}$, $u_{i,b}$ is generated by $u_{i,b} \leftarrow \text{Enc}(\text{MSK}_i, \mathbf{j}, (j\|b, K_{i,0}, 0))$. On the other hand, in $\mathcal{H}_{1,j}^{1,\text{rnd}}$,

$u_{i,b}$ is generated by $u_{i,b} \leftarrow \text{Enc}(\text{MSK}_i, \mathbf{j}, (\mathbf{j} \| b, K_{i,1}, 1))$. Now, in both experiments, the ciphertext $u_{i,b}$ is generated using a truly random string for every $b \in \{0, 1\}$. In addition, in both experiments, $\text{CT}_{i,\alpha}^{\text{ske}}$ is a ciphertext of $C_{i,\alpha}$ under the SKE key $K_{i,\alpha}$ for every $\alpha \in \{0, 1\}$. Hence, we have

$$\begin{aligned} \text{Ev}_i[\text{CT}_{i,0}^{\text{ske}}, \text{CT}_{i,1}^{\text{ske}}](\mathbf{j} \| b, K_{i,0}, 0) &= C_{i,0}(\mathbf{j} \| b) \\ &= C_{i,1}(\mathbf{j} \| b) = \text{Ev}_i[\text{CT}_{i,0}^{\text{ske}}, \text{CT}_{i,1}^{\text{ske}}](\mathbf{j} \| b, K_{i,1}, 1), \end{aligned}$$

since $C_{i,0}$ and $C_{i,1}$ are functionally equivalent.

Therefore, from the semantic security at punctured tag of pSKFE, we obtain

$$\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^{0,\text{rnd}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^{1,\text{rnd}})] \right| \leq 2^{-\tilde{\lambda}^\epsilon}.$$

By the pseudorandomness of PPRF, we have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^{1,\text{rnd}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right| \leq 2^{-\tilde{\lambda}^\epsilon}$.

From these, we see that $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] \right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$. \square (**Lemma 7.9**)

From $\mathcal{H}_{1,j}$ to $\mathcal{H}_{1,j}^0$ and from $\mathcal{H}_{1,j}^1$ to $\mathcal{H}_{1,j+1}$ In the rest of this proof, we estimate $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right|$ and $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})] \right|$ for every $j \in \{0, \dots, 2^{i-1} - 1\}$. Since the difference between $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j}^0$, and that of $\mathcal{H}_{1,j}^1$ and $\mathcal{H}_{1,j+1}$ are almost symmetric, we focus on evaluating $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right|$ here. More precisely, we prove the following lemma.

Lemma 7.10. *Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Let pSKFE satisfy δ -indistinguishability of functionality under puncturing. Let PPRF be δ -secure. Then, for every $\{0, \dots, 2^{i-1} - 1\}$, we have*

$$\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right| \leq 2^{i-1} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}).$$

We can apply the following arguments for the evaluation of $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})] \right|$.

Proof of Lemma 7.10 If the underlying puncturable SKFE satisfies functionality preserving under puncturing property, we can directly estimate $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right|$ for every $j \in \{0, \dots, 2^{i-1} - 1\}$ by using the property. However, our pSKFE satisfies only indistinguishability of functionality under puncturing property. Thus, we need more hybrid experiments between $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j}^0$ defined for every $k \in \{0, \dots, 2^{i-1}\}$ as follows. Below, let $\mathbf{k} \in \{0, 1\}^{i-1} \cup \{1\} \| 0^{i-1}$ be the binary representation of k .

$\mathcal{H}_{1,j,k}$: $\tilde{\mathbf{E}}_{i-1}$ is computed as $\tilde{\mathbf{E}}_{i-1} \leftarrow \text{riO}(1^{\tilde{\lambda}}, i-1, \text{HE}_{i-1}^{j,k}[\text{MSK}_i, \text{MSK}_i^* \{ \mathbf{j} \}, K_{i,0}, K_{i,1}, S_i, \perp, \perp])$ in this experiment. The circuit $\text{HE}_{i-1}^{j,k}$ is defined in Figure 18.

Figure 19 illustrates an overview of hybrid experiments from $\mathcal{H}_{1,j-1}$ to $\mathcal{H}_{1,j}^0$.

Then, we have

$$\begin{aligned} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right| &\leq \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,0})] \right| \\ &\quad + \sum_{k=0}^{2^{i-1}-1} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k+1})] \right| \\ &\quad + \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right|. \end{aligned}$$

To estimate each term of the right hand side of the above inequality, we introduce the following hybrid experiments for $k \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$.

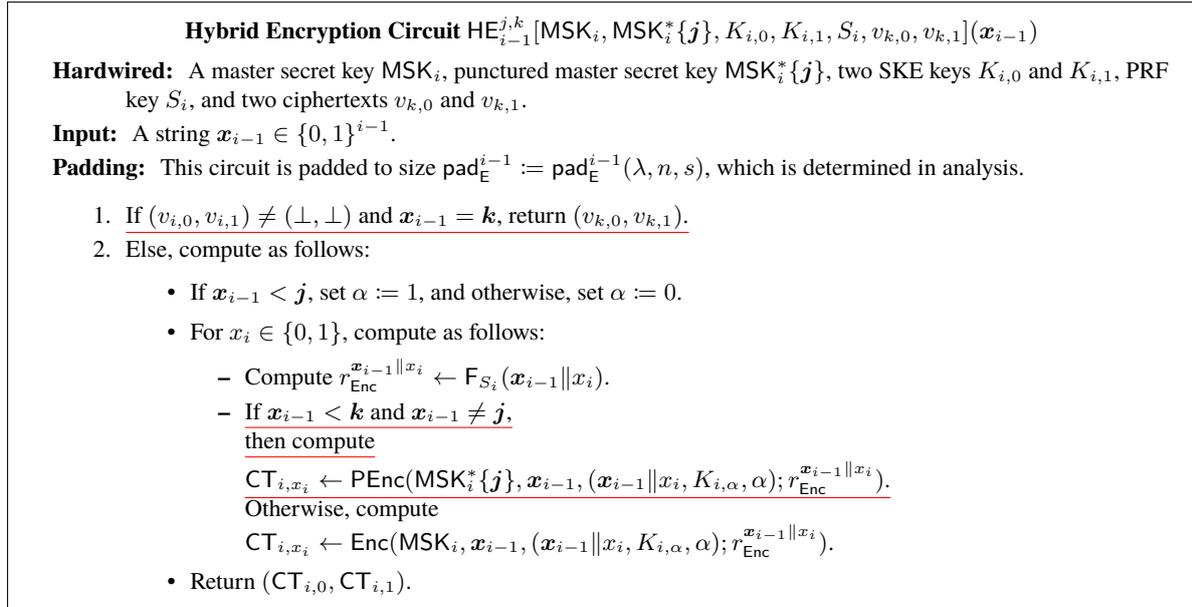


Figure 18: The description of $\text{HE}_{i-1}^{j,k}$. Red underlines are differences from E_{i-1}^j .

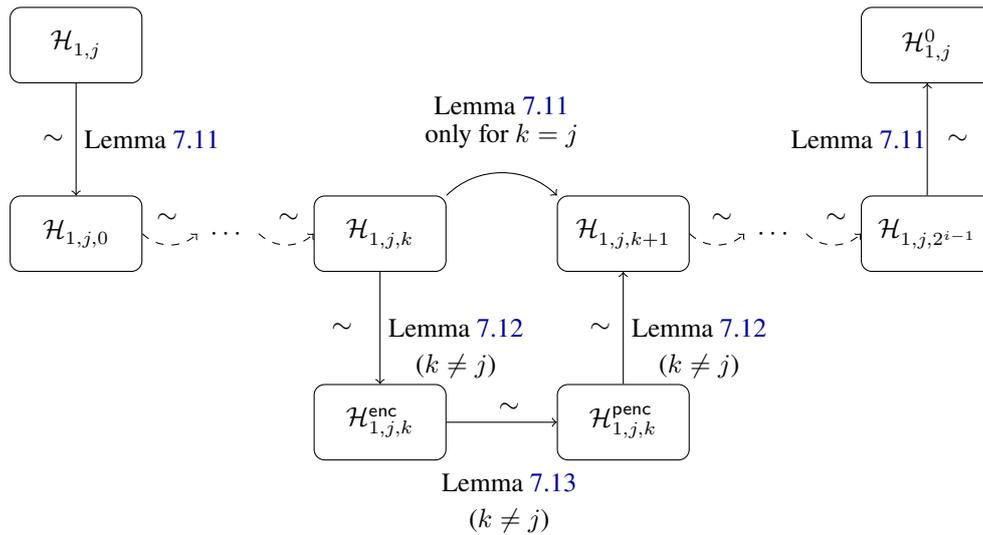


Figure 19: Relations of the hybrid experiments from $\mathcal{H}_{1,j-1}$ to $\mathcal{H}_{1,j}^0$ for Lemma 7.10. Solid lines denote that the indistinguishability is proven by one step. Dashed lines denote that we use for loop with variable k . Note that, only for $k = j$, we do not need $\mathcal{H}_{1,j,k}^{\text{enc}}$ and $\mathcal{H}_{1,j,k}^{\text{penc}}$ ($\mathcal{H}_{1,j,j}^{\text{penc}}$ is not well-defined).

$\mathcal{H}_{1,j,k}^{\text{enc}}$: $\tilde{\mathbb{E}}_{i-1}$ is computed as $\tilde{\mathbb{E}}_{i-1} \leftarrow \text{riO}(1^\lambda, i-1, \text{HE}_{i-1}^{j,k}[\text{MSK}_i, \text{MSK}_i^*\{j\}, K_{i,0}, K_{i,1}, S_i^*\{\mathbf{k} \parallel 0, \mathbf{k} \parallel 1\}, v_{k,0}, v_{k,1}])$ in this experiment, where $v_{k,b} \leftarrow \text{Enc}(\text{MSK}_i, \mathbf{k}, (\mathbf{k} \parallel b, K_{i,\alpha}, \alpha); r_{\text{Enc}}^{\mathbf{k} \parallel b})$ for every $b \in \{0, 1\}$.

$\mathcal{H}_{1,j,k}^{\text{penc}}$: In this experiment, $v_{k,b}$ is computed by $v_{k,b} \leftarrow \text{PEnc}(\text{MSK}_i^*\{j\}, \mathbf{k}, (\mathbf{k} \parallel b, K_{i,\alpha}, \alpha); r_{\text{Enc}}^{\mathbf{k} \parallel b})$ for every $b \in \{0, 1\}$.

Then, we can estimate $\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}^0) \right] \right|$ in more detail and obtain

$$\begin{aligned}
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}^0) \right] \right| &\leq \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,0}) \right] \right| \\
&+ \sum_{k \in \{0, \dots, 2^{i-1}-1\} \setminus \{j\}} \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}}) \right] \right| \\
&+ \sum_{k \in \{0, \dots, 2^{i-1}-1\} \setminus \{j\}} \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}}) \right] \right| \\
&+ \sum_{k \in \{0, \dots, 2^{i-1}-1\} \setminus \{j\}} \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k+1}) \right] \right| \\
&+ \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,j+1}) \right] \right| \\
&+ \left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,2^{i-1}}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}^0) \right] \right|. \tag{11}
\end{aligned}$$

To bound the right hand side of inequality 11, we prove Lemma 7.11, 7.12, and 7.13.

Lemma 7.11. *It holds that*

$$\begin{aligned}
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,0}) \right] \right| &\leq \delta_{i-1} \text{ ,} \\
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,j}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,j+1}) \right] \right| &\leq \delta_{i-1} \text{ ,} \\
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,2^{i-1}}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j}^0) \right] \right| &\leq \delta_{i-1} \text{ .}
\end{aligned}$$

Lemma 7.12. *For $k \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$, it holds that*

$$\begin{aligned}
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}}) \right] \right| &\leq \delta_{i-1} \text{ ,} \\
\left| \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}}) \right] - \Pr \left[\mathcal{D}_i(\mathcal{H}_{1,j,k+1}) \right] \right| &\leq \delta_{i-1} \text{ .}
\end{aligned}$$

Proof of Lemma 7.11 The only difference between two experiments in each inequality is how \tilde{E}_{i-1} is generated. Thus, we verify that circuits of the same functionality are obfuscated to generate \tilde{E}_{i-1} in those two experiments related to each inequality.

For the first inequality, it is easy to verify that E_{i-1}^j in $\mathcal{H}_{1,j}$ is equivalent to $HE_{i-1}^{j,0}$ in $\mathcal{H}_{1,j,0}$ since $HE_{i-1}^{j,0}$ does not compute PEnc. Thus, the first inequality holds due to the security of $ri\mathcal{O}$.

We next show the second inequality. For $x_{i-1} \leq j-1$ and $x_{i-1} \geq j+1$, the behavior of $HE_{i-1}^{j,j}$ and $HE_{i-1}^{j,j+1}$ are the same. On input $x_{i-1} = j$, $HE_{i-1}^{j,k}$ always computes output ciphertexts by using Enc with MSK_i regardless of the value of k , and thus $HE_{i-1}^{j,j}$ and $HE_{i-1}^{j,j+1}$ behave in exactly the same way on input $x_{i-1} = j$. Thus, $HE_{i-1}^{j,j}$ and $HE_{i-1}^{j,j+1}$ are functionally equivalent and the second inequality holds due to the security of $ri\mathcal{O}$.

We finally show the third inequality. On input x_{i-1} , if $x_{i-1} = j$, PE_{i-1}^j in $\mathcal{H}_{1,j}^0$ outputs hardcoded $u_{i,0}$ and $u_{i,1}$ that are ciphertexts of j generated by using Enc with MSK_i . Otherwise, it generates ciphertexts of x_{i-1} using PEnc and $MSK_i^* \{j\}$, and outputs them. On input x_{i-1} , $HE_{i-1}^{j,2^{i-1}}$ in $\mathcal{H}_{1,j,2^{i-1}}$ computes output ciphertexts by using Enc with MSK_i if $x_{i-1} = j$, and by using PEnc with $MSK_i^* \{j\}$ otherwise. From this fact, we see that $HE_{i-1}^{j,2^{i-1}}$ in $\mathcal{H}_{1,j,2^{i-1}}$ is functionally equivalent to PE_{i-1}^j in $\mathcal{H}_{1,j}^0$. Thus, the third inequality holds due to the security of $ri\mathcal{O}$. \square (**Lemma 7.11**)

Proof of Lemma 7.12 We first show the first inequality. On input $\mathbf{x}_{i-1} \neq \mathbf{k}$, $\text{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}$ runs in exactly the same way as $\text{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\text{enc}}$. On input $\mathbf{x}_{i-1} = \mathbf{k}$, $\text{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}$ generate ciphertexts of k by using Enc with MSK_i , and outputs them. On input $\mathbf{x}_{i-1} = \mathbf{k}$, $\text{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\text{enc}}$ outputs hardwired $v_{k,0}$ and $v_{k,1}$ that are ciphertexts of k generated by using Enc with MSK_i . Thus, these circuits are functionally equivalent and the inequality holds due to the security of riO .

We next show the second inequality. In $\mathcal{H}_{1,j,k}^{\text{penc}}$, $\tilde{\text{E}}_{i-1}$ is generated by obfuscating $\text{HE}_{i-1}^{j,k}$ that has hardwired ciphertexts of k , that is $v_{k,0}$ and $v_{k,1}$ generated by using PEnc with $\text{MSK}_i^*\{j\}$. In $\mathcal{H}_{1,j,k+1}$, $\tilde{\text{E}}_{i-1}$ is generated by obfuscating $\text{HE}_{i-1}^{j,k+1}$ that does not have hardwired ciphertexts. On input $\mathbf{x}_{i-1} \neq \mathbf{k}$, these two circuits runs in exactly the same way. On input $\mathbf{x}_{i-1} = \mathbf{k}$, $\text{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\text{penc}}$ outputs hardwired $v_{k,0}$ and $v_{k,1}$. On input $\mathbf{x}_{i-1} = \mathbf{k}$, $\text{HE}_{i-1}^{j,k+1}$ in $\mathcal{H}_{1,j,k+1}$ generates ciphertexts of k by using PEnc with $\text{MSK}_i^*\{j\}$, and outputs them. Thus, two circuits are functionally equivalent and the second inequality holds due to the security of riO . \square (**Lemma 7.12**)

Finally, we estimate the term $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}})] \right|$ for every $k \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$. We can bound this term by using the indistinguishability of functionality under puncturing of pSKFE. Formally, we have the following lemma.

Lemma 7.13. *Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Let pSKFE satisfy δ -indistinguishability of functionality under puncturing. Let PPRF be δ -secure. Then, we have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}})] \right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ for every $j \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$.*

Proof of Lemma 7.13 Let k be any integer in $\{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$. In order to use the indistinguishability of functionality under puncturing of pSKFE, we change the randomness $r_{\text{Enc}}^{k\|b}$ used in $v_{k,b}$ into truly random for every $b \in \{0, 1\}$. Thus, we introduce the following intermediate hybrid experiments $\mathcal{H}_{1,j,k}^{\text{enc,rand}}$ and $\mathcal{H}_{1,j,k}^{\text{penc,rand}}$ between $\mathcal{H}_{1,j,k}^{\text{enc}}$ and $\mathcal{H}_{1,j,k}^{\text{penc}}$.

$\mathcal{H}_{1,j,k}^{\text{enc,rand}}$: Same as $\mathcal{H}_{1,j,k}^{\text{enc}}$ except that $r_{\text{Enc}}^{k\|b}$ is generated as a truly random string for every $b \in \{0, 1\}$.

By the pseudorandomness of PPRF, we have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc,rand}})] \right| \leq 2^{-\tilde{\lambda}^\epsilon}$.

$\mathcal{H}_{1,j,k}^{\text{penc,rand}}$: Same as $\mathcal{H}_{1,j,k}^{\text{penc}}$ except that $r_{\text{Enc}}^{k\|b}$ is generated as a truly random string for every $b \in \{0, 1\}$.

Note that the only difference between $\mathcal{H}_{1,j,k}^{\text{enc,rand}}$ and $\mathcal{H}_{1,j,k}^{\text{penc,rand}}$ is how $v_{k,b}$ is generated for every $b \in \{0, 1\}$. In $\mathcal{H}_{1,j,k}^{\text{enc,rand}}$, $v_{k,b}$ is generated by $v_{k,b} \leftarrow \text{Enc}(\text{MSK}_i, \mathbf{k}, (\mathbf{k}\|b, K_{i,\alpha}, \alpha))$. On the other hand, in $\mathcal{H}_{1,j,k}^{\text{penc,rand}}$, $v_{k,b}$ is generated by $v_{k,b} \leftarrow \text{PEnc}(\text{MSK}_i^*\{j\}, \mathbf{k}, (\mathbf{k}\|b, K_{i,\alpha}, \alpha))$. Now, in both experiments, the ciphertext $v_{k,b}$ is generated using a truly random string for every $b \in \{0, 1\}$.

Therefore, from the indistinguishability of functionality under puncturing of pSKFE, we obtain $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc,rand}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc,rand}})] \right| \leq 2^{-\tilde{\lambda}^\epsilon}$.

We have $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc,rand}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}})] \right| \leq 2^{-\tilde{\lambda}^\epsilon}$ by the pseudorandomness of PPRF.

From these, we obtain $\left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{enc}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\text{penc}})] \right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ for every $k \in \{0, \dots, 2^{i-1} - 1\} \setminus \{j\}$. \square (**Lemma 7.13**)

From inequality 11, and Lemma 7.11, 7.12, and 7.13, it holds that

$$\begin{aligned} \left| \Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)] \right| &\leq 3 \cdot \delta_{i-1} + 2(2^{i-1} - 1) \cdot \delta_{i-1} + (2^{i-1} - 1) \cdot 2^{-\Omega(\tilde{\lambda}^\epsilon)} \\ &\leq 2^{i-1} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) . \end{aligned} \quad (12)$$

This completes the estimation of $|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^0)]|$ for every $j \in \{0, \dots, 2^{i-1} - 1\}$, that is, completes the proof of Lemma 7.10. \square (Lemma 7.10)

From the symmetry of the difference of $\mathcal{H}_{1,j}$ and $\mathcal{H}_{1,j}^0$, and that of $\mathcal{H}_{1,j}^1$ and $\mathcal{H}_{1,j+1}$, by analyzing similarly, we obtain

$$|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j}^1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})]| \leq 2^{i-1} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \quad (13)$$

for every $j \in \{0, \dots, 2^{i-1} - 1\}$.

From inequality 12 and 13, inequality 10, and Lemma 7.8 and 7.9, we have

$$\begin{aligned} & |\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]| \\ & \leq 2 \cdot \delta_{i-1} + 2 \cdot 2^{i-1} \cdot 2^{i-1} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) + 2^{i-1} \cdot 2^{-\Omega(\tilde{\lambda}^\epsilon)} \\ & \leq 2^{2(i-1)} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) . \end{aligned} \quad (14)$$

From the symmetry of the difference of \mathcal{H}_1 and \mathcal{H}_2 , and that of \mathcal{H}_3 and \mathcal{H}_4 , by analyzing similarly, we also have

$$|\Pr[\mathcal{D}_i(\mathcal{H}_3)] - \Pr[\mathcal{D}_i(\mathcal{H}_4)]| \leq 2^{2(i-1)} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) . \quad (15)$$

This completes the estimation of $|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]|$ and $|\Pr[\mathcal{D}_i(\mathcal{H}_3)] - \Pr[\mathcal{D}_i(\mathcal{H}_4)]|$.

By combining inequality 14 and 15, Lemma 7.7, and inequality 9, we obtain inequality 8 in Lemma 7.6. This completes the security analysis for $i\mathcal{O}(1^\lambda, C) = ri\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ in Section 7.1.

7.3 Efficiency Analysis

In this section, we prove that $i\mathcal{O}(1^\lambda, C) = ri\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ runs in polynomial time. In this analysis, let s and n be the upper bound of the size and input length of circuits supported by $i\mathcal{O}$. When we invoke $ri\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ for some circuit C with n -bit input, $i\mathcal{O}$ generates total $n - 1$ master secret keys MSK_2, \dots, MSK_n . In other words, $i\mathcal{O}$ includes $n - 1$ instances of pSKFE. If all of these $n - 1$ instances runs in polynomial of $\tilde{\lambda}, n$ and s , then so does $i\mathcal{O}$. Below, we show it.

In order to accomplish the above task, we clearly distinguish each of these instances. Below, let pSKFE $_i$ denote the instance of pSKFE with respect to MSK_i . Especially, let Enc $_i$ denote the encryption circuit Enc corresponding to MSK_i , that is Enc(MSK_i, \cdot). Moreover, let PEnc $_i$ denote the punctured encryption circuit PEnc with respect to MSK_i^* , that is PEnc(MSK_i^*, \cdot), where MSK_i^* is the punctured master secret key generated by puncturing MSK_i . Note that which tag is punctured does not affect the running time of PEnc, and thus we omit to write the tag.

We start with the estimation of the size of padding pad_E^{i-1} of encryption circuits for every $i \in \{3, \dots, n\}$. Then, using the bound of pad_E^{i-1} , we complete the efficiency analysis of $i\mathcal{O}$.

In the above security analysis, in addition to the encryption circuits $E_{i-1, \alpha}$, we introduce encryption circuits PE_{i-1}^j , E_{i-1}^j , and $HE_{i-1}^{j,k}$, where $j, k \in \{0, \dots, 2^{i-1}\}$. We need to ensure that all of them have the same size, and thus we set

$$\text{pad}_E^{i-1} := \max(|E_{i-1, \alpha}|, |PE_{i-1}^j|, |E_{i-1}^j|, |HE_{i-1}^{j,k}|) .$$

All of above circuits evaluates a puncturable PRF over the domain $\{0, 1\}^n$, and then computes either Enc $_i$ or PEnc $_i$. Without loss of generality, we assume that Enc $_i$ and PEnc $_i$ have the same size for every

$i \in \{2, \dots, n\}$. This can be done by appropriate size padding. In this case, for every $i \in \{3, \dots, n\}$, we can bound pad_E^{i-1} as

$$\text{pad}_E^{i-1} \leq |\text{Enc}_i| \cdot \text{poly}_{\text{pad}}(\tilde{\lambda}, n) , \quad (16)$$

where poly_{pad} is a fixed polynomial.

We move on to the analysis of the running time of $i\mathcal{O}$. Especially, as stated earlier, we show that $|\text{Enc}_i|$ is polynomial of $\tilde{\lambda}, n$ and s for every $i \in \{2, \dots, n\}$.

First, for every $i \in \{2, \dots, n\}$, we specify the upper bound of the size s_i of circuits that pSKFE_i has to support.

Before analysis, we introduce some notations. For every $i \in \{2, \dots, n\}$ and circuit C_i with i -bit input, let $\text{Ev}_i[C_i]$ denote the evaluation circuit Ev_i defined in Figure 13 into which C_i is hardwired after encrypted by SKE. In addition, for every $i \in \{2, \dots, n\}$, let E_{i-1} denote the encryption circuit $E_{i-1,0}[\text{MSK}_i, K_i, S_i]$ defined in Figure 21.

Using this notation, we see that pSKFE_i has to support $\text{Ev}_i[E_i]$ for every $i \in \{2, \dots, n-1\}$. In addition, pSKFE_n has to support $\text{Ev}_n[C]$. Below, we bound the size of these circuits.

From inequality 16, for every $i \in \{3, \dots, n\}$, we have

$$|E_{i-1}| \leq |\text{Enc}_i| \cdot \text{poly}_{\text{pad}}(\tilde{\lambda}, n) . \quad (17)$$

In addition, we can analyze the size of $\text{Ev}_i[C_i]$ as follows. $\text{Ev}_i[C_i]$ includes a decryption procedure of encrypted C_i by SKE, and evaluation of a universal circuit $U(C_i, \mathbf{x}_i)$, where $\mathbf{x}_i \in \{0, 1\}^i$. Decryption procedure of SKE is done in linear time in $|C_i|$. In addition, we can perform the evaluation of a universal circuit in quasi-linear time in $|C_i|$ [Val76]. Thus, for every $i \in \{2, \dots, n\}$, we have

$$|\text{Ev}_i[C_i]| \leq |C_i| \cdot \log |C_i| \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) \leq |C_i|^{1+\gamma_1} \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) , \quad (18)$$

where poly_{Ev} is a fixed polynomial and $\gamma_1 < 1$ is an arbitrary constant. By combining inequalities 17 and 18, for every $i \in \{3, \dots, n\}$, we obtain

$$\begin{aligned} |\text{Ev}_{i-1}[E_{i-1}]| &\leq |E_{i-1}|^{1+\gamma_1} \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) \\ &\leq \left(|\text{Enc}_i| \cdot \text{poly}_{\text{pad}}(\tilde{\lambda}, n) \right)^{1+\gamma_1} \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) \\ &\leq |\text{Enc}_i|^{1+\gamma_1} \cdot \text{poly}_1(\tilde{\lambda}, n) , \end{aligned} \quad (19)$$

where poly_1 is some fixed polynomial.

Therefore, for every $i \in \{3, \dots, n\}$, we have

$$s_{i-1} \leq |\text{Enc}_i|^{1+\gamma_1} \cdot \text{poly}_1(\tilde{\lambda}, n) . \quad (20)$$

Moreover, from inequality 18, we have

$$s_n \leq |C|^{1+\gamma_1} \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) = s^{1+\gamma_1} \cdot \text{poly}_{\text{Ev}}(\tilde{\lambda}, n) . \quad (21)$$

Using the bound of s_i , we complete the efficiency analysis by estimating $|\text{Enc}_i|$ for every $i \in \{2, \dots, n\}$.

Recall that the building block pSKFE is weakly succinct, and for every $i \in \{2, \dots, n\}$, pSKFE_i encrypts $n + \tilde{\lambda} + 1$ bit plaintexts in the construction. Therefore, for every $i \in \{2, \dots, n\}$, we have

$$\begin{aligned} |\text{Enc}_i| &\leq s_i^\gamma \cdot \text{poly}(\tilde{\lambda}, n + \tilde{\lambda} + 1) \\ &\leq s_i^\gamma \cdot \text{poly}_E(\tilde{\lambda}, n) , \end{aligned} \quad (22)$$

where $\gamma < 1$ is some constant and poly_E denotes some fixed polynomial. By substituting inequality 20 into 22, for every $i \in \{3, \dots, n\}$, we obtain

$$\begin{aligned} |\text{Enc}_{i-1}| &\leq \left(|\text{Enc}_i|^{1+\gamma_1} \cdot \text{poly}_1(\tilde{\lambda}, n) \right)^\gamma \cdot \text{poly}_E(\tilde{\lambda}, n) \\ &\leq |\text{Enc}_i|^{\gamma_2} \cdot \text{poly}_2(\tilde{\lambda}, n) , \end{aligned} \quad (23)$$

where γ is a constant such that $(1 + \gamma_1)\gamma < \gamma_2 < 1$ and poly_2 is some fixed polynomial. Note that since we can choose γ_1 as an arbitrary small constant and $\gamma < 1$, by setting $\gamma_1 < \frac{1-\gamma}{\gamma}$, γ_2 satisfying the above condition exists. In addition, from inequality 21, we also have

$$\begin{aligned} |\text{Enc}_n| &\leq \left(s^{1+\gamma_1} \cdot \text{poly}_{E_V}(\tilde{\lambda}, n) \right)^\gamma \cdot \text{poly}_E(\tilde{\lambda}, n) \\ &\leq s^{\gamma_2} \cdot \text{poly}_3(\tilde{\lambda}, n) , \end{aligned} \quad (24)$$

where poly_3 is some fixed polynomial.

Then, from inequalities 23 and 24, for every $i \in \{2, \dots, n\}$, it holds that

$$\begin{aligned} |\text{Enc}_i| &\leq |\text{Enc}_n|^{\gamma_2^{n-i}} \cdot \prod_{j=1}^{n-i} \text{poly}_2(\tilde{\lambda}, n)^{\gamma_2^{j-1}} \\ &\leq \left(s^{\gamma_2} \cdot \text{poly}_3(\tilde{\lambda}, n) \right)^{\gamma_2^{n-i}} \cdot \prod_{j=1}^{n-i} \text{poly}_2(\tilde{\lambda}, n)^{\gamma_2^{j-1}} \\ &\leq s^{\gamma_2} \cdot \text{poly}_3(\tilde{\lambda}, n) \cdot \text{poly}_2(\tilde{\lambda}, n)^{\frac{1}{1-\gamma_2}} . \end{aligned}$$

The third inequality follows from the fact that $\gamma_2 < 1$. The above inequality means that the encryption algorithm of pSKFE_i runs in polynomial time of $\tilde{\lambda}, n$ and s for every $i \in \{2, \dots, n\}$.

In that case, all of the algorithms of pSKFE_i runs in polynomial of $\tilde{\lambda}, n$ and s for every $i \in \{2, \dots, n\}$. Thus, we conclude that $i\mathcal{O}$ runs in polynomial of $\tilde{\lambda}, n$ and s .

We completed the analysis of security and efficiency. Thus, we completed the proof of Theorem 7.1 and proved that we can construct IO for all circuits solely from SKFE for all circuits.

8 Basic Tools for Transformation from Succinctness to Collusion-Resistance

In this section, we introduce some basic constructions that we use to achieve a transformation from weakly succinct single-key SKFE to collusion-resistant SKFE.

8.1 Parallel Construction

For any q which is a polynomial of λ , we show how to construct an iSKFE scheme whose index space is $[q]$ based on a single-key SKFE scheme. The construction is very simple. The construction just runs q instances of the single-key scheme in parallel. Although this construction is essentially the same as the SUM construction by Li and Micciancio [LM16], we recall it to describe and analyze our collusion-resistant scheme in Sections 10.2 and 10.3. The construction is as follows.

Let $1\text{Key} = (1\text{Key.Setup}, 1\text{Key.KG}, 1\text{Key.Enc}, 1\text{Key.Dec})$ be a single-key SKFE scheme. We construct an iSKFE scheme $\text{Parallel}_q = (\text{Para}_q.\text{Setup}, \text{Para}_q.\text{iKG}, \text{Para}_q.\text{Enc}, \text{Para}_q.\text{Dec})$ as follows. Note again that q is a fixed polynomial of λ . Let the function space of 1Key be \mathcal{F} . The function space of Parallel_q is also \mathcal{F} .

Construction. The scheme consists of the following algorithms.

$\text{Para}_q.\text{Setup}(1^\lambda)$:

- For every $k \in [q]$, generate $\text{MSK}_k \leftarrow \text{1Key.Setup}(1^\lambda)$.
- Return $\text{MSK} := \{\text{MSK}_k\}_{k \in [q]}$.

$\text{Para}_q.\text{iKG}(\text{MSK}, f, i)$:

- Parse $\{\text{MSK}_k\}_{k \in [q]} \leftarrow \text{MSK}$.
- Compute $\text{1Key.sk}_f \leftarrow \text{1Key.KG}(\text{MSK}_i, f)$.
- Return $\text{sk}_f := (i, \text{1Key.sk}_f)$.

$\text{Para}_q.\text{Enc}(\text{MSK}, m)$:

- Parse $\{\text{MSK}_k\}_{k \in [q]} \leftarrow \text{MSK}$.
- For every $k \in [q]$, compute $\text{CT}_k \leftarrow \text{1Key.Enc}(\text{MSK}_k, m)$.
- Return $\text{CT} := \{\text{CT}_k\}_{k \in [q]}$.

$\text{Para}_q.\text{Dec}(\text{sk}_f, \text{CT})$:

- Parse $(i, \text{1Key.sk}_f) \leftarrow \text{sk}_f$ and $\{\text{CT}_k\}_{k \in [q]} \leftarrow \text{CT}$.
- Return $y \leftarrow \text{1Key.Dec}(\text{1Key.sk}_f, \text{CT}_i)$.

The correctness of this construction directly follows from that of 1Key .

Efficiency. Let $\text{1Key.t}_{\text{Enc}}$ and $\text{1Key.t}_{\text{KG}}$ be bounds of the running time of 1Key.Enc and 1Key.KG . In addition, let $\text{Para}_q.t_{\text{Enc}}$ and $\text{Para}_q.t_{\text{iKG}}$ be bounds of the running time of $\text{Para}_q.\text{Enc}$ and $\text{Para}_q.\text{iKG}$. See Definition 3.20 for s and n . Then, we have

$$\begin{aligned} \text{Para}_q.t_{\text{Enc}}(\lambda, n, s) &= q \cdot \text{1Key.t}_{\text{Enc}}(\lambda, n, s) , \\ \text{Para}_q.t_{\text{iKG}}(\lambda, n, s) &= \text{1Key.t}_{\text{iKG}}(\lambda, n, s). \end{aligned}$$

Especially, if 1Key is (weakly) succinct and we set $q := \lambda$, then Parallel_q is also (weakly) succinct and we have

$$\text{Para}_q.t_{\text{Enc}}(\lambda, n, s) = \lambda \cdot \text{1Key.t}_{\text{Enc}}(\lambda, n, s) = s^\gamma \cdot \text{poly}_{\text{Para}}(\lambda, n), \quad (25)$$

where $\gamma < 1$ is a constant and $\text{poly}_{\text{Para}}$ is a fixed polynomial. Note that the encryption algorithm of Parallel_λ runs that of 1Key λ times. Therefore, $\text{Para}.t_{\text{Enc}}$ is λ times bigger than the encryption time of 1Key , but the factor λ is absorbed in $\text{poly}_{\text{Para}}(\lambda, n)$, and thus we can bound $\text{Para}.t_{\text{Enc}}^{(k)}$ by inequality (25). At the concrete instantiation in Section 10, we set $q := \lambda$ and use this bound.

Security. We have the following theorem.

Theorem 8.1 ([LM16, Adapted]). *Let 1Key be $(1, \delta)$ -selective-message function private SKFE. Then, Parallel_q is (q, δ) -selective-message function private iSKFE.*

We omit the proof of Theorem 8.1 since the proof is essentially the same as the proof for the SUM construction by Li and Micciancio [LM16]. See the unpublished manuscript by Kitagawa et al. [KNT17a] for the full proof.

8.2 Single-Ciphertext Collusion-Resistant Fully Succinct SKFE

Next, we show how to construct a succinct SKFE scheme 1CT based solely on one-way functions. The scheme is single-ciphertext collusion-resistant, that is, it is secure against adversaries who make only one encryption query and unbounded many key queries. In addition, the length of a master secret key of 1CT is λ bit, regardless of the length of a message to be encrypted. The construction uses a garbling scheme, an SKE scheme, and a PRF all of which can be constructed from one-way functions. The construction can be essentially seen as a flipped variant of the construction proposed by Sahai and Seyalioglu [SS10].

Recall that n is the size of the message. So, we let $n := |m|$ where m is a message. Let $\text{GC} = (\text{Grbl}, \text{Eval})$ be a garbling scheme, $\text{SKE} = (\text{E}, \text{D})$ an SKE scheme, and $\{F_S : \{0, \dots, n\} \times \{0, 1\} \rightarrow \{0, 1\}^n \mid S \in \{0, 1\}^\lambda\}$ a PRF. Using GC , SKE , and F , we construct an SKFE scheme $1\text{CT} = (1\text{CT}.\text{Setup}, 1\text{CT}.\text{KG}, 1\text{CT}.\text{Enc}, 1\text{CT}.\text{Dec})$ as follows.

Construction. The scheme consists of the following algorithms.

$1\text{CT}.\text{Setup}(1^\lambda)$:

- Generate $S \xleftarrow{r} \{0, 1\}^\lambda$ and return $\text{MSK} := S$.

$1\text{CT}.\text{KG}(\text{MSK}, f)$:

- Parse $S \leftarrow \text{MSK}$.
- Compute $K_{j,\alpha} \leftarrow F_S(j\|\alpha)$ for every $j \in [n]$ and $\alpha \in \{0, 1\}$, and $R \leftarrow F_S(0\|0)$ ¹⁵.
- Compute $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C_f)$, where C_f is a circuit computing f .
- For every $j \in [n]$, compute $c_{j,0} \leftarrow \text{E}(K_{j,0}, L_{j,R[j]})$ and $c_{j,1} \leftarrow \text{E}(K_{j,1}, L_{j,1-R[j]})$.
- Return $sk_f := (\tilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}})$.

$1\text{CT}.\text{Enc}(\text{MSK}, m)$:

- Parse $S \leftarrow \text{MSK}$.
- Compute $K_{j,\alpha} \leftarrow F_S(j\|\alpha)$ for every $j \in [n]$ and $\alpha \in \{0, 1\}$.
- Compute $R \leftarrow F_S(0\|0)$ and $x := m \oplus R$.
- Return $\text{CT} := (x, \{K_{j,x[j]}\}_{j \in [n]})$.

$1\text{CT}.\text{Dec}(sk_f, \text{CT})$:

- Parse $(\tilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow sk_f$ and $(x, \{K_j\}_{j \in [n]}) \leftarrow \text{CT}$.
- For every $j \in [n]$, compute $L_j \leftarrow \text{D}(K_j, c_{j,x[j]})$.
- Return $y \leftarrow \text{Eval}(\tilde{C}, \{L_j\}_{j \in [n]})$.

Correctness. If $R[j] = 0$, the SKE ciphertext $c_{j,\alpha}$ is an encryption of $L_{j,\alpha}$, and $x[j] = m[j]$ holds for every $j \in [n]$ and $\alpha \in \{0, 1\}$. If $R[j] = 1$, the SKE ciphertext $c_{j,\alpha}$ is an encryption of $L_{j,1-\alpha}$, and $x[j] = 1 - m[j]$ holds for every $j \in [n]$ and $\alpha \in \{0, 1\}$. Then, We see that for every $j \in [n]$, $c_{j,x[j]}$ is an encryption of $L_{j,m[j]}$. Therefore, the correctness of 1CT directly follows from those of building blocks.

¹⁵ We assume that $n \geq \lambda$ and $K_{j,\alpha}$ is the first λ bit of $F_S(j\|\alpha)$ for every $j \in [n]$ and $\alpha \in \{0, 1\}$.

Efficiency. We use Yao's garbled circuit for GC [Yao86, BHR12]. We observe that the running time of Grbl, $\text{GC}.t_{\text{Grbl}}(\lambda, |f|) = |f| \cdot \text{poly}(\lambda)$ (linear in $|f|$) from Yao's construction. Other computations included in the description of 1CT is just computing XOR, PRF over domain $[2n]$, and encryption of SKE. Let $\text{1CT}.t_{\text{Enc}}$, $\text{1CT}.\ell^{\text{Enc}}$, and $\text{1CT}.t_{\text{KG}}$ be bounds of the running time and output length of $\text{1CT}. \text{Enc}$ and the running time of $\text{1CT}. \text{KG}$. In addition, t_{F} , t_{Grbl} , and t_{E} are bounds of the running time of F, Grbl, and E, respectively. Then, we have

$$\begin{aligned} |\text{MSK}| &= \lambda, \quad \text{1CT}.\ell^{\text{Enc}}(\lambda, n) = (\lambda + 1)n, \\ \text{1CT}.t_{\text{Enc}}(\lambda, n) &= n \cdot \text{poly}_{\text{1CT}}(\lambda, \log n), \\ \text{1CT}.t_{\text{KG}}(\lambda, n, s) &= t_{\text{F}} + t_{\text{Grbl}}(\lambda, |f|) + 2 \cdot n \cdot t_{\text{E}} \\ &\leq \text{poly}(\lambda, \log n) + |f| \cdot \text{poly}(\lambda) + n \cdot \text{poly}(\lambda) \\ &\leq (|f| + n) \cdot \text{poly}(\lambda), \end{aligned} \tag{26}$$

$$\tag{27}$$

where poly_{1CT} is a fixed polynomial.

The master secret-key length is $|\text{MSK}| = \lambda$ thus is independent of n . The encryption time is independent of the size of f , i.e., this scheme is fully succinct. Moreover, we stress that the running time of the key generation is linear in $|f|$.

Security. We have the following theorem.

Theorem 8.2. *Let GC be δ -secure garbling scheme, SKE δ -secure SKE, and F δ -secure PRF. Then, 1CT is single-ciphertext (poly, δ) -selective-message function private SKFE.*

Proof of Theorem 8.2. We assume that the advantage of any adversary attacking GC, SKE, and F is bounded by ϵ_{GC} , ϵ_{SKE} , and ϵ_{PRF} , respectively. Let \mathcal{A} be an adversary that attacks the selective-message function privacy of 1CT. Moreover, we assume that \mathcal{A} makes q key queries at most, where q is a polynomial of λ . Then, it holds that

$$\text{Adv}_{\text{1CT}, \mathcal{A}}^{\text{sm-fp}}(\lambda) \leq 2(q \cdot \epsilon_{\text{GC}} + \epsilon_{\text{SKE}} + \epsilon_{\text{PRF}}). \tag{28}$$

This means that if all of GC, SKE, and F are δ -secure, then so is 1CT. Below, we prove this via a sequence of games. First, consider the following sequence of games.

Game 0 This is the selective-message function privacy game regarding 1CT.

Initialization First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \mathcal{A} sends (m_0, m_1) to the challenger. Next, the challenger generates $S \xleftarrow{r} \{0, 1\}^\lambda$ and chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Then, the challenger computes $K_{j,\alpha} \leftarrow F_S(j||\alpha)$ for every $j \in [n]$ and $\alpha \in \{0, 1\}$, $R \leftarrow F_S(0||0)$, and $x := m_b \oplus R$. Finally, the challenger returns $(x, \{K_{j,x[j]}\}_{j \in [n]})$ to \mathcal{A} .

key queries For a key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ from \mathcal{A} , the challenger first computes $K_{j,\alpha} \leftarrow F_S(j||\alpha)$ for every $j \in [n]$ and $\alpha \in \{0, 1\}$, and $R \leftarrow F_S(0||0)$. Then, the challenger computes $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C_{f_b})$, where C_{f_b} is a circuit computing f_b . Finally, the challenger compute $c_{j,0} \leftarrow E(K_{j,0}, L_{j,R[j]})$ and $c_{j,1} \leftarrow E(K_{j,1}, L_{j,1-R[j]})$ for every $j \in [n]$, and returns $(\tilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}})$ to \mathcal{A} .

Final phase \mathcal{A} outputs $b' \in \{0, 1\}$.

Game 1 Same as Game 0 except that the challenger generates $\{K_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}$ and R as truly random strings.

Game 2 Same as Game 1 except that for every $j \in [n]$, the challenger generates $c_{j,1-x[j]} \leftarrow \mathbf{E}(K_{j,1-x[j]}, 0^\lambda)$.

Game 3 Same as Game 2 except that when \mathcal{A} makes a key query (f_0, f_1) , the challenger computes $(\tilde{C}, \{L_j\}_{j \in [n]}) \leftarrow \text{Sim}(1^\lambda, s, y)$, where $s = |f_0| = |f_1|$ and $y = f_0(m_0) = f_1(m_1)$. Here, Sim is a simulator for GC. In addition, the challenger computes $c_{j,x[j]} \leftarrow \mathbf{E}(K_{j,x[j]}, L_j)$ for every $j \in [n]$.

Game 4 Same as Game 3 except that the challenger generates $x \xleftarrow{r} \{0, 1\}^n$.

For $h = 0, \dots, 4$, let SUC_i be the event that \mathcal{A} succeeds in guessing the challenge bit, that is, $b = b'$ occurs in Game i . In Game 4, the challenge bit b is information theoretically hidden from the view of \mathcal{A} thus $|\Pr[\text{SUC}_4] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\frac{1}{2} \cdot \text{Adv}_{\text{ICT}, \mathcal{A}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_0] - \frac{1}{2}| \leq \sum_{h=0}^3 |\Pr[\text{SUC}_h] - \Pr[\text{SUC}_{h+1}]|. \quad (29)$$

Below, we estimate that each term on the right side of inequality (29) is negligible.

Lemma 8.3. $|\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| \leq \epsilon_{\text{PRF}}$.

The proof is straightforward thus omitted.

Lemma 8.4. $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \leq \epsilon_{\text{SKE}}$.

Proof of Lemma 8.4. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks SKE.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} generates $R \xleftarrow{r} \{0, 1\}^\lambda$, $b \xleftarrow{r} \{0, 1\}$. When, \mathcal{A} sends (m_0, m_1) , \mathcal{B} computes $x := m_b \oplus R$, and generates $K_{j,x[j]} \xleftarrow{r} \{0, 1\}^\lambda$ for every $j \in [n]$. Finally, \mathcal{B} sends $\text{CT} := \{K_{j,x[j]}\}_{j \in [n]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$, \mathcal{B} first computes $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C_{f_b})$. Then, for every $j \in [n]$, \mathcal{B} makes an encryption query $(j, L_{j,1-x[j]}, 0^\lambda)$, and obtains the answer $c_{j,1-x[j]}$. Next, for every $j \in [n]$, \mathcal{B} computes $c_{j,x[j]} \leftarrow \mathbf{E}(K_{j,x[j]}, L_{j,x[j]})$. Finally, \mathcal{B} returns $sk_f := (\tilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in [n]})$ to \mathcal{A} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs $\beta' = 1$ if $b = b'$. Otherwise, \mathcal{B} outputs $\beta' = 0$.

Let β be the challenge bit between the challenger and \mathcal{B} . Then, the advantage of \mathcal{B} is estimated as $\text{Adv}_{\text{SKE}, \mathcal{B}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 1] - \Pr[\beta' = 1 \mid \beta = 0]|$. When $\beta = 0$, \mathcal{B} perfectly simulates Game 0 for \mathcal{A} . On the other hand, when $\beta = 1$, \mathcal{B} perfectly simulates Game 1 for \mathcal{A} . In addition, \mathcal{B} outputs 1 if and only if $b = b'$ occurs. Therefore, we have $\text{Adv}_{\text{SKE}, n, \mathcal{B}}^{\text{cpa}}(\lambda) = |\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]|$, and thus $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \leq \epsilon_{\text{SKE}}$ holds. \square (**Lemma 8.4**)

Lemma 8.5. $|\Pr[\text{SUC}_2] - \Pr[\text{SUC}_3]| \leq q \cdot \epsilon_{\text{GC}}$.

Proof of Lemma 8.5. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks GC.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} generates $R \xleftarrow{r} \{0, 1\}^\lambda$, $k^* \xleftarrow{r} [Q]$, and $b \xleftarrow{r} \{0, 1\}$. When, \mathcal{A} sends (m_0, m_1) , \mathcal{B} computes $x := m_b \oplus R$, and generates $K_{j,x[j]} \xleftarrow{r} \{0, 1\}^\lambda$ for every $j \in [n]$. Finally, \mathcal{B} sends $\text{CT} \leftarrow (x, \{K_{j,x[j]}\}_{j \in [n]})$ to \mathcal{A} .

Key queries For the k -th key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ made by \mathcal{A} , \mathcal{B} first computes responds as follows.

- If $i < k^*$, \mathcal{B} first computes $(\tilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \text{Grbl}(1^\lambda, C_{f_b})$, where C_{f_b} is a circuit computing f .
- If $i = k^*$, \mathcal{B} first sends C_{f_b} and m_b to the challenger, and obtains the answer $(\tilde{C}, \{L_j\}_{j \in [n]})$.
- If $i > k^*$, \mathcal{B} first computes $(\tilde{C}, \{L_j\}_{j \in [n]}) \leftarrow \text{Sim}(1^\lambda, s, y)$, where $s = |C_{f_0}| = |C_{f_1}|$ and $y = f_0(m_0) = f_1(m_1)$.

Then, for every $j \in [n]$, \mathcal{B} computes $c_{j,x[j]} \leftarrow \text{E}(K_{j,x[j]}, L_{j,x[j]})$ and $c_{j,1-x[j]} \leftarrow \text{E}(K_{j,1-x[j]}, 0^\lambda)$. Finally, \mathcal{B} returns $sk_f := (\tilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}})$ to \mathcal{A} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs $\beta' = 1$ if $b = b'$. Otherwise, \mathcal{B} outputs $\beta' = 0$.

Let β be the challenge bit between the challenger and \mathcal{B} . Then, we have

$$\begin{aligned} \text{Adv}_{\text{GC,Sim},\mathcal{B}}^{\text{GC}}(\lambda) &= |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]| \\ &= \left| \sum_{Q=1}^q \Pr[\beta' = 1 \wedge k^* = Q \mid \beta = 0] - \sum_{Q=1}^q \Pr[\beta' = 1 \wedge k^* = Q \mid \beta = 1] \right| \\ &= \frac{1}{q} \left| \sum_{Q=1}^q \Pr[b' = 1 \mid k^* = Q \wedge \beta = 0] - \sum_{Q=1}^q \Pr[b' = 1 \mid k^* = Q \wedge \beta = 1] \right|. \end{aligned}$$

We note that for every $Q \in [q-1]$, the view of \mathcal{A} when $k^* = Q$ and $\beta = 0$ is exactly the same as that of when $k^* = Q+1$ and $\beta = 1$. Thus, we have $\Pr[b' = 1 \mid k^* = Q \wedge \beta = 0] = \Pr[b' = 1 \mid k^* = Q+1 \wedge \beta = 1]$ for every $Q \in [q-1]$. Therefore, we also have

$$\text{Adv}_{\text{GC,Sim},\mathcal{B}}^{\text{GC}}(\lambda) = \frac{1}{q} |\Pr[b' = 1 \mid k^* = 1 \wedge \beta = 1] - \Pr[b' = 1 \mid k^* = q \wedge \beta = 0]|.$$

When $k^* = 1$ and $\beta = 1$, \mathcal{B} perfectly simulates Game 2 for \mathcal{A} . On the other hand, when $k^* = q$ and $\beta = 0$, \mathcal{B} perfectly simulates Game 3 for \mathcal{A} . Therefore, we have $|\Pr[\text{SUC}_2] - \Pr[\text{SUC}_3]| = |\Pr[b' = 1 \mid k^* = 1 \wedge \beta = 1] - \Pr[b' = 1 \mid k^* = q \wedge \beta = 0]|$, and thus we obtain $|\Pr[\text{SUC}_2] - \Pr[\text{SUC}_3]| = q \cdot \text{Adv}_{\text{GC,Sim},\mathcal{B}}^{\text{GC}}(\lambda) \leq q \cdot \epsilon_{\text{GC}}$. \square (**Lemma 8.5**)

Lemma 8.6. $|\Pr[\text{SUC}_3] - \Pr[\text{SUC}_4]| = 0$.

Proof of Lemma 8.6. The difference between Game 3 and 4 is how the challenger generates x . However, x is uniformly distributed in both of games thus $|\Pr[\text{SUC}_3] - \Pr[\text{SUC}_4]| = 0$. \square (**Lemma 8.6**)

From inequality (29) and Lemmas 8.4 to 8.6, we see that inequality (28) holds. \square (**Theorem 8.2**)

8.3 Hybrid Encryption Construction

We next introduce a construction based on the hybrid encryption methodology. The construction combines an iSKFE scheme and 1CT we constructed in Section 8.2, and leads to a new iSKFE scheme. The construction is similar to the SKFE variant of the construction proposed by Ananth et al. [ABSV15] except the following. First, our construction works even if one of the building block is an iSKFE scheme. Second, in our construction, if both building block schemes satisfy function privacy, then so does the resulting scheme.

Let iSKFE = (Setup, iKG, Enc, Dec) be an iSKFE scheme whose index space is \mathcal{I} . Let 1CT = (1CT.Setup, 1CT.KG, 1CT.Enc, 1CT.Dec) be an SKFE scheme. Let $\{F_S : \mathcal{I} \rightarrow \mathcal{R} \mid S \in \{0,1\}^\lambda\}$ be a PRF, where \mathcal{R} is the randomness space of 1CT.KG. We construct an iSKFE scheme HYBRD = (Hyb.Setup, Hyb.iKG, Hyb.Enc, Hyb.Dec) as follows. The index space of HYBRD is the same as

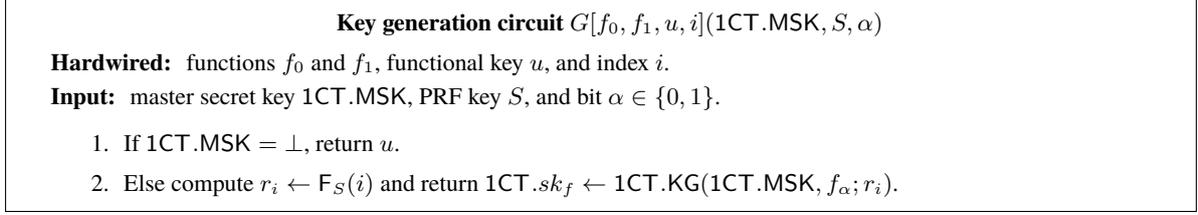


Figure 20: Construction of a key generation circuit G .

iSKFE, that is, \mathcal{I} . Moreover, if the function space of 1CT is \mathcal{F} , then that of HYBRD is also \mathcal{F} . We assume that iSKFE supports a sufficiently large function class that particularly includes the key generation circuit G described in Figure 20.

The features of HYBRD are as follows.

- HYBRD preserves q -bounded collusion-resistance, weak-succinctness, and many-ciphertext of iSKFE.
- Even if the message m is large, the encryption time of HYBRD does not increase too much since m is encrypted by 1CT, which is fully succinct. iSKFE encrypts the master secret key of 1CT, whose length depends only on the security parameter.

Construction. The scheme consists of the following algorithms.

Hyb.Setup(1^λ) :

- Return $MSK \leftarrow \text{Setup}(1^\lambda)$.

Hyb.iKG(MSK, f, i) :

- Compute $sk_G \leftarrow \text{iKG}(MSK, G[f, \perp, \perp, i], i)$. The circuit G is defined in Figure 20.
- Return $sk_f := sk_G$.

Hyb.Enc(MSK, m) :

- Generate $1CT.MSK \leftarrow 1CT.Setup(1^\lambda)$ and $S \xleftarrow{r} \{0, 1\}^\lambda$.
- Compute $CT \leftarrow \text{Enc}(MSK, (1CT.MSK, S, 0))$ and $1CT.CT \leftarrow 1CT.Enc(1CT.MSK, m)$.
- Return $\text{Hyb.CT} := (CT, 1CT.CT)$.

Hyb.Dec($sk_f, \text{Hyb.CT}$) :

- Parse $sk_G \leftarrow sk_f$ and $(CT, 1CT.CT) \leftarrow \text{Hyb.CT}$.
- Compute $1CT.sk_f \leftarrow \text{Dec}(sk_G, CT)$.
- Return $y \leftarrow 1CT.Dec(1CT.sk_f, 1CT.CT)$.

The correctness of HYBRD follows from those of building blocks.

Efficiency. We estimate the size of G since we need it in the efficiency analysis of our main construction in Section 10.

Let $|1CT.KG|$ and $|F|$ denote the size of the circuits computing $1CT.KG$ and F , respectively. Then, the size of G is

$$\begin{aligned} |G| &= 2|f| + |1CT.sk_f| + |i| + |1CT.KG| + |F| \\ &\leq 2|f| + (|f| + |m|) \cdot \text{poly}(\lambda) + \log q + (|f| + |m|) \cdot \text{poly}(\lambda) + \text{poly}(\lambda, \log q) \\ &\leq (|f| + |m|) \cdot \text{poly}_G(\lambda, \log q) \quad , \end{aligned} \tag{30}$$

where poly_G is some fixed polynomial. The first inequality holds due to the efficiency of $1CT$ in Section 8.2. More specifically, the running time of $1CT.KG$ (Equation (27)) yields the first inequality. Note that $1CT.sk_f$ is output by $1CT.KG$ thus $|1CT.sk_f|$ is bounded by $|1CT.KG|$.

Security. We have the following theorem.

Theorem 8.7. *Let $iSKFE$ be (q, δ) -selective-message function private $iSKFE$, where q is a fixed function of λ . Let $1CT$ be single-ciphertext (poly, δ) -selective-message function private $SKFE$. Let F be a δ -secure PRF. Then, $HYBRD$ is (q, δ) -selective-message function private $iSKFE$.*

Note that the above theorem holds even if q is not polynomial of λ . In fact, in the concrete instantiation in Section 10, we set q as a super-polynomial of λ . To prove Theorem 8.7, we use the following lemma.

Lemma 8.8. *We assume that the advantage of any adversary attacking $iSKFE$, $1CT$, and PRF is bounded by ϵ_{iSKFE} , ϵ_{1CT} , and ϵ_{PRF} , respectively. Let \mathcal{A} be an adversary that attacks the selective-message function privacy of $HYBRD$. We assume that \mathcal{A} sends p message pairs at most to the challenger at the initialization step. Then, it holds that*

$$\text{Adv}_{HYBRD, \mathcal{A}}^{\text{sm-fp}}(\lambda) \leq 2((2p + 1) \cdot \epsilon_{iSKFE} + p \cdot \epsilon_{1CT} + p \cdot \epsilon_{PRF}) .$$

This lemma immediately implies Theorem 8.7 since if all of $iSKFE$, $1CT$, and F are δ -secure, then so is $HYBRD$. Below, we prove Lemma 8.8.

Proof of Lemma 8.8. We define a sequence of games.

Game 0 This is the selective-message function privacy game regarding $HYBRD$.

Initialization First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [p]}$ to the challenger. Next, the challenger generates $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$ and chooses a challenge bit $b \xleftarrow{r} \{0, 1\}$. Then, for every $\ell \in [p]$, the challenger generates $1CT.\text{MSK}^{(\ell)} \leftarrow 1CT.\text{Setup}(1^\lambda)$, $S^{(\ell)} \xleftarrow{r} \{0, 1\}^\lambda$, and $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1CT.\text{MSK}^{(\ell)}, S^{(\ell)}, 0))$. Finally, the challenger generates $1CT.\text{CT}^{(\ell)} \leftarrow 1CT.\text{Enc}(1CT.\text{MSK}^{(\ell)}, m_b^\ell)$ for every $\ell \in [p]$, and returns $\{(\text{CT}^{(\ell)}, 1CT.\text{CT}^{(\ell)})\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, the challenger returns $sk_G^{(i)} \leftarrow iKG(\text{MSK}, G[f_b^i, \perp, \perp, i], i)$ to \mathcal{A} .

Final phase \mathcal{A} output b' .

For every $\ell^* \in [p]$, we define the following games. The goal is changing the ciphertext of $m_0^{\ell^*}$ into that of $m_1^{\ell^*}$. The issue is that we need to erase $1CT.\text{MSK}^{(\ell^*)}$ from $\text{CT}^{(\ell^*)}$ to use the security of $1CT$. To do so, we directly embed $1CT.sk_f \leftarrow 1CT.KG(1CT.\text{MSK}, f)$ into function G , and use the function hiding property of $iSKFE$ to hide $1CT.sk_f$.

Game (5, 0) We define Game (5, 0) as the same game as Game 0.

Game (1, ℓ^*) Same as Game (5, $\ell^* - 1$) except the following. The challenger generates $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT}.\text{MSK}^{(\ell)}, S^{(\ell)}, 1))$ for every $\ell \in [\ell^* - 1]$, and $\text{CT}^{(\ell^*)} \leftarrow \text{Enc}(\text{MSK}, (\perp, \perp, 0))$.

In addition, when \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, the challenger computes $r_i^{(\ell^*)} \leftarrow F_{S^{(\ell^*)}}(i)$ and $u_i \leftarrow 1\text{CT}.\text{KG}(1\text{CT}.\text{MSK}^{(\ell^*)}, f_b^i, r_i^{(\ell^*)})$, and returns $sk_G^{(i)} \leftarrow \text{iKG}(\text{MSK}, G[f_b^i, f_1^i, u_i, i], i)$.

Game (2, ℓ^*) Same as Game (1, ℓ^*) except that the challenger generates $r_i^{(\ell^*)}$ as a truly random string when \mathcal{A} makes a key query (i, f_0^i, f_1^i) .

Game (3, ℓ^*) Same as Game (2, ℓ^*) except the following. The challenger generates $1\text{CT}.\text{CT}^{(\ell^*)} \leftarrow 1\text{CT}.\text{Enc}(1\text{CT}.\text{MSK}^{(\ell^*)}, m_1^{\ell^*})$. In addition, the challenger generates $u_i \leftarrow 1\text{CT}.\text{KG}(1\text{CT}.\text{MSK}^{(\ell^*)}, f_1^i, r_i^{(\ell^*)})$ when \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$.

Game (4, ℓ^*) Same as Game (3, ℓ^*) except that the challenger generates $r_i^{(\ell^*)} \leftarrow F_{S^{(\ell^*)}}(i)$ when \mathcal{A} makes a key query (i, f_0^i, f_1^i) .

Game (5, ℓ^*) Same as Game (4, ℓ^*) except the following. The challenger generates $\text{CT}^{(\ell^*)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT}.\text{MSK}^{(\ell^*)}, S^{(\ell^*)}, 1))$. In addition, when \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, the challenger responds with $sk_G^{(i)} \leftarrow \text{iKG}(\text{MSK}, G[f_b^i, f_1^i, \perp, i], i)$.

We define one additional game.

Game 6 Same as Game (5, p) except that when \mathcal{A} makes a key query (i, f_0^i, f_1^i) , the challenger generates $sk_G^{(i)} \leftarrow \text{iKG}(\text{MSK}, G[\perp, f_1^i, \perp, i], i)$. In this game, the challenger generates $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT}.\text{MSK}^{(\ell)}, S^{(\ell)}, 1))$ for every $\ell \in [p]$.

Let SUC_0 and SUC_6 be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game 0 and 6, respectively. Similarly, for every $h \in \{1, \dots, 5\}$ and $\ell^* \in [p]$, let $\text{SUC}_{(h, \ell^*)}$ be the event that \mathcal{A} succeeds in guessing b in Game (h, ℓ^*) . In Game 6, the challenge bit b is information theoretically hidden from the view of \mathcal{A} , and thus $|\Pr[\text{SUC}_6] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned} \frac{1}{2} \cdot \text{Adv}_{\text{HYBRD}, \mathcal{A}}^{\text{sm-fp}}(\lambda) &= |\Pr[\text{SUC}_0] - \frac{1}{2}| \\ &\leq \sum_{\ell^* \in [p]} |\Pr[\text{SUC}_{(5, \ell^* - 1)}] - \Pr[\text{SUC}_{(1, \ell^*)}]| \\ &\quad + \sum_{\ell^* \in [p]} \sum_{h=1}^4 |\Pr[\text{SUC}_{(h, \ell^*)}] - \Pr[\text{SUC}_{(h+1, \ell^*)}]| \\ &\quad + |\Pr[\text{SUC}_{(5, p)}] - \Pr[\text{SUC}_6]| \end{aligned} \tag{31}$$

Below, we estimate each term on the right side of inequality (31).

Lemma 8.9. For every $\ell^* \in [p]$, $|\Pr[\text{SUC}_{(5, \ell^* - 1)}] - \Pr[\text{SUC}_{(1, \ell^*)}]| \leq \epsilon_{\text{ISKFE}}$.

Proof of Lemma 8.9. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks iSKFE.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} chooses $b \leftarrow \{0, 1\}$. When, \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, \mathcal{B} sets $\{(M_0^\ell, M_1^\ell)\}_{\ell \in [p]}$ as follows.

- For every $\ell < \ell^*$, \mathcal{B} sets $M_0^\ell = M_1^\ell = (1\text{CT.MSK}^{(\ell)}, S^{(\ell)}, 1)$.
- \mathcal{B} sets $M_0^{\ell^*} = (1\text{CT.MSK}^{(\ell^*)}, S^{(\ell^*)}, 0)$ and $M_1^{\ell^*} = (\perp, \perp, 0)$.
- For every $\ell > \ell^*$, \mathcal{B} sets $M_0^\ell = M_1^\ell = (1\text{CT.MSK}^{(\ell)}, S^{(\ell)}, 0)$.

Then, \mathcal{B} sends $\{(M_0^\ell, M_1^\ell)\}_{\ell \in [p]}$ to the challenger and gets the answer $\{\text{CT}^{(\ell)}\}_{\ell \in [p]}$. Next, \mathcal{B} computes $1\text{CT.CT}^{(\ell)} \leftarrow 1\text{CT.Enc}(1\text{CT.MSK}^{(\ell)}, m_1^\ell)$ for every $\ell < \ell^*$, and $1\text{CT.CT}^{(\ell)} \leftarrow 1\text{CT.Enc}(1\text{CT.MSK}^{(\ell)}, m_b^\ell)$ for every $\ell \geq \ell^*$. Finally, \mathcal{B} sets $\text{Hyb.CT}^{(\ell)} \leftarrow (\text{CT}^{(\ell)}, 1\text{CT.CT}^{(\ell)})$ for every $\ell \in [p]$, and sends $\{\text{Hyb.CT}^{(\ell)}\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, \mathcal{B} first computes $u_i \leftarrow 1\text{CT.KG}(\text{MSK}^{(\ell^*)}, f_b^i; r_i^{(\ell^*)})$, where $r_i^{(\ell^*)} \leftarrow F_{S^{(\ell^*)}}(i)$. Then, \mathcal{B} queries $(i, G[f_b^i, f_1^i, \perp, i], G[f_b^i, f_1^i, u_i, i])$ to the challenger and returns the answer to \mathcal{A} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs 1 if $b = b'$. Otherwise, \mathcal{B} outputs 0.

Let β be the challenge bit between the challenger and \mathcal{B} . For every $\ell \neq \ell^*$ and key query (i, f_0^i, f_1^i) made by \mathcal{A} , $G[f_b^i, f_1^i, \perp, i](M_0^\ell) = G[f_b^i, f_1^i, u_i, i](M_1^\ell)$ holds if $f_0^i(m_0^\ell) = f_1^i(m_0^\ell)$ holds. Moreover, we have

$$G[f_b^i, f_1^i, \perp, i](1\text{CT.MSK}^{(\ell^*)}, S^{(\ell^*)}, 0) = u_i = G[f_b^i, f_1^i, u_i, i](\perp, \perp, 0).$$

If \mathcal{A} makes only one key query under every index $i \in [q]$, then so does \mathcal{B} . Therefore, since \mathcal{A} is a valid adversary for HYBRD, \mathcal{B} is a valid adversary for iSKFE, and thus we have $\text{Adv}_{\text{iSKFE}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]|$. \mathcal{B} perfectly simulates Game (5, $\ell^* - 1$) if $\beta = 0$. On the other hand, \mathcal{B} perfectly simulates Game (1, ℓ^*) if $\beta = 1$. Moreover, \mathcal{B} outputs 1 if and only if \mathcal{A} succeeds in guessing the value of b . Therefore, we have $\text{Adv}_{\text{iSKFE}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_{(5, \ell^* - 1)}] - \Pr[\text{SUC}_{(1, \ell^*)}]|$ thus $|\Pr[\text{SUC}_{(5, \ell^* - 1)}] - \Pr[\text{SUC}_{(1, \ell^*)}]| \leq \epsilon_{\text{iSKFE}}$ holds. \square (**Lemma 8.9**)

Lemma 8.10. For every $\ell^* \in [p]$, $|\Pr[\text{SUC}_{(1, \ell^*)}] - \Pr[\text{SUC}_{(2, \ell^*)}]| \leq \epsilon_{\text{PRF}}$.

The proof is straightforward thus omitted.

Lemma 8.11. For every $\ell^* \in [p]$, $|\Pr[\text{SUC}_{(2, \ell^*)}] - \Pr[\text{SUC}_{(3, \ell^*)}]| \leq \epsilon_{1\text{CT}}$.

Proof of Lemma 8.11. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks 1CT.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} chooses $b \leftarrow \{0, 1\}$. When, \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, \mathcal{B} first computes $\{\text{CT}^{(\ell)}\}_{\ell \in [p]}$ and $\{1\text{CT.CT}^{(\ell)}\}_{\ell \in [p]}$ as follows.

- For every $\ell < \ell^*$, \mathcal{B} computes $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT.MSK}^{(\ell)}, S^{(\ell)}, 1))$ and $1\text{CT.CT}^{(\ell)} \leftarrow 1\text{CT.Enc}(1\text{CT.MSK}^{(\ell)}, m_1^\ell)$.
- \mathcal{B} computes $\text{CT}^{(\ell^*)} \leftarrow \text{Enc}(\text{MSK}, (\perp, \perp, 0))$. In addition, \mathcal{B} sends $(m_b^{\ell^*}, m_1^{\ell^*})$ to the challenger and gets the answer $1\text{CT.CT}^{(\ell^*)}$.
- For every $\ell > \ell^*$, \mathcal{B} computes $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT.MSK}^{(\ell)}, S^{(\ell)}, 0))$ and $1\text{CT.CT}^{(\ell)} \leftarrow 1\text{CT.Enc}(1\text{CT.MSK}^{(\ell)}, m_b^\ell)$.

Finally, \mathcal{B} sets $\text{Hyb.CT}^{(\ell)} \leftarrow (\text{CT}^{(\ell)}, 1\text{CT.CT}^{(\ell)})$ for every $\ell \in [p]$, and sends $\{\text{Hyb.CT}^{(\ell)}\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, \mathcal{B} first queries (f_b^i, f_1^i) to the challenger as a key query and gets the answer $1\text{CT}.sk_f^i$. Then, \mathcal{B} generates $sk_G^i \leftarrow \text{iKG}(\text{MSK}, G[f_b^i, f_1^i, 1\text{CT}.sk_f^i], i)$ and returns it to \mathcal{A} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs 1 if $b = b'$. Otherwise, \mathcal{B} outputs 0.

Let β be the challenge bit between the challenger and \mathcal{B} . For every key query (i, f_0^i, f_1^i) made by \mathcal{A} , $f_0^i(m_0^{\ell^*}) = f_1^i(m_0^{\ell^*})$ holds since \mathcal{A} is a valid adversary for HYBRD. In addition, \mathcal{B} sends only one message tuple in the initialization step. Therefore, \mathcal{B} is a valid adversary for 1CT, and thus we have $\text{Adv}_{1\text{CT}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]|$. We see that \mathcal{B} perfectly simulates Game $(2, \ell^*)$ if $\beta = 0$. On the other hand, \mathcal{B} perfectly simulates Game $(3, \ell^*)$ if $\beta = 1$. Moreover, \mathcal{B} outputs 1 if and only if \mathcal{A} succeeds in guessing the value of b . Therefore, we have $\text{Adv}_{1\text{CT}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_{(2, \ell^*)}] - \Pr[\text{SUC}_{(3, \ell^*)}]|$ thus $|\Pr[\text{SUC}_{(2, \ell^*)}] - \Pr[\text{SUC}_{(3, \ell^*)}]| \leq \epsilon_{1\text{CT}}$ holds. \square (**Lemma 8.11**)

Lemma 8.12. For every $\ell^* \in [p]$, $|\Pr[\text{SUC}_{(3, \ell^*)}] - \Pr[\text{SUC}_{(4, \ell^*)}]| \leq \epsilon_{\text{PRF}}$.

The proof is straightforward thus omitted.

Lemma 8.13. For every $\ell^* \in [p]$, $|\Pr[\text{SUC}_{(4, \ell^*)}] - \Pr[\text{SUC}_{(5, \ell^*)}]| \leq \epsilon_{\text{iSKFE}}$.

The proof is almost the same as that of Lemma 8.9 thus is omitted.

Lemma 8.14. $|\Pr[\text{SUC}_{(5, p)}] - \Pr[\text{SUC}_6]| \leq \epsilon_{\text{iSKFE}}$.

Proof of Lemma 8.14. The only difference between Game $(5, p)$ and 6 is how $sk_G^{(i)}$ is generated for every $i \in [q]$. In Game $(5, p)$, it is generated as $sk_G^{(i)} \leftarrow \text{iKG}(\text{MSK}, G[f_b^i, f_1^i, \perp, i], i)$. On the other hand, in Game 6, it is generated as $sk_G^{(i)} \leftarrow \text{iKG}(\text{MSK}, G[\perp, f_1^i, \perp, i], i)$. Here, in both games, for every $\ell \in [p]$, $\text{CT}^{(\ell)}$ is generated as $\text{CT}^{(\ell)} \leftarrow \text{Enc}(\text{MSK}, (1\text{CT}.MSK^{(\ell)}, S^{(\ell)}, 1))$. Then, for every $i \in [q]$ and $\ell \in [p]$, we have

$$G[f_b^i, f_1^i, \perp, i](1\text{CT}.MSK^{(\ell)}, S^{(\ell)}, 1) = G[\perp, f_1^i, \perp, i](1\text{CT}.MSK^{(\ell)}, S^{(\ell)}, 1).$$

This is because f_b^i is ignored in the left hand side. Therefore, we can construct an adversary attacking iSKFE whose advantage is $|\Pr[\text{SUC}_{(5, p)}] - \Pr[\text{SUC}_6]|$ thus $|\Pr[\text{SUC}_{(5, p)}] - \Pr[\text{SUC}_6]| \leq \epsilon_{\text{iSKFE}}$ holds. \square (**Lemma 8.14**)

From inequality (31) and Lemmas 8.9 to 8.14, we see that inequality (8.8) holds. \square (**Theorem 8.7**)

9 New PRODUCT Construction for iSKFE

We now introduce our main tool for increasing the number of functional decryption keys of an iSKFE scheme. By using two iSKFE schemes as building blocks, the construction produces a new iSKFE scheme whose index space is the product of those of the building block schemes. Our PRODUCT construction is based on the PRODUCT construction for PKFE schemes proposed by Li and Micciancio [LM16]. However, as mentioned in Section 2, in order to accomplish the security proof, we cannot use their construction in the secret-key setting straightforwardly. Then, we adopt a ciphertext-embedding strategy used by Brakerski et al. [BKS18] in the context of multi-input SKFE.

Let $\text{Outer} = (\text{Out.Setup}, \text{Out.iKG}, \text{Out.Enc}, \text{Out.Dec})$ and $\text{Inner} = (\text{In.Setup}, \text{In.iKG}, \text{In.Enc}, \text{In.Dec})$ be iSKFE schemes. We assume that the index spaces of Outer and Inner are \mathcal{I}_{Out} and \mathcal{I}_{In} , respectively. Let $\{F_S : \mathcal{I}_{\text{Out}} \times [0, 2] \rightarrow \{0, 1\}^\lambda \mid S \in \{0, 1\}^\lambda\}$ and $\{F'_S : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \mid$

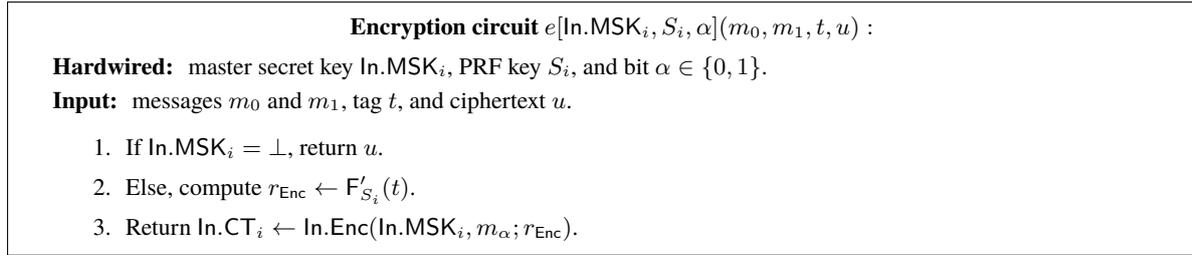


Figure 21: Construction of an encryption circuit e .

$S \in \{0, 1\}^\lambda$ be PRFs. Then, using Outer , Inner , F , and F' , we construct an iSKFE scheme $\text{PRDCT} = (\text{Prd.Setup}, \text{Prd.iKG}, \text{Prd.Enc}, \text{Prd.Dec})$ as follows. Note that the index space of PRDCT is $\mathcal{I}_{\text{Out}} \times \mathcal{I}_{\text{In}}$. Moreover, if the function space of Inner is \mathcal{F} , then that of PRDCT is also \mathcal{F} . We assume that Outer supports sufficiently large function class that particularly includes the encryption circuit e described in Figure 21. In addition, we assume that all of randomness spaces of In.Setup , In.iKG , and In.Enc are $\{0, 1\}^\lambda$.

Construction. The scheme consists of the following algorithms.

$\text{Prd.Setup}(1^\lambda) :$

- Generate $\text{Out.MSK} \leftarrow \text{Out.Setup}(1^\lambda)$ and $S \xleftarrow{r} \{0, 1\}^\lambda$.
- Return $\text{MSK} := (\text{Out.MSK}, S)$.

$\text{Prd.iKG}(\text{MSK}, f, (i, j)) :$

- Parse $(\text{Out.MSK}, S) \leftarrow \text{MSK}$.
- Compute $r_{\text{Setup}}^i \leftarrow F_S(i\|0)$, $S_i \leftarrow F_S(i\|1)$, and $r_{\text{iKG}}^i \leftarrow F_S(i\|2)$.
- Generate $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda; r_{\text{Setup}}^i)$.
- Compute $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}, e[\text{In.MSK}_i, S_i, 0], i; r_{\text{iKG}}^i)$ and $\text{In.sk}_f^{i,j} \leftarrow \text{In.iKG}(\text{In.MSK}_i, f, j)$.
- Return $sk_f := (\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j})$.

$\text{Prd.Enc}(\text{MSK}, m) :$

- Parse $(\text{Out.MSK}, S) \leftarrow \text{MSK}$.
- Generate $t \xleftarrow{r} \{0, 1\}^\lambda$.
- Compute $\text{Out.CT} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m, \perp, t, \perp))$.
- Return $\text{CT} := \text{Out.CT}$.

$\text{Prd.Dec}(sk_f, \text{CT}) :$

- Parse $(\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j}) \leftarrow sk_f$ and $\text{Out.CT} \leftarrow \text{CT}$.
- Compute $\text{In.CT} \leftarrow \text{Out.Dec}(\text{Out.sk}_{e_i}, \text{Out.CT})$.
- Return $y \leftarrow \text{In.Dec}(\text{In.sk}_f^{i,j}, \text{In.CT})$.

Remark 9.1. In the key generation algorithm, we use a PRF value to generate Out.sk_{e_i} , but do not to generate $\text{In.sk}_f^{i,j}$. The reason is as follows. Suppose that the adversary already sent a key query $((i, j), f_0^{i,j}, f_1^{i,j})$ and the challenger generates $(\text{Out.sk}_{e_i}, \text{In.sk}_{f_b}^{i,j})$ where $b \in \{0, 1\}$. After that, if the adversary sends a key query $((i, j'), f_0^{i,j'}, f_1^{i,j'})$ where $j' \neq j$ (this is allowed), the challenger must generate $(\text{Out.sk}_{e_i}, \text{In.sk}_{f_b}^{i,j'})$. In other words, the functional key of the outer scheme must be the same as that returned for the key query $((i, j), f_0^{i,j}, f_1^{i,j})$. Otherwise, we cannot complete the simulation when we use the security of the outer scheme since Out.sk_{e_i} is given to the simulator from the challenger. On the other hand, $\text{In.sk}_f^{i,j}$ is returned only once since the same index key query (i, j) is not allowed. Thus, we need a PRF for generating Out.sk_{e_i} .

Correctness. Let $|\mathcal{I}_{\text{Out}}| = q_{\text{Out}}$. In the construction, we use q_{Out} instances of Inner and thus q_{Out} master secret keys are generated in Prd.iKG. In addition, we let Out.iKG release the same functional key Out.sk_{e_i} for the same index $i \in \mathcal{I}_{\text{Out}}$ since Outer can release only q_{Out} functional keys. In order to ensure that only q_{Out} master secret keys $\{\text{In.MSK}_i\}_{i \in [q_{\text{Out}}]}$ and functional keys $\{\text{Out.sk}_{e_i}\}_{i \in [q_{\text{Out}}]}$ are generated, we manage them as one PRF key S . Then, if we decrypt Out.CT by Out.sk_{e_i} , it is re-encrypted to a ciphertext under the master secret key In.MSK_i . Thus, the correctness of PRDCT follows from those of Outer and Inner.

Efficiency. Let $|\text{In.Enc}|$ and $|F'|$ denote the size of the circuits computing In.Enc and F' , respectively. Then, the size of e is

$$\begin{aligned} |e| &= |\text{In.MSK}| + \lambda + 1 + |\text{In.Enc}| + |F'| \leq 2|\text{In.Enc}| + \text{poly}(\lambda) \\ &\leq |\text{In.Enc}| \cdot \text{poly}_e(\lambda), \end{aligned} \quad (32)$$

where poly_e is a fixed polynomial.

Let $\text{Out.t}_{\text{Enc}}$ and $\text{Out.t}_{\text{iKG}}$ be bounds of the running time of Out.Enc and Out.iKG. Let In.t_{Enc} and In.t_{iKG} be bounds of the running time of In.Enc and In.iKG. Let $\text{Prd.t}_{\text{Enc}}$ and $\text{Prd.t}_{\text{iKG}}$ be bounds of the running time of Prd.Enc and Prd.iKG. Note that the length of a ciphertext output by In.Enc is bounded by its running time In.t_{Enc} . Then, we have

$$\begin{aligned} \text{Prd.t}_{\text{Enc}}(\lambda, n, s) &= \text{Out.t}_{\text{Enc}}(\lambda, 2n + \lambda + \text{In.t}_{\text{Enc}}(\lambda, n, s), |e|), \\ \text{Prd.t}_{\text{iKG}}(\lambda, n, s) &= \text{Out.t}_{\text{iKG}}(\lambda, 2n + \lambda + \text{In.t}_{\text{Enc}}(\lambda, n, s), |e|) + \text{In.t}_{\text{iKG}}(\lambda, n, s). \end{aligned}$$

Security. Let $|\mathcal{I}_{\text{Out}}| = q_{\text{Out}}$ and $|\mathcal{I}_{\text{In}}| = q_{\text{In}}$. Then, we have the following theorem.

Theorem 9.2. *Let Outer be (q_{Out}, δ) -selective-message function private iSKFE, and Inner (q_{In}, δ) -selective-message function private iSKFE. Let F and F' be δ -secure PRF. Then, PRDCT is $(q_{\text{Out}} \cdot q_{\text{In}}, \delta)$ -selective-message function private iSKFE.*

To prove Theorem 9.2, we use the following lemma.

Lemma 9.3. *We assume that the advantage of any adversary attacking Outer, Inner, F , and F' is bounded by ϵ_{Out} , ϵ_{In} , ϵ_{PRF} , and ϵ_{PRF} , respectively. Let \mathcal{A} be an adversary that attacks the selective message function privacy of PRDCT. We assume that \mathcal{A} makes key queries for at most q different indices $\{i_k\}_{k \in [q]} \in \mathcal{I}_{\text{Out}}^q$. Then, we have*

$$\text{Adv}_{\text{PRDCT}, \mathcal{A}}^{\text{sm-fp}}(\lambda) \leq 2((2q + 2) \cdot \epsilon_{\text{Out}} + q \cdot \epsilon_{\text{In}} + (2q + 1)\epsilon_{\text{PRF}}).$$

This lemma immediately implies Theorem 9.2 since if all of Outer, Inner, F , and F' are δ -secure, then so does PRDCT. Below, we prove Lemma 9.3.

Proof of Lemma 9.3. First, consider the following sequence of games. We assume that the number of message tuples \mathcal{A} queries as the challenge messages is p at most, where p is a polynomial of λ .

Game 0 This is the original selective-message function privacy game regarding PRDCT.

Initialization First, the challenger sends security parameter 1^λ to \mathcal{A} . Then, \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [p]}$ to the challenger. Next, the challenger generates $\text{Out.MSK} \leftarrow \text{Out.Setup}(1^\lambda)$ and $S \leftarrow \{0, 1\}^\lambda$, and chooses a challenge bit $b \leftarrow \{0, 1\}$. Then, the challenger generates $t^{(\ell)} \leftarrow \{0, 1\}^\lambda$ and $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, \perp, t^{(\ell)}, \perp))$ for every $\ell \in [p]$, and returns $\{(\text{Out.CT}^{(\ell)})\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\text{Out}} \times \mathcal{I}_{\text{In}} \times \mathcal{F} \times \mathcal{F}$, the challenger first generates $r_{\text{Setup}}^i \leftarrow F_S(i||0)$, $S_i \leftarrow F_S(i||1)$, $r_{\text{iKG}}^i \leftarrow F_S(i||2)$, and $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda; r_{\text{Setup}}^i)$. Then the challenger computes $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}, e[\text{In.MSK}_i, S_i, 0], i; r_{\text{iKG}}^i)$ and $\text{In.sk}_f^{i,j} \leftarrow \text{In.iKG}(\text{In.MSK}_i, f_b^{i,j}, j)$, and return $(\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j})$ to \mathcal{A} .

Final phase \mathcal{A} output b' .

Game 1 Same as Game 0 except how the challenger responds to a key query made by \mathcal{A} . At the initialization step, the challenger prepares a list \mathcal{L} which stores an index $i \in \mathcal{I}$ and corresponding master secret key In.MSK_i , a PRF key S_i , and a functional key Out.sk_{e_i} . At the beginning of the game, \mathcal{L} is empty. Recall that the adversary sends not only two functions but also the index as a key query in the index-based SKFE. When \mathcal{A} makes a key query $((i, j), f_0^{i,j}, f_1^{i,j})$, the challenger first checks whether there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$ in \mathcal{L} . If so, the challenger responds to the key query using $(\text{In.MSK}_i, S_i)$. Otherwise, the challenger generates r_{Setup}^i , S_i , and r_{iKG}^i as truly random strings, and generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda; r_{\text{Setup}}^i)$ and $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{MSK}, e[\text{In.MSK}_i, S_i, 0], i)$. Then, the challenger responds to the key query, and finally adds the entry $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$ to \mathcal{L} .

Game 2 Same as Game 1 except that the challenger generates $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \perp))$ for every $\ell \in [p]$.

For every $k^* \in [q]$, we define the following games.

Game (7, 0) We define Game (7, 0) as the same game as Game 2.

The goal is changing the ciphertext of m_b^ℓ into that of m_1^ℓ regardless of the value of b . The issue is that we need to erase In.MSK_i from Out.sk_{e_i} to use the security of Inner for every i . To do so, we perform the following steps for every i . First, we directly embed $\text{In.Enc}(\text{In.MSK}_i, m_b^\ell)$ into $\text{Out.CT}^{(\ell)}$ using the security of Outer. After we change $\text{In.Enc}(\text{In.MSK}_i, m_b)$ into $\text{In.Enc}(\text{In.MSK}_i, m_1)$, we can change encryption of m_b by Outer into encryption of m_1 by Outer.

Game (3, k^*) Same as Game (7, $k^* - 1$) except the manner the challenger generates challenge ciphertext and responds to key queries.

In the initialization step, the challenger generates $\text{In.MSK}^* \leftarrow \text{In.Setup}(1^\lambda)$ and $S^* \leftarrow \{0, 1\}^\lambda$. Then, for every $\ell \in [p]$, the challenger generates $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)}))$, where $u^{*(\ell)} \leftarrow \text{In.Enc}(\text{In.MSK}^*, m_b^\ell; r_{\text{Enc}}^{*(\ell)})$ and $r_{\text{Enc}}^{*(\ell)} \leftarrow F_{S^*}^*(t^{(\ell)})$.

In addition, when \mathcal{A} makes a key query $((i, j), f_0^{i,j}, f_1^{i,j})$, the challenger responds as follows.

- When $|\mathcal{L}| < k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, the challenger responds to the query by using them. Otherwise, the challenger first generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda)$, $S_i \leftarrow \{0, 1\}^\lambda$, and $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{MSK}, e[\text{In.MSK}_i, S_i, 1], i)$. Then, the challenger responds using them, and adds them to \mathcal{L} .

- When $|\mathcal{L}| = k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, the challenger responds to the query by using them. Otherwise, the challenger first sets $\text{In.MSK}_i \leftarrow \text{In.MSK}^*$, $S_i \leftarrow S^*$, and $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}_i, e[\perp, \perp, 0], i)$. Then, the challenger responds using them, and adds them to \mathcal{L} . We call this index i^* .
- When $|\mathcal{L}| > k^* - 1$, the challenger responds in the same way as Game $(6, k^* - 1)$.

Game $(4, k^*)$ Same as Game $(3, k^*)$ except that the challenger generates $r_{\text{Enc}}^{(\ell)}$ as a truly random string for every $\ell \in [p]$.

Game $(5, k^*)$ Same as Game $(4, k^*)$ except that the challenger generates $u^{*(\ell)} \leftarrow \text{In.Enc}(\text{In.MSK}_{i^*}, m_1^\ell)$ for every $\ell \in [p]$. In addition, the challenger generates $\text{In.sk}_f^{(i^*, j)} \leftarrow \text{In.iKG}(\text{In.MSK}_{i^*}, f_1^{(i^*, j)}, j)$ for every $j \in \mathcal{I}_{\text{In}}$.

Game $(6, k^*)$ Same as Game $(5, k^*)$ except that the challenger generates $r_{\text{Enc}}^{(\ell)} \leftarrow F'_{S_{i^*}}(t^{(\ell)})$ for every $\ell \in [p]$.

Game $(7, k^*)$ Same as Game $(6, k^*)$ except that for every $\ell \in [p]$, the challenger generates $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \perp))$. In addition, the challenger generates $\text{Out.sk}_{e_{i^*}} \leftarrow \text{Out.iKG}(\text{MSK}, e[\text{In.MSK}_{i^*}, S_{i^*}, 1], i^*)$.

We define one additional game.

Game 8 Same as Game $(7, q)$ except that the challenger generates $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (\perp, m_1^\ell, t^{(\ell)}, \perp))$ for every $\ell \in [p]$.

For every $h \in \{0, 1, 2, 8\}$, let SUC_h be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game h . Similarly, for every $h \in \{3, \dots, 7\}$ and $k^* \in [q]$, let $\text{SUC}_{(h, k^*)}$ be the event that \mathcal{A} succeeds in guessing b in Game (h, k^*) . In Game 8, the challenge bit b is information theoretically hidden from the view of \mathcal{A} thus $|\Pr[\text{SUC}_8] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned}
\frac{1}{2} \cdot \text{Adv}_{\text{PRDCT}, \mathcal{A}}^{\text{sm-fp}}(\lambda) &= |\Pr[\text{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| + |\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \\
&\quad + \sum_{k^* \in [q]} |\Pr[\text{SUC}_{(7, k^*-1)}] - \Pr[\text{SUC}_{(3, k^*)}]| \\
&\quad + \sum_{k^* \in [q]} \sum_{h=3}^6 |\Pr[\text{SUC}_{(h, k^*)}] - \Pr[\text{SUC}_{(h+1, k^*)}]| \\
&\quad + |\Pr[\text{SUC}_{(7, q)}] - \Pr[\text{SUC}_8]| \tag{33}
\end{aligned}$$

Below, we estimate each term on the right side of inequality (33).

Lemma 9.4. $|\Pr[\text{SUC}_0] - \Pr[\text{SUC}_1]| \leq \epsilon_{\text{PRF}}$.

The proof of it is straightforward thus omitted.

Lemma 9.5. $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \leq \epsilon_{\text{Out}}$.

Proof of Lemma 9.5. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks Outer.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} chooses $b \xleftarrow{r} \{0, 1\}$. When \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, \mathcal{B} sends $\{(m_b^\ell, \perp, t^{(\ell)}, \perp), (m_b^\ell, m_1^\ell, t^{(\ell)}, \perp)\}_{\ell \in [p]}$ to the challenger and returns the answer $\{\text{Out.CT}^{(\ell)}\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\text{Out}} \times \mathcal{I}_{\text{In}} \times \mathcal{F} \times \mathcal{F}$, \mathcal{B} first checks whether there is an entry whose first component is i in \mathcal{L} . If so, using the entry $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds to the key query. Otherwise, \mathcal{B} first generates $r_{\text{Setup}}^i, S_i, r_{\text{iKG}}^i \xleftarrow{r} \{0, 1\}^\lambda$, and computes $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda; r_{\text{Setup}}^i)$. Then, \mathcal{B} queries $(i, e[\text{In.MSK}_i, S_i, 0], e[\text{In.MSK}_i, S_i, 0])$ to the challenger as a key query, and obtains the answer Out.sk_{e_i} . Next, \mathcal{B} generates $\text{In.sk}_f^{i,j} \leftarrow \text{In.iKG}(\text{In.MSK}_i, f_b^{i,j}; j)$ and returns $(\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j})$ to \mathcal{A} . Finally, \mathcal{B} adds $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$ to \mathcal{L} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs 1 if $b = b'$. Otherwise, \mathcal{B} outputs 0.

Let β be the challenge bit between the challenger and \mathcal{B} . Note that the encryption circuit $e[\text{In.MSK}_i, S_i, 0]$ ignores second component of the input, and thus for every $\ell \in [p]$ and every key query $i \in \mathcal{I}_{\text{Out}}$, we have $e[\text{In.MSK}_i, S_i, 0](m_b^\ell, \perp, t^{(\ell)}, \perp) = e[\text{In.MSK}_i, S_i, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, \perp)$. In addition, \mathcal{B} makes 1 key query under every index $i \in \mathcal{I}$ at most. Therefore, \mathcal{B} is a valid adversary for Outer, and thus we have $\text{Adv}_{\text{Outer}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]|$. We see that \mathcal{B} perfectly simulates Game 1 if $\beta = 0$. On the other hand, \mathcal{B} perfectly simulates Game 2 if $\beta = 1$. Moreover, \mathcal{B} outputs 1 if and only if \mathcal{A} succeeds in guessing the value of b . Therefore, we have $\text{Adv}_{\text{Outer}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]|$ thus $|\Pr[\text{SUC}_1] - \Pr[\text{SUC}_2]| \leq \epsilon_{\text{Out}}$ holds. \square (**Lemma 9.5**)

Lemma 9.6. For every $k^* \in [q]$, $|\Pr[\text{SUC}_{(7, k^* - 1)}] - \Pr[\text{SUC}_{(3, k^*)}]| \leq \epsilon_{\text{Out}}$.

Proof of Lemma 9.6. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks Outer.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} chooses $b \xleftarrow{r} \{0, 1\}$, and generates $\text{In.MSK}^* \leftarrow \text{In.Setup}(1^\lambda)$ and $S^* \xleftarrow{r} \{0, 1\}^\lambda$. When \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, for every $\ell \in [p]$, \mathcal{B} first generates $u^{*(\ell)} \leftarrow \text{In.Enc}(\text{In.MSK}^*, m_b^\ell; r_{\text{Enc}}^{*(\ell)})$, where $r_{\text{Enc}}^{*(\ell)} \leftarrow F_{S^*}(t^{(\ell)})$. \mathcal{B} sends $\{(m_b^\ell, m_1^\ell, t^{(\ell)}, \perp), (m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)})\}_{\ell \in [p]}$ to the challenger and returns the answer $\{\text{Out.CT}^{(\ell)}\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\text{Out}} \times \mathcal{I}_{\text{In}} \times \mathcal{F} \times \mathcal{F}$, \mathcal{B} responds as follows.

- In the case $|\mathcal{L}| < k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds using them. Otherwise, \mathcal{B} first generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0, 1\}^\lambda$. Then, \mathcal{B} makes a key query $(i, e[\text{In.MSK}_i, S_i, 1], e[\text{In.MSK}_i, S_i, 1])$ to the challenger and obtains the answer Out.sk_{e_i} . Then, \mathcal{B} responds using them, and adds them to \mathcal{L} .
- In the case $|\mathcal{L}| = k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds using them. Otherwise, \mathcal{B} first sets $\text{In.MSK}_i := \text{In.MSK}^*$, $S_i := S^*$. Then, \mathcal{B} makes a key query $(i, e[\text{In.MSK}_i, S_i, 0], e[\perp, \perp, 0])$ to the challenger and obtains the answer Out.sk_{e_i} . Then, \mathcal{B} responds using them, and adds them to \mathcal{L} .
- In the case $|\mathcal{L}| > k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds using them. Otherwise, \mathcal{B} first generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0, 1\}^\lambda$. Then, \mathcal{B} makes a key query $(i, e[\text{In.MSK}_i, S_i, 0], e[\text{In.MSK}_i, S_i, 0])$ to the challenger and obtains the answer Out.sk_{e_i} . Then, \mathcal{B} responds using them, and adds them to \mathcal{L} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs 1 if $b = b'$. Otherwise, \mathcal{B} outputs 0.

Let β be the challenge bit between the challenger and \mathcal{B} . We can easily see that

$$e[\text{In.MSK}_i, S_i, \alpha](m_b^\ell, m_1^\ell, t^{(\ell)}, \perp) = e[\text{In.MSK}_i, S_i, \alpha](m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)})$$

hold for every $\ell \in [p]$, $i \in \mathcal{I}_{\text{Out}}$, and $\alpha \in \{0, 1\}$ since $u^{*(\ell)}$ is ignored. In addition, we have

$$e[\text{In.MSK}^*, S^*, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, \perp) = \text{In.Enc}(\text{In.MSK}^*, m_b^\ell; r_{\text{Enc}}^{*(\ell)}) = u^{*(\ell)} = e[\perp, \perp, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)})$$

for every $\ell \in [p]$. Therefore, \mathcal{B} is a valid adversary for Outer, and thus we have $\text{Adv}_{\text{Outer}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]|$. We see that \mathcal{B} perfectly simulates Game (7, $k^* - 1$) if $\beta = 0$. On the other hand, \mathcal{B} perfectly simulates Game (3, k^*) if $\beta = 1$. Moreover, \mathcal{B} outputs 1 if and only if \mathcal{A} succeeds in guessing the value of b . Therefore, we have $\text{Adv}_{\text{Outer}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_{(7, k^* - 1)}] - \Pr[\text{SUC}_{(3, k^*)}]|$, and thus $|\Pr[\text{SUC}_{(7, k^* - 1)}] - \Pr[\text{SUC}_{(3, k^*)}]| \leq \epsilon_{\text{Out}}$ holds. \square (**Lemma 9.6**)

Lemma 9.7. For every $k^* \in [q]$, $|\Pr[\text{SUC}_{(3, k^*)}] - \Pr[\text{SUC}_{(4, k^*)}]| \leq \epsilon_{\text{PRF}}$.

The proof is straightforward thus omitted.

Lemma 9.8. For every $k^* \in [q]$, $|\Pr[\text{SUC}_{(4, k^*)}] - \Pr[\text{SUC}_{(5, k^*)}]| \leq \epsilon_{\text{In}}$.

Proof of Lemma 9.8. Using the adversary \mathcal{A} , we construct the following adversary \mathcal{B} that attacks Inner.

Initialization On input security parameter 1^λ , \mathcal{B} sends it to \mathcal{A} . Then, \mathcal{B} chooses $b \xleftarrow{r} \{0, 1\}$, and generates $\text{Out.MSK}^* \leftarrow \text{Out.Setup}(1^\lambda)$. When, \mathcal{A} sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, \mathcal{B} sends $\{(m_b^\ell, m_1^\ell)\}_{\ell \in [p]}$ to the challenger and obtains the answer $\{\text{In.CT}^{(\ell)}\}_{\ell \in [p]}$. Finally, \mathcal{B} compute $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \text{In.CT}^{(\ell)}))$, and returns $\{\text{Out.CT}^{(\ell)}\}_{\ell \in [p]}$ to \mathcal{A} .

Key queries When \mathcal{A} makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\text{Out}} \times \mathcal{I}_{\text{In}} \times \mathcal{F} \times \mathcal{F}$, \mathcal{B} responds as follows.

- In the case $|\mathcal{L}| < k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds using them. Otherwise, \mathcal{B} first generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0, 1\}^\lambda$, and $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}, e[\text{In.MSK}_i, S_i, 1], i)$. Then, \mathcal{B} responds using them, and adds them to \mathcal{L} .
- In the case $|\mathcal{L}| = k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$, \mathcal{B} responds using them. Otherwise, \mathcal{B} first makes a key query $(j, f_b^{i,j}, f_1^{i,j})$ to the challenger and obtains the answer $\text{In.sk}_f^{i,j}$. Then, \mathcal{B} generates $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}, e[\perp, \perp, 0], i)$, and returns $(\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j})$. Finally, \mathcal{B} sets $i^* = i$ and adds $(i^*, \perp, \perp, \text{Out.sk}_{e_i})$ to \mathcal{L} .
- In the case $|\mathcal{L}| > k^* - 1$, if there is an entry of the form $(i, \text{In.MSK}_i, S_i, \text{Out.sk}_{e_i})$ and $i \neq i^*$, the challenger responds using them. If $i = i^*$, \mathcal{B} makes a key query $(j, f_b^{i,j}, f_1^{i,j})$ to the challenger, obtains the answer $\text{In.sk}_f^{i,j}$, and returns $(\text{Out.sk}_{e_i}, \text{In.sk}_f^{i,j})$ to \mathcal{A} . Otherwise, \mathcal{B} first generates $\text{In.MSK}_i \leftarrow \text{In.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0, 1\}^\lambda$, and $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{Out.MSK}, e[\text{In.MSK}_i, S_i, 0], i)$. Then, \mathcal{B} responds using them, and adds them to \mathcal{L} .

Final phase When \mathcal{A} terminates with output b' , \mathcal{B} outputs 1 if $b = b'$. Otherwise, \mathcal{B} outputs 0.

Let β be the challenge bit between the challenger and \mathcal{B} . Since \mathcal{A} is a valid adversary for PRDCT, for every ℓ and function query $(i, j, f_0^{i,j}, f_1^{i,j})$, it holds that $f_0^{i,j}(m_0^\ell) = f_1^{i,j}(m_1^\ell)$, and \mathcal{B} makes 1 key query under every index $j \in \mathcal{I}_{\text{In}}$ at most. Therefore, \mathcal{B} is a valid adversary for Inner, and thus we have $\text{Adv}_{\text{Inner}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\beta' = 1 \mid \beta = 0] - \Pr[\beta' = 1 \mid \beta = 1]|$. We see that \mathcal{B} perfectly simulates Game $(4, k^*)$ if $\beta = 0$. On the other hand, \mathcal{B} perfectly simulates Game $(5, k^*)$ if $\beta = 1$. Moreover, \mathcal{B} outputs 1 if and only if \mathcal{A} succeeds in guessing the value of b . Therefore, we have $\text{Adv}_{\text{Inner}, \mathcal{B}}^{\text{sm-fp}}(\lambda) = |\Pr[\text{SUC}_{(4, k^*)}] - \Pr[\text{SUC}_{(5, k^*)}]|$, and thus $|\Pr[\text{SUC}_{(4, k^*)}] - \Pr[\text{SUC}_{(5, k^*)}]| \leq \epsilon_{\text{In}}$ holds. \square (**Lemma 9.8**)

Lemma 9.9. For every $k^* \in [q]$, $|\Pr[\text{SUC}_{(5, k^*)}] - \Pr[\text{SUC}_{(6, k^*)}]| \leq \epsilon_{\text{PRF}}$.

The proof is straightforward thus omitted.

Lemma 9.10. For every $k^* \in [q]$, $|\Pr[\text{SUC}_{(6, k^*)}] - \Pr[\text{SUC}_{(7, k^*)}]| \leq \epsilon_{\text{Out}}$.

The proof is almost the same as that of Lemma 9.6 thus is omitted.

Lemma 9.11. $|\Pr[\text{SUC}_{(7, p)}] - \Pr[\text{SUC}_8]| \leq \epsilon_{\text{Out}}$.

Proof of Lemma 9.11. The only difference between Game $(7, q)$ and 8 is how $\text{Out.CT}^{(\ell)}$ is generated for every $\ell \in [p]$. In Game $(7, q)$, it is generated as $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \perp))$. On the other hand, in Game 8, it is generated as $\text{Out.CT}^{(\ell)} \leftarrow \text{Out.Enc}(\text{Out.MSK}, (\perp, m_1^\ell, t^{(\ell)}, \perp))$. Here, in both games, for every $i \in \mathcal{I}_{\text{Out}}$, Out.sk_{e_i} is generated as $\text{Out.sk}_{e_i} \leftarrow \text{Out.iKG}(\text{MSK}, e[\text{In.MSK}_i, S_i, 1], i)$, or not generated. Then, for every $i \in \mathcal{I}_{\text{Out}}$ and $\ell \in [p]$, we have

$$e[\text{In.MSK}_i, S_i, 1](m_b^\ell, m_1^\ell, t^{(\ell)}, \perp) = e[\text{In.MSK}_i, S_i, 1](\perp, m_1^\ell, t^{(\ell)}, \perp).$$

This is because m_b^ℓ is ignored in the left hand side. Therefore, we can construct an adversary attacking Outer whose advantage is $|\Pr[\text{SUC}_{(7, q)}] - \Pr[\text{SUC}_8]|$, and thus $|\Pr[\text{SUC}_{(7, q)}] - \Pr[\text{SUC}_8]| \leq \epsilon$ holds. \square (**Lemma 9.11**)

From inequality (33) and Lemmas 9.4 to 9.11, inequality (9.3) holds. \square (**Theorem 9.2**)

10 Collusion-Resistant SKFE via Size-Shifting

In this section, we show how to construct collusion-resistant SKFE using the constructions introduced in the previous sections. More specifically, we show we can construct a collusion-resistant iSKFE scheme whose index space size is $\lambda^{\omega(1)}$ then transform the iSKFE scheme into a standard SKFE scheme (i.e., stateless scheme).

We basically increase the index space by using our new PRODUCT construction introduced in Section 9 repeatedly. However, the encryption time blows up polynomially whenever we apply our new PRODUCT construction, and thus we cannot directly repeat this construction $\omega(1)$ times. Therefore, we sandwich the size-shifting procedure using 1CT introduced in Section 8.3 between each application of our new PRODUCT construction to reduce the blow-up of the encryption time.

Below, we first give an intuition of our construction using size-shifting. Then, we show the actual construction of collusion-resistant iSKFE and analyze the efficiency and security of it. Finally, we give the transformation from iSKFE into SKFE.

10.1 Intuition of Size-Shifting

We give an intuition why we need size-shifting procedure in the construction. This intuition ignores many details, but we think it is helpful to understand the essence of the size-shifting procedure.

We first construct a λ -key iSKFE scheme Parallel_λ from the underlying single-key SKFE scheme. This is done by simply running λ instances of the single-key scheme as in Section 8.1. For simplicity, we assume that the underlying single-key scheme is fully succinct, and the encryption time of Parallel_λ is bounded by $|m|^c + O(\lambda^c)$,¹⁶ where m is a message to be encrypted and c is a constant.

Then, we construct λ^2 -key scheme PRDCT_2 by combining 2 instances of Parallel_λ using our PRODUCT construction in Section 9. The encryption time of PRDCT_2 is roughly

$$(|m|^c + O(\lambda^c))^c + O(\lambda^c) = |m|^{c^2} + O(\lambda^{c^2})$$

since a ciphertext is embedded into another ciphertext in the security proof of our PRODUCT construction. We note that the size of a ciphertext is bounded by the encryption time.

Analogously, the straightforward iterated application of our PRODUCT construction results in double exponential size blow-up in the number of iterations. Thus, we reduce the size blow-up by size-shifting.

Let 1CT be SKFE constructed in Section 8.2. For simplicity, we suppose that we can bound the encryption time of 1CT by $|m|^c + O(\lambda^c)$, where m is a message to be encrypted and c is the constant same as above. We construct HYBRD_2 by combining PRDCT_2 whose encryption time is $|m|^{c^2} + O(\lambda^{c^2})$ and a fresh instance of 1CT via the hybrid construction in Section 8.3.

Recall that the length of a master secret-key of 1CT is $O(\lambda)$ and thus the length of a message to be encrypted by PRDCT_2 in the hybrid construction is also $O(\lambda)$. Therefore, the encryption time of HYBRD_2 is roughly

$$(O(\lambda))^{c^2} + O(\lambda^{c^2}) + |m|^c + O(\lambda^c) = |m|^c + O(\lambda^{c^2}) ,$$

where m is a message to be encrypted by HYBRD_2 . Thus, we can separate double exponential term from the term related to the message length.

Then, we again increase the number of functional keys by our PRODUCT construction. We construct PRDCT_3 by using HYBRD_2 as Outer and a fresh instance of Parallel_λ as Inner. In this case, a ciphertext of Parallel_λ is embedded into a ciphertext of HYBRD_2 in the security proof. Therefore, the encryption time of PRDCT_3 is roughly

$$(|m|^c + O(\lambda^c))^c + O(\lambda^{c^2}) = |m|^{c^2} + O(\lambda^{c^2}) ,$$

where m is a message to be encrypted by PRDCT_3 . The encryption time no longer blows up double exponentially in the number of iterations.

Analogously, by applying the size-shifting between each application of our PRODUCT construction, the encryption time stays $|m|^{c^2} + O(\lambda^{c^2})$ no matter how many times we iterate the construction. Of course, the term $O(\lambda^{c^2})$ includes coefficient depends on the number of iterations. However, we can easily verify that the dependence is linear. Thus, we can iterate our PRODUCT construction with size-shifting $\omega(1)$ times and achieve a collusion-resistant scheme.

Remark 10.1 (Iterated linear and iterated square composition). In our iterated construction, on the k -th application of PRODUCT construction, we use λ^k -key scheme constructed so far as Outer and a fresh

¹⁶Precisely speaking, the time is bounded by $\lambda|m|^c + O(\lambda^c)$ due to the parallel construction. However, the factor λ (coefficient of $|m|^c$) is not a dominant factor, so we omit here. It is easy to see that the construction works even if we consider the factor. See Section 10.3.

instance of Parallel_λ as Inner. This iterated composition method is called iterated linear composition by Li and Micciancio [LM16] in the context of PKFE.

They also proposed another composition method called iterated square composition. In the iterated square composition, on the k -th application of PRODUCT construction, both Outer and Inner are the resulting scheme of the previous $k - 1$ compositions. In this composition, we can construct λ^{2^k} -key scheme by k times applications of PRODUCT construction.

One might think iterated square composition increases functional keys more efficiently than iterated linear composition. This is true for PKFE. However, the situation is different in SKFE since we use the nested-ciphertext-embedding technique in our PRODUCT construction for SKFE. We cannot iterate our PRODUCT construction for SKFE $\omega(1)$ times if we adopt iterated square composition. We can see the fact from the above intuition.

Suppose that we construct PRDCT_3 by using HYBRD_2 as both Outer and Inner in our PRODUCT construction. In this case, a ciphertext of HYBRD_2 is embedded into another ciphertext of HYBRD_2 in the security proof. Thus, the encryption time of PRDCT_3 is roughly

$$\left(|m|^c + O\left(\lambda^{c^2}\right)\right)^c + O\left(\lambda^{c^2}\right) = |m|^{c^2} + O\left(\lambda^{c^3}\right),$$

where m is a message to be encrypted by PRDCT_3 . We see that the additive term blows up double exponentially in the number of iterations while the term related to the message length does not due to size-shifting. This is the reason we adopt iterated linear composition in this work.

10.2 Construction of Collusion-Resistant iSKFE

To precisely define our collusion-resistant iSKFE, we introduce useful notations. We let

$$\text{iSKFE} = \langle \text{iSKFE}_{\text{Out}}, \text{iSKFE}_{\text{In}} \rangle_{\text{product}}$$

denote that an iSKFE scheme iSKFE is constructed from our new PRODUCT construction in Section 9 by using $\text{iSKFE}_{\text{Out}}$ as Outer and iSKFE_{In} as Inner. Moreover, we let

$$\text{iSKFE}' = \langle \text{iSKFE}, \text{1CT} \rangle_{\text{hyb}}$$

denote that an SKFE scheme SKFE' is constructed from our proposed hybrid encryption construction introduced in Section 8.3 by using iSKFE as a building block iSKFE scheme together with 1CT.

Let 1Key be single-key weakly succinct SKFE. We show how to construct collusion-resistant iSKFE based solely on 1Key.

First, we construct an iSKFE scheme Parallel_λ by applying the parallel construction introduced in Section 8.1 that the number of parallelization is λ . That is, we set $q = \lambda$ and use 1Key in the construction introduced in Section 8.1. We denote this construction by $\text{Parallel}_\lambda := \lambda \cdot \text{1Key}$. Note that the index space of Parallel_λ is $[\lambda]$.

We then recursively increase the number of functional keys as follows:

$$\begin{aligned} \text{PRDCT}_1 &:= \text{Parallel}_\lambda, \\ \text{HYBRD}_k &:= \langle \text{PRDCT}_k, \text{1CT} \rangle_{\text{hyb}} \quad (k = 1, \dots, \eta), \\ \text{PRDCT}_k &:= \langle \text{HYBRD}_{k-1}, \text{Parallel}_\lambda \rangle_{\text{product}} \quad (k = 2, \dots, \eta), \end{aligned}$$

where $\eta = \omega(1)$ which is concretely determined by the efficiency and security analysis. The second line is the size-shifting procedure by using our proposed hybrid construction. The third line is our new PRODUCT construction. The correctness of HYBRD_η follows from those of building block constructions.

Note that for every $k \in [\eta]$, both HYBRD_k and PRDCT_k support λ^k functional keys. In particular, the number of functional keys supported by HYBRD_η is λ^η thus is super-polynomial since $\eta = \omega(1)$.

Our collusion-resistant iSKFE scheme. Based on the notion above, our collusion-resistant iSKFE is HYBRD_η where $\eta = \omega(1)$. HYBRD_η is based solely on 1Key (recall that 1CT is constructed from OWFs), and the following theorem holds.

Theorem 10.2. *Assuming there exists $(1, \delta)$ -selective-message function private SKFE that is weakly succinct, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, for any polynomial poly, there exists (poly, δ) -selective-message function private iSKFE the size of whose index space is $\lambda^{\zeta^{1/2}}$ thus is super-polynomial in λ .*

10.3 Analysis of Our Collusion-Resistant iSKFE

In this section, we formally analyze the security and efficiency of HYBRD_η and prove Theorem 10.2.

Proof of Theorem 10.2. We start with the security bound analysis then move to the security analysis.

Security bound analysis. We assume that the advantages of any adversary attacking Parallel_λ , 1CT, and PRF is bounded by ϵ_{Para} , ϵ_{1CT} , and ϵ_{PRF} , respectively. For every $k \in [\eta]$, let ϵ_{Hyb_k} and ϵ_{Prd_k} be the upper bounds of the advantage of any adversary attacking HYBRD_k and PRDCT_k , respectively. Then, by Lemmata 8.8 and 9.3, for every $k \in [\eta - 1]$, we have

$$\begin{aligned} \epsilon_{\text{Hyb}_{k+1}} &\leq 2 \left((2p+1)\epsilon_{\text{Prd}_{k+1}} + p \cdot \epsilon_{\text{1CT}} + p \cdot \epsilon_{\text{PRF}} \right) \\ &\leq 4(2p+1) \left((2q+2)\epsilon_{\text{Hyb}_k} + q \cdot \epsilon_{\text{Para}} + (2q+1)\epsilon_{\text{PRF}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}} \right) \\ &\leq 8(2p+1)(q+1) \cdot \epsilon_{\text{Hyb}_k} + 8(2p+1)(q+1) (\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}), \end{aligned}$$

where p is a polynomial of λ denoting the number of message pairs an adversary attacking HYBRD_η queries, and q is the number of key queries made by the adversary. Then, by setting $Q := 8(2p+1)(q+1)$, we get

$$\epsilon_{\text{Hyb}_{k+1}} \leq Q \cdot \epsilon_{\text{Hyb}_k} + Q \cdot (\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}).$$

Therefore, it holds that

$$\begin{aligned} \epsilon_{\text{Hyb}_\eta} &\leq Q^{\eta-1} \cdot \epsilon_{\text{Hyb}_1} + \left(\sum_{k=0}^{\eta-2} Q^k \right) \cdot Q(\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}) \\ &\leq Q^{\eta-1} \cdot 2 \left((p+1) \cdot \epsilon_{\text{Para}} + p \cdot \epsilon_{\text{1CT}} + p \cdot \epsilon_{\text{PRF}} \right) \\ &\quad + (\eta-2) \cdot Q^{\eta-2} \cdot (\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}) \\ &\leq (\eta-2) \cdot Q^{\eta-1} \cdot 3(p+1) \cdot (\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}). \end{aligned}$$

Since p and q are polynomials of λ , and $Q = 8(2p+1)(q+1)$, we get

$$\epsilon_{\text{Hyb}_\eta} \leq \lambda^{O(\eta)} \cdot (\epsilon_{\text{Para}} + \epsilon_{\text{1CT}} + \epsilon_{\text{PRF}}). \quad (34)$$

Our assumption is that 1Key is a $(1, \delta)$ -selective-message function private SKFE scheme, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, from Theorem 8.1, Parallel_λ is also δ -secure. If there exists a $(1, \delta)$ -selective-message function private SKFE scheme, then there also exists a single-ciphertext (poly, δ) -selective-message function private SKFE scheme 1CT since it can be constructed based only on δ -secure one-way functions as we show in Section 8.2. In addition, we can construct a δ -secure PRF from

δ -secure one-way functions. Then, by using such δ -secure primitives as building blocks of HYBRD_η and setting $\eta = \zeta^{1/2}$, we obtain

$$\epsilon_{\text{Hyb}_\eta} \leq \lambda^{O(\eta)} \cdot \delta^{\Omega(1)} = \lambda^{O(\zeta^{1/2})} \cdot (\lambda^{-\zeta})^{\Omega(1)} \leq (\lambda^{-\zeta})^{\Omega(1)} = \delta^{\Omega(1)}.$$

One might think it is sufficient that we set η as slightly super-constant (e.g., $\log \log \lambda$) regardless of the security bound of the building block scheme since we can ensure that HYBRD_η supports super-polynomial number of keys and is δ -secure if we do so. This is not the case. HYBRD_η is an iSKFE scheme and thus we finally need to transform it into an SKFE scheme. As stated in Remark 3.19, in this transformation, the security bound of the resulting SKFE scheme is $\delta + \lambda^{-\eta}$. Therefore, we cannot make the resulting collusion-resistant SKFE scheme quasi-polynomially (resp. sub-exponentially) secure if we set η as slightly super-constant such as $\log \log \lambda$ when transforming quasi-polynomially (resp. sub-exponentially) secure single-key SKFE scheme. See Section 10.4 for more details.

This completes the security bound analysis.

Efficiency analysis. We show that each algorithm of HYBRD_η runs in polynomial time of λ, n and s , where s and n are the maximum size and input length of functions supported by HYBRD_η .

For every $k \in [\eta]$, we use new instances of Parallel_λ and 1CT when constructing PRDCT_k and HYBRD_k , respectively. In the following, we denote these two schemes as $\text{Parallel}_\lambda^{(k)}$ and $1\text{CT}^{(k)}$ to emphasize that they are instances of Parallel_λ and 1CT used when constructing PRDCT_k and HYBRD_k . Note that this notation is useful for our efficiency analysis since the encryption time of each instance of Parallel_λ and 1CT is different for each $k \in [\eta]$.

As stated in the security bound analysis, we set $\eta = \zeta^{1/2}$ to transform a $\lambda^{-\zeta}$ -secure single-key SKFE scheme. Thus, when transforming quasi-polynomially (resp. sub-exponentially) secure scheme, we set $\eta = O(\text{polylog}(\lambda))$ (resp. $\eta = O(\lambda^\gamma)$ for some positive constant $\gamma < 1$). To accomplish the analysis for HYBRD_η , it is sufficient to prove that each algorithm of $\text{Parallel}_\lambda^{(k)}$ and $1\text{CT}^{(k)}$ for every $k \in [\eta]$ used in HYBRD_η runs in polynomial time of λ, n and s . First, we estimate the running time of the encryption algorithm of $\text{Parallel}_\lambda^{(k)}$ for every $k \in [\eta]$.

Before analysis, we bound the size of the key generation circuit G and encryption circuit e defined in Figure 20 and 21, respectively. Below, let $G[f]$ denote the key generation circuit $G[f, \perp, \perp, i]$, where f is a function and i is an index. Note that which index is hardwired does not affect the size of G thus we omit writing the index. In addition, let $e[\text{Enc}]$ denote the encryption circuit computing Enc (and one PRF evaluation). From the bounds (30) and (32) that we show in Section 8.3 and 9, we can bound the size of G and e as,

$$|G[f]| \leq (|f| + n_f) \cdot \text{poly}_G(\lambda, \log q), \quad (35)$$

$$|e[\text{Enc}]| \leq |\text{Enc}| \cdot \text{poly}_e(\lambda), \quad (36)$$

where poly_G and poly_e are fixed polynomials, n_f is the input length of f , and q is the number of functional keys supported by the resulting scheme of the hybrid construction.

For every $k \in [\eta]$, let $\text{Para}.t_{\text{Enc}}^{(k)}$ be the bound of the running time of the encryption algorithm of $\text{Parallel}_\lambda^{(k)}$. Since 1Key is weakly succinct, from the bound (25) in Section 8.1, for every $k \in [\eta]$, $\text{Para}.t_{\text{Enc}}^{(k)}$ is bounded as

$$\text{Para}.t_{\text{Enc}}^{(k)} \leq |f|^\gamma \cdot \text{poly}_{\text{Para}}(\lambda, n_f), \quad (37)$$

where f is a function supported by $\text{Parallel}_\lambda^{(k)}$, n_f is the input length of f , $\gamma < 1$ is a constant, and $\text{poly}_{\text{Para}}$ is a fixed polynomial. As mentioned above, for every $k \in [\eta]$, $\text{Parallel}_\lambda^{(k)}$ denotes different instances of the same scheme Parallel_λ and thus $\text{poly}_{\text{Para}}$ is independent of k .

For every $k \in \{2, \dots, \eta\}$, $\text{Parallel}_\lambda^{(k-1)}$ has to generate a functional key tied to the circuit $G[e[\text{Para.Enc}^{(k)}]]$. Therefore, from (37), we have

$$\text{Para}.t_{\text{Enc}}^{(k-1)} \leq |G[e[\text{Para.Enc}^{(k)}]]|^\gamma \cdot \text{poly}_{\text{Para}}(\lambda, 2\lambda + 1). \quad (38)$$

Here, the input length of the circuit $e[\text{Para.Enc}^{(k)}]$ is bounded by $2(2\lambda + 1) + \lambda + |\text{Para.Enc}^{(k)}|$ since the length of a ciphertext output by $\text{Para.Enc}^{(k)}$ is bounded by the size of $\text{Para.Enc}^{(k)}$. Then, from (35) and (36), for every $k \in [\eta]$, we can estimate $|G[e[\text{Para.Enc}^{(k)}]]|$ as

$$\begin{aligned} |G[e[\text{Para.Enc}^{(k)}]]| &\leq \left(|e[\text{Para.Enc}^{(k)}]| + (2(2\lambda + 1) + \lambda + |\text{Para.Enc}^{(k)}|) \right) \cdot \text{poly}_G(\lambda, \log \lambda^k) \\ &= \left(|\text{Para.Enc}^{(k)}| \cdot \text{poly}_e(\lambda) + ((2(2\lambda + 1) + \lambda + |\text{Para.Enc}^{(k)}|)) \right) \cdot \\ &\quad \text{poly}_G(\lambda, \log \lambda^k) \\ &\leq |\text{Para.Enc}^{(k)}| \cdot \text{poly}_1(\lambda, k) \leq |\text{Para.Enc}^{(k)}| \cdot \text{poly}_1(\lambda, \eta), \end{aligned} \quad (39)$$

where poly_1 is a fixed polynomial. Thus, from (38) and (39), for every $k \in \{2, \dots, \eta\}$, we have

$$\begin{aligned} \text{Para}.t_{\text{Enc}}^{(k-1)} &\leq \left(|\text{Para.Enc}^{(k)}| \cdot \text{poly}_1(\lambda, \eta) \right)^\gamma \cdot \text{poly}_{\text{Para}}(\lambda, 2\lambda + 1) \\ &\leq |\text{Para.Enc}^{(k)}|^\gamma \cdot \text{poly}_2(\lambda, \eta) = (\text{Para}.t_{\text{Enc}}^{(k)})^\gamma \cdot \text{poly}_2(\lambda, \eta), \end{aligned} \quad (40)$$

where poly_2 is also a fixed polynomial. In the last equality, we consider $|\text{Para.Enc}^{(k)}|$ equals its running time.

In addition, $\text{Parallel}_\lambda^{(\eta)}$ has to release a functional key tied to a circuit G in which a function supported by HYBRD_η is hardwired. Therefore, by using (35) and (37) again, we can bound $\text{Para}.t_{\text{Enc}}^{(\eta)}$ as

$$\begin{aligned} \text{Para}.t_{\text{Enc}}^{(\eta)} &\leq ((s + n) \cdot \text{poly}_G(\lambda, \log \lambda^\eta))^\gamma \cdot \text{poly}_{\text{Para}}(\lambda, 2\lambda + 1) \\ &\leq s^\gamma \cdot \text{poly}_3(\lambda, n, \eta), \end{aligned} \quad (41)$$

where poly_3 is also a fixed polynomial.

From (40) and (41), for every $k \in [\eta]$, it holds that

$$\begin{aligned} \text{Para}.t_{\text{Enc}}^{(k)} &\leq (s^\gamma \cdot \text{poly}_3(\lambda, n, \eta))^{\gamma^{\eta-k}} \cdot \prod_{j=0}^{\eta-k-1} \text{poly}_2(\lambda, \eta)^{\gamma^j} \\ &\leq s^\gamma \cdot \text{poly}_3(\lambda, n, \eta) \cdot \text{poly}_2(\lambda, \eta)^{\frac{1}{1-\gamma}} \leq s^\gamma \cdot \text{poly}_4(\lambda, n, \eta), \end{aligned} \quad (42)$$

where poly_4 is a fixed polynomial. The second inequality comes from the fact $\gamma < 1$. Thus, we see that the encryption algorithm of $\text{Parallel}_\lambda^{(k)}$ for every $k \in [\eta]$ runs in polynomial of λ, n, η and s .

Next, we analyze the encryption time of $1\text{CT}^{(k)}$ for every $k \in [\eta]$. For every $k \in [\eta]$, let $1\text{CT}.t_{\text{Enc}}^{(k)}$ be the bound of the running time of the encryption algorithm of $1\text{CT}^{(k)}$. For every $k \in [\eta]$, $1\text{CT}^{(k)}$ generates a functional key tied to the circuit $e[\text{Para.Enc}^{(k)}]$. From (42), the input length of $e[\text{Para.Enc}^{(k)}]$ is bounded by

$$\begin{aligned} 2(2\lambda + 1) + \lambda + |\text{Para.Enc}^{(k)}| &\leq 2(2\lambda + 1) + \lambda + s^\gamma \cdot \text{poly}_4(\lambda, n, \eta) \\ &\leq s^\gamma \cdot \text{poly}_5(\lambda, n, \eta), \end{aligned}$$

where poly_5 is a fixed polynomial. Then, from the quasi-linear efficiency of 1CT that we show as (26) in Section 8.2, for every $k \in [\eta]$, we have

$$\begin{aligned} 1\text{CT}.t_{\text{Enc}}^{(k)} &\leq (s^\gamma \cdot \text{poly}_5(\lambda, n, \eta)) \cdot \text{poly}_{1\text{CT}}(\lambda, \log(s^\gamma \cdot \text{poly}_5(\lambda, n, \eta))) \\ &\leq s^{\gamma'} \cdot \text{poly}_6(\lambda, n, \eta), \end{aligned} \quad (43)$$

where $\text{poly}_{1\text{CT}}$ and poly_6 are fixed polynomials and γ' is a constant such that $\gamma < \gamma' < 1$. Therefore, for $k \in [\eta]$, we can see that the encryption algorithm of $1\text{CT}^{(k)}$ runs in polynomial of λ, n, η , and s .

From (42) and (43), the encryption time of HYBRD_η is bounded by

$$\eta \cdot \left(s^\gamma \cdot \text{poly}_5(\lambda, n, \eta) + s^{\gamma'} \cdot \text{poly}_6(\lambda, n, \eta) \right) \leq s^{\gamma'} \cdot \text{poly}_7(\lambda, n, \eta), \quad (44)$$

where poly_7 is a fixed polynomial.

As stated earlier, η is at most sub-linear in λ in the construction. Thus, (44) means that the encryption algorithm of HYBRD_η runs in polynomial time of λ, n and s . In this case, we can see that all of algorithms of HYBRD_η runs in polynomial of λ, n and s . Moreover, (44) means that HYBRD_η is weakly succinct.¹⁷

From these analysis, HYBRD_η is $\delta = \lambda^{-\zeta}$ -secure and supports $\lambda^\eta = \lambda^{\zeta^{1/2}}$ functional keys. More formally, HYBRD_η is $(\lambda^{\zeta^{1/2}}, \delta)$ -selective-message function private iSKFE. Since $\zeta = \omega(1)$, $\lambda^{\zeta^{1/2}}$ is super-polynomial. Therefore, HYBRD_η is (poly, δ)-selective-message function private. \square (**Theorem 10.2**)

10.4 Converting iSKFE into SKFE

Finally, we show how to transform iSKFE into SKFE. We provide the overview of this transformation in Remark 3.19 in Section 3.2. Here, we give the formal description of the transformation and prove its security. Let iSKFE = (Setup, iKG, Enc, Dec) be an iSKFE scheme whose index space is \mathcal{I} . We construct an SKFE scheme SKFE = (Setup', KG, Enc', Dec') as follows. Setup', Enc', and Dec' are exactly the same as Setup, Enc, and Dec, respectively. We define KG as follows.

KG(MSK, f) :

- Generate $i \xleftarrow{\mathcal{I}}$ and return $sk_f \leftarrow \text{iKG}(\text{MSK}, f, i)$.

The message and function spaces of SKFE is the same as those of iSKFE. The correctness of SKFE directly follows from that of iSKFE. Then, we have the following theorem.

Theorem 10.3. *Let iSKFE be (poly, δ)-selective-message function private iSKFE the size of whose index space is S . Then, SKFE is (poly, δ')-selective-message function private SKFE, where $\delta' = \frac{1}{S} + \delta$.*

Proof of Theorem 10.3. We assume that the advantage of any adversary attacking iSKFE is bounded by ϵ . Let \mathcal{A} be an adversary that attacks the selective-message function privacy of SKFE. We assume that \mathcal{A} makes q key queries at most, where q is a polynomial of λ and $q \leq S$. Then, we have

$$\text{Adv}_{\text{SKFE}, \mathcal{A}}^{\text{sm-fp}}(\lambda) \leq 2 \left(\frac{q(q-1)}{S} + \frac{\epsilon}{2} \right). \quad (45)$$

This means that if iSKFE is δ -secure, then SKFE is δ' secure, where $\delta' = \frac{1}{S} + \delta$. Below, we prove the above inequality (45). First, consider the following sequence of games.

Game 0 This is the original selective-message function privacy game regarding SKFE.

Game 1 Same as Game 0 except how the challenger responds to key queries made by \mathcal{A} . At the initialization step, the challenger prepares a list \mathcal{L} . The challenger initializes \mathcal{L} to the empty set at the beginning. The list \mathcal{L} will stores indices during the game. When \mathcal{A} sends a function f as a key query, the challenger first generates an index $i \xleftarrow{\mathcal{I}}$, and checks whether the list \mathcal{L} contains the index i or not. If so, the challenger generates $i' \xleftarrow{\mathcal{I} \setminus \mathcal{L}}$ and returns $sk_f \leftarrow \text{iKG}(\text{MSK}, f, i')$ to \mathcal{A} . Otherwise, the challenger returns $sk_f \leftarrow \text{iKG}(\text{MSK}, f, i)$ to \mathcal{A} and adds i to \mathcal{L} .

¹⁷ Analogously, we see that if the underlying single-key SKFE is succinct, then so does HYBRD_η .

Let SUC_0 (resp. SUC_1) be the event that \mathcal{A} succeeds in guessing the challenge bit b in Game 0 (resp. Game 1). Then, we can estimate the advantage of \mathcal{A} as

$$\begin{aligned} \frac{1}{2} \cdot \text{Adv}_{\text{PRDCT}, \mathcal{A}}^{\text{sm-fp}}(\lambda) &= \left| \Pr[\text{SUC}_0] - \frac{1}{2} \right| \\ &\leq \left| \Pr[\text{SUC}_0] - \Pr[\text{SUC}_1] \right| + \left| \Pr[\text{SUC}_1] - \frac{1}{2} \right|. \end{aligned} \quad (46)$$

Below, we estimate each term on the right side of the inequality (46).

Lemma 10.4. $\left| \Pr[\text{SUC}_0] - \Pr[\text{SUC}_1] \right| \leq \frac{q(q-1)}{S}$.

Proof of Lemma 10.4. Let Collision be the event that the same index is generated by the challenger in order to respond to key queries made by \mathcal{A} . Note that Game 0 and 1 are exactly the same game unless the event Collision occurs. Therefore, we have $\left| \Pr[\text{SUC}_0] - \Pr[\text{SUC}_1] \right| \leq \Pr[\text{Collision}]$. In addition, $\Pr[\text{Collision}]$ is bounded by $\frac{q(q-1)}{S}$ using the union bound. Thus, the lemma holds. \square (**Lemma 10.4**)

Lemma 10.5. $\left| \Pr[\text{SUC}_1] - \frac{1}{2} \right| \leq \frac{\epsilon}{2}$.

Proof of Lemma 10.5. In Game 1, every key query made by \mathcal{A} is replied using a different index in \mathcal{I} . Therefore, we can construct an adversary that attacks iSKFE and whose advantage is the same as that of \mathcal{A} in Game 1, that is $2 \left| \Pr[\text{SUC}_1] - \frac{1}{2} \right|$. Thus, $\left| \Pr[\text{SUC}_1] - \frac{1}{2} \right| \leq \frac{\epsilon}{2}$ \square (**Lemma 10.5**)

From the inequality (46) and Lemmas 10.4 and 10.5, inequality (45) holds. \square (**Theorem 10.3**)

10.5 From Single-Key SKFE to Collusion-Resistant SKFE

We combine the results that we proved though Section 10 to give our main result. First, we combine Theorem 10.2 and 10.3, and obtain the following main theorem.

Theorem 10.6. *Assuming there exists $(1, \delta)$ -selective-message function private SKFE for P/poly that is weakly succinct, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, there exists (poly, δ') -selective-message function private SKFE for P/poly , where $\delta'(\lambda) = \lambda^{-\zeta^{1/2}}$.¹⁸*

Theorem 10.6 states that if the underlying single-key scheme is sub-exponentially secure, then so is the resulting scheme. Formally, we have the following theorem.

Theorem 10.7. *Assuming there exists $(1, \delta)$ -selective-message function private SKFE for P/poly that is weakly succinct, where $\delta(\lambda) = 2^{-\lambda^\gamma}$ and $\gamma < 1$ is a constant. Then, there exists (poly, δ') -selective-message function private SKFE for P/poly , where $\delta'(\lambda) = 2^{-\lambda^{\gamma/2}}$.*

Therefore, by combining Theorem 10.7 with the result by Kitagawa et al. [KNT17b], we have the following corollary.

Corollary 10.8. *Assuming there exist sub-exponentially secure single-key weakly succinct SKFE for P/poly . Then, there exists IO for P/poly .*

¹⁸We can slightly generalize the result. By setting $\eta = \zeta^{1/c}$ in the construction for any constant $c > 1$, we can achieve $\delta'(\lambda) = \lambda^{-\zeta^{1/c}}$.

11 Upgrading Succinctness and Security of SKFE

We can upgrade succinctness and security of SKFE with polynomial security loss. More specifically,

1. We can transform weakly succinct SKFE into succinct one with polynomial security loss.
2. We can transform selective-message-function message private SKFE scheme into selective-message function private one with polynomial security loss.

By accommodating these upgrades into our main results, that is Theorem 10.6 and 10.7, we obtain the following corollary.

Corollary 11.1. *If there exists quasi-polynomially (resp. sub-exponentially) secure single-key SKFE that is selective-message-function message private and weakly succinct, there exists quasi-polynomially (resp. sub-exponentially) secure collusion-resistant SKFE that is selective-message function private and succinct.*

We give details of the above upgrades in subsequent sections.

11.1 Transforming Weakly Succinct SKFE into Succinct One

Ananth et al. [AJS15] proved that we can transform a collusion-resistant SKFE scheme into a succinct one. By using this result and Theorem 10.6, we can construct succinct SKFE based on weakly succinct one. However, the construction incurs at least quasi-polynomial security loss due to the security loss of our result.

Achieving succinctness with polynomial security loss. In fact, we can transform weakly succinct SKFE into succinct one with only polynomial security loss by using our transformation in Section 10.2 for $\eta = O(1)$.

By setting $\eta = O(1)$ in the construction of HYBRD_η , from the inequality (34), we can construct an iSKFE scheme supporting λ^η functional keys with polynomial security loss. Let $q = \lambda^\eta$. Then, from the inequality (44), the encryption time of HYBRD_η depends on q only in poly-logarithmic. Namely, HYBRD_η is collusion succinct. Formally, we obtain the following theorem from the analysis in Section 10.3.

Theorem 11.2. *Assuming there exists a $(1, \delta)$ -selective-message function private SKFE scheme that is weakly succinct. Then, there exists a (q, δ) -selective-message function private iSKFE scheme that is collusion succinct, where q is a fixed polynomial of λ .*

Such a collusion-succinct scheme can be transformed into a single-key succinct scheme by the technique utilizing decomposable randomized encoding used by Bitansky and Vaikuntanathan [BV15, Proposition IV.1]. This fact was observed by Kitagawa et al. [KNT21]. They proposed the transformation for a PKFE scheme, but we can easily observe that it works even if the building block scheme is an iSKFE scheme. That is, the following theorem is known.

Theorem 11.3 ([BV18, KNT21, Adapted]). *Assuming there exists a (q, δ) -selective-message function private SKFE scheme that is collusion succinct, where q is a fixed polynomial of λ . Then, there exists a $(1, \delta)$ -selective-message function private iSKFE scheme that is succinct.*

Note that the resulting scheme is a single-key iSKFE scheme and thus is also a single-key SKFE scheme from the discussion in Remark 3.19. From Theorems 11.2 and 11.3, the following theorem holds.

Theorem 11.4. *Assuming there exists a $(1, \delta)$ -selective-message function private SKFE scheme that is weakly succinct. Then, there exists a $(1, \delta)$ -selective-message function private SKFE scheme that is succinct.*

11.2 Transforming Selective-message-function secure SKFE into Selective-secure One

We can transform a selective-message-function message private SKFE scheme into a selective-message function private one.

Theorem 11.5. *If there exists a $(1, \delta)$ -selective-message-function message private SKFE scheme that is weakly succinct, there exists a $(1, \delta)$ -selective-message function private SKFE scheme that is weakly succinct.*

In fact, this theorem is easily obtained by known facts. We introduce the following theorem stating that we can transform selective-message-function message private SKFE into selective-message message private one.

Theorem 11.6 ([KNT21]). *If there exists a $(1, \delta)$ -selective-message-function message private SKFE scheme that is weakly succinct, there exists a $(1, \delta)$ -selective-message message private SKFE scheme that is weakly succinct.*

We obtain Theorem 11.6 by the result of Kitagawa et al. Their construction does not directly use underlying SKFE and first transform it into strong exponentially-efficient IO (SXIO). SXIO that is sufficient for their construction can be based on single-key weakly succinct SKFE [BNPW20]. In the construction of SXIO, we can observe that it is sufficient that the underlying SKFE satisfies selective-message-function message privacy though this fact is not explicitly stated. Thus, we obtain Theorem 11.6.

In addition, by Theorem 3.14 shown by Brakerski and Segev [BS18], we can transform a selective-message message private scheme into a selective-message function private one. They do not refer to succinctness in their paper, but we observe that their transformation preserves (weak) succinctness.

These two transformations incur only polynomial security loss. Thus, we can transform single-key selective-message-function message private SKFE into single-key selective-message function private one with polynomial security loss.

References

- [AB15] Benny Applebaum and Zvika Brakerski. Obfuscating circuits via composite-order graded encoding. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 528–556. Springer, Heidelberg, March 2015. (Cited on page 2.)
- [ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015. (Cited on page 18, 19, 25, 62.)
- [ADGM17] Daniel Apon, Nico Döttling, Sanjam Garg, and Pratyay Mukherjee. Cryptanalysis of indistinguishability obfuscations of circuits over GGH13. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP 2017*, volume 80 of *LIPICs*, pages 38:1–38:16. Schloss Dagstuhl, July 2017. (Cited on page 2.)
- [AGIS14] Prabhanjan Vijendra Ananth, Divya Gupta, Yuval Ishai, and Amit Sahai. Optimizing obfuscation: Avoiding Barrington’s theorem. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 646–658. ACM Press, November 2014. (Cited on page 2.)
- [Agr17] Shweta Agrawal. Stronger security for reusable garbled circuits, general definitions and attacks. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 3–35. Springer, Heidelberg, August 2017. (Cited on page 5.)

- [Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019. (Cited on page 2.)
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006. (Cited on page 23.)
- [AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015. (Cited on page 2, 4, 7, 17.)
- [AJL⁺19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019. (Cited on page 2.)
- [AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. <https://eprint.iacr.org/2015/730>. (Cited on page 2, 5, 6, 11, 14, 40, 82.)
- [AJS18] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. <https://eprint.iacr.org/2018/615>. (Cited on page 2.)
- [AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205. Springer, Heidelberg, November 2017. (Cited on page 5.)
- [AS16] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM Journal on Computing*, 45(6):2117–2176, 2016. (Cited on page 3, 4.)
- [AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017. (Cited on page 2.)
- [AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, December 2019. (Cited on page 5.)
- [BCG⁺18] Nir Bitansky, Ran Canetti, Sanjam Garg, Justin Holmgren, Abhishek Jain, Huijia Lin, Rafael Pass, Sidharth Telang, and Vinod Vaikuntanathan. Indistinguishability obfuscation for RAM programs and succinct randomized encodings. *SIAM Journal on Computing*, 47(3):1123–1210, 2018. (Cited on page 2.)
- [BDGM20a] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Candidate iO from homomorphic encryption schemes. In Anne Canteaut and Yuval Ishai, editors,

EUROCRYPT 2020, Part I, volume 12105 of *LNCS*, pages 79–109. Springer, Heidelberg, May 2020. (Cited on page 2.)

- [BDGM20b] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for iO: Circular-secure LWE suffices. *Cryptology ePrint Archive*, Report 2020/1024, 2020. <https://eprint.iacr.org/2020/1024>. (Cited on page 2.)
- [BGI⁺12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 2, 9.)
- [BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 501–519. Springer, Heidelberg, March 2014. (Cited on page 21.)
- [BGK⁺14] Boaz Barak, Sanjam Garg, Yael Tauman Kalai, Omer Paneth, and Amit Sahai. Protecting obfuscation against algebraic attacks. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 221–238. Springer, Heidelberg, May 2014. (Cited on page 2.)
- [BGMS15] Dan Boneh, Divya Gupta, Ilya Mironov, and Amit Sahai. Hosting services on an untrusted cloud. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 404–436. Springer, Heidelberg, April 2015. (Cited on page 2.)
- [BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018. (Cited on page 2.)
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 784–796. ACM Press, October 2012. (Cited on page 22, 60.)
- [BKS16] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 852–880. Springer, Heidelberg, May 2016. (Cited on page 18.)
- [BKS18] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. *Journal of Cryptology*, 31(2):434–520, April 2018. (Cited on page 3, 4, 12, 17, 67.)
- [BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 764–791. Springer, Heidelberg, May 2016. (Cited on page 2.)
- [BNPW20] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. *Journal of Cryptology*, 33(2):357–405, April 2020. (Cited on page 2, 3, 4, 11, 12, 27, 34, 36, 83.)
- [BPR⁺08] Dan Boneh, Periklis A. Papakonstantinou, Charles Rackoff, Yevgeniy Vahlis, and Brent Waters. On the impossibility of basing identity based encryption on trapdoor permutations.

- In *49th FOCS*, pages 283–292. IEEE Computer Society Press, October 2008. (Cited on page 4.)
- [BPW16] Nir Bitansky, Omer Paneth, and Daniel Wichs. Perfect structure on the edge of chaos - trapdoor permutations from indistinguishability obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 474–502. Springer, Heidelberg, January 2016. (Cited on page 2.)
- [BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 1–25. Springer, Heidelberg, February 2014. (Cited on page 2.)
- [BS18] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *Journal of Cryptology*, 31(1):202–225, January 2018. (Cited on page 8, 17, 24, 25, 31, 83.)
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. (Cited on page 2.)
- [BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015. (Cited on page 3, 4, 5, 6, 7, 9, 10, 14, 16, 17, 28, 30, 31, 40, 44, 46, 82.)
- [BV18] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *Journal of the ACM*, 65(6):39:1–39:37, 2018. (Cited on page 2, 3, 4, 5, 6, 7, 11, 14, 16, 17, 28, 31, 82.)
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013. (Cited on page 21.)
- [CGH⁺15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancreède Lepoint, Hemanta K. Maji, Eric Miles, Mariana Raykova, Amit Sahai, and Mehdi Tibouchi. Zeroizing without low-level zeroes: New MMAP attacks and their limitations. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 247–266. Springer, Heidelberg, August 2015. (Cited on page 2.)
- [CGH17] Yilei Chen, Craig Gentry, and Shai Halevi. Cryptanalyses of candidate branching program obfuscators. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 278–307. Springer, Heidelberg, April / May 2017. (Cited on page 2.)
- [CHN⁺18] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. *SIAM Journal on Computing*, 47(6):2157–2202, 2018. (Cited on page 2.)
- [CHVW19] Yilei Chen, Minki Hhan, Vinod Vaikuntanathan, and Hoeteck Wee. Matrix PRFs: Constructions, attacks, and applications to obfuscation. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 55–80. Springer, Heidelberg, December 2019. (Cited on page 2.)

- [CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrede Lepoint, and Mehdi Tibouchi. Zeroizing attacks on indistinguishability obfuscation over CLT13. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 41–58. Springer, Heidelberg, March 2017. (Cited on page 2.)
- [CLTV15] Ran Canetti, Huijia Lin, Stefano Tessaro, and Vinod Vaikuntanathan. Obfuscation of probabilistic circuits and applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 468–497. Springer, Heidelberg, March 2015. (Cited on page 9, 15.)
- [CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 577–607. Springer, Heidelberg, August 2018. (Cited on page 2.)
- [FRS17] Rex Fernando, Peter M. R. Rasmussen, and Amit Sahai. Preventing CLT attacks on obfuscation with linear overhead. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 242–271. Springer, Heidelberg, December 2017. (Cited on page 2.)
- [GGG⁺14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014. (Cited on page 7, 17.)
- [GGH15] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 498–527. Springer, Heidelberg, March 2015. (Cited on page 2.)
- [GGH⁺16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. (Cited on page 2, 5.)
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. (Cited on page 21.)
- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 555–564. ACM Press, June 2013. (Cited on page 5.)
- [GMM⁺16] Sanjam Garg, Eric Miles, Pratyay Mukherjee, Amit Sahai, Akshayaram Srinivasan, and Mark Zhandry. Secure obfuscation in a weak multilinear map model. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 241–268. Springer, Heidelberg, October / November 2016. (Cited on page 2.)
- [GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Heidelberg, August 2017. (Cited on page 4.)
- [GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM*

SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021, pages 736–749. ACM, 2021. (Cited on page 2.)

- [GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfuscation. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 156–181. Springer, Heidelberg, April / May 2017. (Cited on page 4.)
- [GS16] Sanjam Garg and Akshayaram Srinivasan. Single-key to multi-key functional encryption with polynomial loss. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 419–442. Springer, Heidelberg, October / November 2016. (Cited on page 4, 5, 17.)
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 5.)
- [HJK⁺16] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. How to generate and use universal samplers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 715–744. Springer, Heidelberg, December 2016. (Cited on page 2.)
- [HJL21] Samuel B. Hopkins, Aayush Jain, and Huijia Lin. Counterexamples to new circular security assumptions underlying $i\mathcal{O}$. *IACR Cryptol. ePrint Arch.*, 2021:889, 2021. CRYPTO 2021 (to appear). (Cited on page 2.)
- [HSW14] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 201–220. Springer, Heidelberg, May 2014. (Cited on page 2.)
- [Imp95] Russell 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*, pages 134–147. IEEE Computer Society, 1995. (Cited on page 4.)
- [IR89] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *21st ACM STOC*, pages 44–61. ACM Press, May 1989. (Cited on page 4.)
- [JLMS19] Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials over \mathbb{R} to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019. (Cited on page 2.)
- [JLS19] Aayush Jain, Huijia Lin, and Amit Sahai. Simplifying constructions and assumptions for $i\mathcal{O}$. Cryptology ePrint Archive, Report 2019/1252, 2019. <https://eprint.iacr.org/2019/1252>. (Cited on page 2.)
- [JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. (Cited on page 2.)

- [KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 419–428. ACM Press, June 2015. (Cited on page 2.)
- [KMN⁺14] Ilan Komargodski, Tal Moran, Moni Naor, Rafael Pass, Alon Rosen, and Eylon Yogev. One-way functions and (im)perfect obfuscation. In *55th FOCS*, pages 374–383. IEEE Computer Society Press, October 2014. (Cited on page 4.)
- [KNT17a] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. From single-key to collusion-resistant secret-key functional encryption by leveraging succinctness. Cryptology ePrint Archive, Report 2017/638, 2017. <https://eprint.iacr.org/2017/638>. (Cited on page 4, 58.)
- [KNT17b] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. Cryptology ePrint Archive, Report 2017/361, 2017. <https://eprint.iacr.org/2017/361>. (Cited on page 81.)
- [KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Obustopia built on secret-key functional encryption. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 603–648. Springer, Heidelberg, April / May 2018. (Cited on page 1.)
- [KNT21] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. *J. Cryptol.*, 34(3):25, 2021. (Cited on page 11, 12, 34, 82, 83.)
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 669–684. ACM Press, November 2013. (Cited on page 21.)
- [KS20] Ilan Komargodski and Gil Segev. From minicrypt to obustopia via private-key functional encryption. *Journal of Cryptology*, 33(2):406–458, April 2020. (Cited on page 3, 4, 5, 44.)
- [KY09] Jonathan Katz and Arkady Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 197–213. Springer, Heidelberg, December 2009. (Cited on page 4.)
- [Lin16] Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 28–57. Springer, Heidelberg, May 2016. (Cited on page 2.)
- [Lin17] Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017. (Cited on page 2.)
- [LM16] Baiyu Li and Daniele Micciancio. Compactness vs collusion resistance in functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 443–468. Springer, Heidelberg, October / November 2016. (Cited on page 4, 5, 7, 17, 20, 25, 57, 58, 67, 76.)

- [LM18] Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. Cryptology ePrint Archive, Report 2018/646, 2018. <https://eprint.iacr.org/2018/646>. (Cited on page 2.)
- [LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *J. Cryptology*, 22(2):161–188, 2009. (Cited on page 22.)
- [LPST16] Huijia Lin, Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation with non-trivial efficiency. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 447–462. Springer, Heidelberg, March 2016. (Cited on page 2, 12, 34.)
- [LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988. (Cited on page 21.)
- [LV16] Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016. (Cited on page 2.)
- [MSZ16] Eric Miles, Amit Sahai, and Mark Zhandry. Annihilation attacks for multilinear maps: Cryptanalysis of indistinguishability obfuscation over GGH13. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 629–658. Springer, Heidelberg, August 2016. (Cited on page 2.)
- [MZ18] Fermi Ma and Mark Zhandry. The MMap strikes back: Obfuscation and new multilinear maps immune to CLT13 zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 513–543. Springer, Heidelberg, November 2018. (Cited on page 2.)
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. <https://eprint.iacr.org/2010/556>. (Cited on page 2.)
- [PST14] Rafael Pass, Karn Seth, and Sidharth Telang. Indistinguishability obfuscation from semantically-secure multilinear encodings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 500–517. Springer, Heidelberg, August 2014. (Cited on page 2.)
- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. (Cited on page 5, 10, 19, 31, 59.)
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. (Cited on page 2.)
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *46th ACM STOC*, pages 475–484. ACM Press, May / June 2014. (Cited on page 2, 6, 13.)
- [Val76] Leslie G. Valiant. Universal circuits (preliminary report). In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *Proceedings of the 8th Annual ACM Symposium on Theory of Computing, May 3-5, 1976, Hershey, Pennsylvania, USA*, pages 196–203. ACM, 1976. (Cited on page 56.)

- [Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015. (Cited on page 5.)
- [WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, volume 12698 of *Lecture Notes in Computer Science*, pages 127–156. Springer, 2021. (Cited on page 2.)
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 18, 20, 22, 23, 60.)
- [Zim15] Joe Zimmerman. How to obfuscate programs directly. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 439–467. Springer, Heidelberg, April 2015. (Cited on page 2.)