T2R2 東京科学大学 リサーチリポジトリ
Science Tokyo Research Repository

論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Cryptographic Obfuscation Based on Secret Key Primitives
著者(和文)	北川冬航
Author(English)	Fuyuki Kitagawa
出典(和文)	学位:博士(理学), 学位授与機関:東京工業大学, 報告番号:甲第11063号, 授与年月日:2019年3月26日, 学位の種別:課程博士, 審査員:田中 圭介,伊藤 利昭,尾形 わかは,鹿島 亮,森 立平,藤崎 英一郎
Citation(English)	Degree:Doctor (Science), Conferring organization: Tokyo Institute of Technology, Report number:甲第11063号, Conferred date:2019/3/26, Degree Type:Course doctor, Examiner:,,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Powered by T2R2 (Science Tokyo Research Repository)

Cryptographic Obfuscation Based on Secret Key Primitives

Fuyuki Kitagawa
Supervisor: Keisuke Tanaka

Department of Mathematical and Computing Science
Tokyo Institute of Technology

February 27, 2019

謝辞

本学位論文は私が東京工業大学 情報理工学院 数理計算科学系 田中圭介研究室に在籍している間に得た研究成果の一部をまとめたものです。同研究室に在籍した六年間に渡り、多くのご支援とご指導を賜りました田中圭介教授に、深く感謝します。田中先生は研究内容だけに留まらず、研究者としてのあり方など非常に多くのことを教えて下さいました。私は同研究室に在籍しなければ、研究者を志すことも学位を取得することもなかったと思います。

また本論文の審査員を引き受けて下さり、多くのご指導を頂きました伊東利哉教授、尾形わかは教授、鹿島亮准教授、森立平助教授、藤崎英一郎特定教授に感謝致します。

本論文の研究に関し多くの助言を下さった NTT セキュアプラットフォーム研究所の西巻陵氏に深謝します。私は博士課程一年目に同研究所をインターンシップで訪問しており、その際に受け入れて下さったのが西巻さんでした。また、その際に取り組んだ研究が本学位論文のテーマである秘密鍵関数型暗号及び暗号学的難読化です。私が同研究所に滞在した二ヶ月間、西巻さんは毎日私との研究に時間を費やして下さり、またインターンシップ終了後も引き続き、私との研究に非常に多くの時間を割いて下さいました。

修士課程の頃より参加させて頂いた新明るい暗号勉強会の皆様にも感謝致します。特に産業技術総合研究所 サイバーフィジカルセキュリティ研究センターの松田隆宏氏には、私が修士課程の頃から継続して議論に多くの時間を割いて頂き、ご指導を頂きましたこと深謝します。

また私が在籍した田中圭介研究室の方々、及び修了生の皆様にも感謝します。特に私の博士課程在籍中に共に博士課程に在籍し研究に励んだ、石田愛氏と Yuyu Wang 氏には深く感謝しております。

工学院 電気電子系 宮本恭幸教授を始めとする、本学水泳部員及びその出身者から構成される燕水会の方々にも深く感謝申し上げます。燕水会の方々から頂いた励ましは、博士課程に進学し学位を取得する上で非常に私の支えとなりました。

最後に、私を支え続けてくれた家族に感謝致します。家族の支えがなければ博士課程を修了することはできませんでした。深く感謝しております。

Contents

Chapter 1

Introduction

1.1 Background

1.1.1 Cryptographic Obfuscation

Over the past three decades, cryptographic community has given beautiful solutions to many cryptographic tasks and problems. As a result, we now have many cryptographic tools such as encryption, signature, authentication, protocols, and so on. However, there are still several important cryptographic problems for which we still do not have a solution developed enough. Program obfuscation is one of such cryptographic problems.

Program obfuscation aims to turn programs "unintelligible" while preserving its functionality. The goal of program obfuscation is to protect information of program codes against reverse engineerings. There are many heuristic approaches to program obfuscation in practice. However, most of such practical attempts had been broken. This fact motivates us to realize program obfuscation whose security is guaranteed in the perspective of cryptography. Program obfuscation is now one of the central topics in cryptography.

The theoretical study of cryptographic program obfuscation was initiated by Barak, Goldreich, Impagliazzo, Rudich, Sahai, Vadhan, and Yang [BGI+01]. 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+13] proposed the first candidate construction of IO for all circuits. Subsequently, many works have shown that IO is powerful enough in the sense that we can achieve a wide variety of cryptographic primitives based on IO though it is weaker than virtual-black-box obfuscation [GGH+13,

SW14, HSW14, BGMS15, KLW15, BGL+15, CHJV15, BPW16, CHN+16, HJK+16].

1.1.2 IO Based on Secret-Key Primitives

While we know the usefulness of IO well, we know very little about how to achieve IO. Although the first candidate construction was demonstrated, we are still at the embryonic stage for constructing IO. All known constructions of IO are based on a little-studied cryptographic tool called multi-linear maps [GGH+13, BGK+14, BR14, AGIS14, PST14, Zim15, AB15, BMSZ16, GMM+16, Lin16, LV16, FRS16, AS17, Lin17]. Moreover, security flaws were discovered in some IO constructions [CGH+15, MSZ16, ADGM17, CLLT17, CGH17].

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

Functional encryption is one of the most advanced cryptographic primitives which enables 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 x except $f(m)$ is revealed from the ciphertext of m. This feature enables us to construct a cryptographic system with fine-grained access control. In addition, it is known that functional encryption is a versatile building block to construct other cryptographic primitives. In particular, we can construct IO for all circuits by using functional encryption that satisfies certain security notions and efficiency requirements [AJ15, BV15, AJS15, BNPW16].

Bitansky and Vaikuntanathan [BV15] and Ananth and Jain [AJ15] independently showed that we can construct IO based on single-key public-key functional encryption (PKFE) which satisfies an efficiency property called weak-succinctness. A functional encryption scheme that supports a single functional decryption key is called a *single-key* scheme. Moreover, we say that a functional encryption scheme is *succinct* if the size of its encryption circuit is independent of the size of functions. *Weak-succinctness* is a relaxed variant of succinctness that allows the size of the encryption circuit to depend on the size of functions if the dependency is only sub-linear.

Bitansky, Nishimaki, Passelègue, and Wichs [BNPW16] subsequently showed that *collusion-resistant* secret-key functional encryption (SKFE) is powerful enough to yield IO if we additionally assume plain public-key encryption. Collusion-resistant functional encryption is functional encryption that can securely issue a-priori unbounded number of

functional keys.

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

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

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

The real power of IO appears in the fact that it can transform secret-key primitives into public-key ones. Therefore, solving the above problem is a key advancement to discover the exact requirements for achieving IO. Moreover, solving the above question makes a progress on the theoretical study of cryptographic primitives as we note below.

1.1.3 Impacts on The Hierarchy of Cryptographic Primitives

It is known that we can classify cryptographic primitives into two hierarchies Minicrypt and Cryptomania since the beautiful work of Impagliazzo and Rudich [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 have recently started to consider a new hierarchy called Obfustopia. Garg, Pandey, Srinivasan, and Zhandry [GPSZ17] introduced the term Obfustopia, which seems to indicate the "world" where there exists IO. Garg et al. did not give a formal definition of Obfustopia. We explicitly define Obfustopia as the "world" where there exists efficient IO for all circuits and one-way functions.[1] It is known that we can construct almost all existing cryptographic primitives which are stronger than public-key encryption by using IO. This is the reason why we consider the new hierarchy beyond Cryptomania.[2]

The landscape of Obfustopia is not clear while those of Minicrypt and Cryptomania are. In particular, we do not know how to construct IO based on standard

[1] Komargodski, Moran, Naor, Pass, Rosen, and Yogev [KMN+14] proved that IO implies one-way functions under a mild complexity theoretic assumption. More specifically, the complexity assumption is $\mathsf{NP} \not\subseteq \mathsf{io\text{-}BPP}$, where $\mathsf{io\text{-}BPP}$ is the class of languages that is decided by probabilistic polynomial-time algorithms for infinitely many input sizes. Therefore, under the assumption, we say that Obfustopia is the complexity spectrum where efficient IO for all circuits exists.

[2] Strictly speaking, it was known that there are stronger primitives than public-key encryption before the candidate of obfuscation appeared. For example, public-key encryption does not imply identity-based encryption [BPR+08].

assumptions. There has been significant effort to find out cryptographic primitives that are in Obfustopia. That is, we have been asking what kind of cryptographic primitive implies the existence of IO. As said before, we know that sub-exponentially secure succinct PKFE exists in Obfustopia [BV15, AJ15].

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 [AS15] gave a somewhat negative answer to this question. They showed that SKFE is unlikely to imply IO as long as we use black-box techniques. They also showed that SKFE does not imply any primitive in Cryptomania via black-box reductions. Moreover, it was not known whether SKFE implies any primitive outside Minicrypt even if we use it in a non-black-box manner before the work of Bitansky et al. [BNPW16].

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

Komargodski and Segev [KS17] showed that quasi-polynomially secure IO for circuits of sub-polynomial size with input of poly-logarithmic length can be constructed from quasi-polynomially secure collusion-resistant SKFE for all circuits. In addition, they showed that by combining quasi-polynomially secure collusion-resistant SKFE and sub-exponentially secure one-way functions, we can construct quasi-polynomially secure succinct PKFE. However, in this construction, the resulting PKFE supports only circuits of sub-polynomial size with input of poly-logarithmic length though the building block SKFE supports all polynomial size circuits.

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

1.2 Our Results

We show the following results that give an affirmative answer to the question above.

1.2.1 IO for All Circuits from Collusion-Resistant SKFE

We prove the following theorem.

Theorem 1.2.1 (Informal) *Assuming there exists sub-exponentially secure collusion-resistant SKFE for all circuits. Then, there exists IO for all circuits.*

Since our construction of IO is *non-black-box*, we can circumvent the impossibility result shown by Asharov and Segev [AS15].

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 result shown by Komargodski and Segev [KS17]. 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 1.2.1 by using *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 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 1.2.2 (Informal) *Assuming there exists collusion-resistant SKFE for all circuits. Then, there exists single-key weakly-succinct puncturable SKFE for all circuits.*

Note that our definition of puncturable SKFE is slightly different from that proposed by Bitansky and Vaikuntanathan. Our requirement for puncturable SKFE looks weaker than that of Bitansky and Vaikuntanathan. However, they are actually incomparable. In fact, we show that puncturable SKFE defined in this work is also sufficient for a building block of IO. See Section 3.2 for the details of the notion of puncturable SKFE and the difference between our definition and that of Bitansky and Vaikuntanathan.

We believe that the above results make a progress on the theoretical study of IO and functional encryption. Through our transformation, we can construct sub-exponentially secure IO for all circuits from sub-exponentially secure collusion-resistant SKFE for all circuits by setting security parameter appropriately. This result means that sub-exponentially secure collusion-resistant SKFE exists in Obfustopia. In addition, by combining this result and the result by Garg et al. [GGH+13], 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.

1.2.2 Collusion-Resistant SKFE from Succinct SKFE

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

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

Such a transformation is already known for PKFE. Ananth, Jain, and Sahai [AJS15] showed how to construct collusion-resistant and succinct PKFE from collusion-resistant one. 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. Their transformations preserve succinctness of the building block scheme.[3] From these results, collusion-resistance and succinctness are equivalent for PKFE.

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

As stated above, some recent results including Theorem 1.2.1 show that SKFE is a strong cryptographic primitive beyond Minicrypt if we consider non-black-box reductions. However, all of those results assume collusion-resistant SKFE as a building block. Thus, while we see that collusion-resistant SKFE is outside Minicrypt, it is still open whether succinct SKFE is also a strong cryptographic primitive beyond Minicrypt since we do not know how to construct collusion-resistant SKFE from succinct one.

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

Based on the above motivation, we show the following result.

Theorem 1.2.3 (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.*

[3] 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.

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.[4]

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 1.2.3, we discover that the existence of collusion-resistant SKFE and that of succinct one are actually equivalent if we allow quasi-polynomial security loss. Due to this equivalence, we see that succinct SKFE is also a strong cryptographic primitive beyond MINICRYPT similarly to collusion-resistant SKFE. Especially, we obtain the following corollary from Theorem 1.2.1 and 1.2.3.

Corollary 1.2.1 (Informal) *Assume that there exists sub-exponentially secure single-key weakly-succinct SKFE for all circuits. Then, there exists IO for all circuits.*

From this result, we can remove the learning with errors (LWE) assumption from recent state-of-the-art constructions of IO based on multi-linear maps and (block-wise) local pseudorandom generators [Lin17, LT17].

These works first construct single-key weakly-succinct *SKFE* based on multi-linear maps and (block-wise) local pseudorandom generators. Then, assuming the LWE assumption, they transform it into IO using the result by Bitansky et al. [BNPW16]. By relying on Corollary 1.2.1 instead of the result by Bitansky et al. [BNPW16] in their construction, we can obtain IO based only on multi-linear maps and (block-wise) local pseudorandom generators.

Additional feature of our transformation. While the above result stated in Theorem 1.2.3 incurs quasi-polynomial security loss, our transformation technique used to obtain it also leads to the following additional result *with polynomial security loss*.

By combining our transformation technique and that proposed by Bitansky and Vaikuntanathan [BV15, Proposition IV.1], we can construct single-key *succinct* SKFE from single-key *weakly-succinct* one with *polynomial security loss*. Namely, we can upgrade the succinctness property of SKFE with polynomial security loss. We note that

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

the upgrade of succinctness by the straightforward combination of Theorem 1.2.3 and the result of Ananth et al. [AJS15] incurs quasi-polynomial security loss.

Moreover, we show how to transform *weakly*-selective secure[5] SKFE that is weakly-succinct into a selectively-secure one preserving weak-succinctness property by using some existing results [BNPW16, KNT18].

By applying the above upgrades, we can transform single-key SKFE that is weakly-selective secure and weakly-succinct into single-key SKFE that is selectively secure and succinct with polynomial security loss. We can also accommodate these two upgrades into our transformation used to obtain Theorem 1.2.3. Namely, by applying these upgrades before the transformation, we can construct selective-secure collusion-resistant and succinct SKFE even if the building block single-key scheme is only weakly-selective-secure and weakly-succinct.

1.3 Future Direction

In this thesis, we show IO for all circuits can be constructed solely from collusion-resistant SKFE. Moreover, we show that the requirement for the underlying SKFE, that is, collusion-resistance can be replaced with weak-succinctness.

Prior to our work, all existing generic constructions of IO for all circuits are achieved via PKFE [BV15, AJ15, BNPW16]. Since SKFE is a weaker primitive than PKFE, the requirements for constructing IO looks to be relaxed than ever by our result. In fact, we can remove the LWE assumption from the state-of-the-art constructions of IO [Lin17, LT17] by using our result. We believe that our result makes easier to design IO based on other cryptographic primitives.

One obvious future direction is to find a new assumption that implies SKFE satisfying collusion-resistance or weak-succinctness. By our result, we can immediately achieve IO for all circuits based on such an assumption without using any other assumption.

Another interesting future direction is to explore the power of polynomially secure SKFE. In this work, we show that sub-exponentially secure SKFE satisfying collusion-resistance or weak-succinctness implies IO for all circuits and PKFE. However, it is still unclear how much SKFE that is only polynomially secure is strong as a cryptographic primitive. Especially, it is interesting to clarify whether polynomially secure SKFE with some property implies any public-key primitive or not.

[5] In "weakly" selective security game, adversaries must submit not only challenge message queries but also function queries at the beginning of the game.

Chapter 2

Preliminaries

We define some notations and cryptographic primitives.

2.1 Notations

We write $x \xleftarrow{r} X$ to denote that an element x is chosen from a finite set X uniformly at random and $y \leftarrow \mathsf{A}(x;r)$ to denote that the output of an algorithm A on an input x and a randomness r is assigned to y. When there is no need to write the randomness explicitly, we omit it and simply write $y \leftarrow \mathsf{A}(x)$. For strings x and y, $x\|y$ denotes the concatenation of x and y. Throughout this thesis, λ denotes a security parameter. poly denotes an unspecified polynomial. A function $f(\lambda)$ is a negligible function if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. PPT stands for probabilistic polynomial time. Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$.

2.2 Standard Cryptographic Tools

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

Definition 2.2.1 (Pseudorandom functions) *For sets $\mathcal{D}$ and $\mathcal{R}$, let $\{\mathsf{F}_S(\cdot) : \mathcal{D} \to \mathcal{R} | S \in \{0,1\}^\lambda\}$ be a family of polynomially computable functions. We say that F is secure if for any PPT adversary $\mathcal{A}$, it holds that*

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

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

Goldreich, Goldwasser, and Micali [GGM86] showed the following theorem.

Theorem 2.2.1 ([GGM86]) *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists pseudorandom functions that maps $n(\lambda)$ bits to $m(\lambda)$ bits (i.e., $\mathcal{D} := \{0,1\}^{n(\lambda)}$ and $\mathcal{R} := \{0,1\}^{m(\lambda)}$).*

Definition 2.2.2 (Puncturable pseudorandom function) *For sets $\mathcal{D}$ and $\mathcal{R}$, a puncturable pseudorandom function PPRF consists of a tuple of algorithms $(\mathsf{F}, \mathsf{Punc})$ that satisfies the following two conditions.*

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

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

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

where $S \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, $S^* \leftarrow \mathsf{Punc}(S, \{x_i\}_{i\in[k]})$, and U denotes the uniform distribution over $\mathcal{R}$.

Moreover, for some concrete negligible function $\epsilon(\cdot)$, we say that PPRF is ϵ-secure if for any $\mathcal{A}$ the above indistinguishability gap is smaller than $\epsilon(\lambda)^{\Omega(1)}$.

Similarly to ordinary PRF, puncturable PRF can be realized solely from one-way functions. Concretely, the following theorem holds.

Theorem 2.2.2 ([GGM86, BW13, BGI14, KPTZ13]) *If one-way functions exist, then for all efficiently computable functions $n(\lambda)$ and $m(\lambda)$, there exists a puncturable pseudorandom function that maps $n(\lambda)$ bits to $m(\lambda)$ bits (i.e., $\mathcal{D} := \{0,1\}^{n(\lambda)}$ and $\mathcal{R} := \{0,1\}^{m(\lambda)}$).*

We next review the definition of secret-key encryption (SKE).

Definition 2.2.3 (Secret-key encryption) *An SKE scheme SKE is a two tuple (E, D) of PPT algorithms.*

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

- *The decryption algorithm* D*, given a key* K *and a ciphertext* c*, outputs a message* $\tilde{m} \in \{\bot\} \cup \mathcal{M}$*. This algorithm is deterministic.*

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

CPA security: *We define the security game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

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

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

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

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

CPA-secure SKE can also be constructed based only on one-way functions [LR88].

Theorem 2.2.3 ([LR88]) *If there exist one-way functions, there exists CPA-secure SKE.*

We next review the definition of garbling scheme.

Definition 2.2.4 (Garbling scheme) *Let* $\{\mathcal{C}_n\}_{n\in\mathbb{N}}$ *be a family of circuits where each circuit in* $\mathcal{C}_n$ *takes an* n*-bit input. A circuit garbling scheme* GC *is a two tuple* $(\mathsf{Garble}, \mathsf{Eval})$ *of PPT algorithms.*

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

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

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* $(\widetilde{C}, \{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C)$ *and returns* $(\widetilde{C}, \{L_{j,x_j}\}_{j\in[n]})$ *to* $\mathcal{A}$*. Otherwise, the challenger returns* $(\widetilde{C}, \{L_j\}_{j\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))$.
4. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$.

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

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

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

The following theorem holds.

Theorem 2.2.4 ([Yao86, BHR12, LP09]) *If there exist one-way functions, there exists a garbling scheme for any polynomial size circuits.*

We finally review the definition of decomposable randomized encoding.

Definition 2.2.5 (Decomposable randomized encoding) *Let* $c \geq 1$ *be an integer constant. A* c*-local decomposable randomized encoding* RE*, given security parameter* 1^λ *and a function* f *of size* s *and* n*-bit input, outputs a function* $\hat{f} : \{0,1\}^n \times \{0,1\}^\rho \to \{0,1\}^\mu$ *with the following properties.* ρ *and* μ *are polynomials bounded by* $s \cdot \mathsf{poly}_{\mathsf{RE}}(\lambda, n)$*, where* $\mathsf{poly}_{\mathsf{RE}}$ *is a fixed polynomial.*

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

Decomposability: *There exist* μ *functions* $\hat{f}_1, \cdots, \hat{f}_\mu$ *such that* $\hat{f}(x; r) = (\hat{f}_1(x; r), \cdots, \hat{f}_\mu(x; r))$ *and each* $\hat{f}_i$ *depends on a single bit of* x *at most and* c *bits of* r*. We write* $\hat{f}(x; r) = (\hat{f}_1(x; r_{S_1}), \cdots, \hat{f}_\mu(x; r_{S_\mu}))$*, where* S_i *denotes the subset of bits of* r *that* $\hat{f}_i$ *depends on.*

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

1. *The challenger chooses a bit* $b \xleftarrow{r} \{0,1\}$ *and sends security parameter* 1^λ *to* $\mathcal{A}$.

2. *$\mathcal{A}$ sends a function f of size s and n-bit input and an input $x \in \{0,1\}^n$ to the challenger.*
3. *If $b = 0$, the challenger computes $\hat{f} \leftarrow \mathsf{RE}(1^\lambda, f)$, generates $r \leftarrow \{0,1\}^\rho$, and returns $\hat{f}(x; r)$ to $\mathcal{A}$. Otherwise, the challenger returns $\mathsf{Sim}(1^\lambda, s, f(x))$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

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

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

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

Decomposable randomized encoding can be based on one-way functions.

Theorem 2.2.5 ([Yao86, AIK06]) *If there exist one-way functions, there exists decomposable randomized encoding for all polynomial size functions.*

2.3 Secret-Key Functional Encryption

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

Definition 2.3.1 (Secret-key functional encryption) *An SKFE scheme SKFE is a four tuple of PPT algorithms $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. Below, let $\mathcal{M}$ and $\mathcal{F}$ be the message space and function space of SKFE, respectively.*

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

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

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

Definition 2.3.2 (Selective-message message privacy) *Let* SKFE *be an SKFE scheme whose message space and function space are* $\mathcal{M}$ *and* $\mathcal{F}$*, respectively. Let* q *be a polynomial of* λ*. We define the selective-message message privacy game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

1. *The challenger generates a master secret key* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses the challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. Then, the challenger sends security parameter* 1^λ *to* $\mathcal{A}$*.*

2. $\mathcal{A}$ *sends* $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[p]}$ *to the challenger, where* p *is an a-priori unbounded polynomial of* λ*. The challenger generates ciphertexts* $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b^\ell)(\ell \in [p])$ *and sends them to* $\mathcal{A}$*.*

3. $\mathcal{A}$ *may adaptively make key queries* q *times at most. For a key query* $f \in \mathcal{F}$ *from* $\mathcal{A}$*, the challenger generates* $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f)$*, and returns* sk_f *to* $\mathcal{A}$*. Here,* f *needs to satisfy* $f(m_0^\ell) = f(m_1^\ell)$ *for all* $\ell \in [p]$*.*

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

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

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

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

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

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

Definition 2.3.3 (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.*

1. *The challenger generates a master secret key* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses the challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$*. Then, the challenger sends security parameter* 1^λ *to* $\mathcal{A}$*.*

2. $\mathcal{A}$ *sends* $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[p]}$ *to the challenger, where* p *is an a-priori unbounded polynomial of* λ*. The challenger generates ciphertexts* $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b^\ell)(\ell \in [p])$ *and sends them to* $\mathcal{A}$*.*

3. *$\mathcal{A}$ may adaptively make key queries q times at most. For a key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ from $\mathcal{A}$, the challenger generates $\mathsf{sk}_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f_b)$, 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 $\ell \in [p]$.*

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

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

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

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

We say that an SKFE scheme is $(\mathsf{poly}, \epsilon)$-selective-message function private if it is (q, ϵ)-selective-message function private for any polynomial q. Note that if an SKFE scheme is $(\mathsf{poly}, \epsilon)$-selective-message function private, then it is also ϵ-secure collusion-resistant.

Brakerski and Segev [BS15] showed the following theorem that states we can generically transform a selective-message message private SKFE into selective-message function private one.

Theorem 2.3.1 ([BS15]) *If there exists a (q, δ)-selective-message message private SKFE scheme, there exists a (q, δ)-selective-message function private SKFE scheme, where q is any fixed polynomial.*

We can consider a weaker security notion called weakly-selective-message function privacy.

Definition 2.3.4 (Weakly-selective-message function privacy) *The weakly-selective-message function privacy game is the same as the selective-message function privacy game except that $\mathcal{A}$ must submit not only messages $(m_0^1, m_1^1), \cdots, (m_0^p, m_1^p)$ but also functions $(f_0^1, f_1^1), \ldots, (f_0^q, f_1^q)$ to the challenger at the beginning of the game. For an SKFE scheme SKFE and adversary $\mathcal{A}$, the modified advantage $\mathsf{Adv}^{\mathsf{sm^*\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda)$ is similarly defined as $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda)$. Then, SKFE is said to be weakly-selective-message function private if $\mathsf{Adv}^{\mathsf{sm^*\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda)$ is negligible for any PPT $\mathcal{A}$.*

Next, we define succinctness for SKFE.

Definition 2.3.5 (Succinctness) *Let s and n be the maximum size and input length of functions contained in $\mathcal{F}$, respectively.*

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

Weakly-succinct: *We say that SKFE is weakly-succinct if the size of the encryption circuit is bounded by* $s^{\gamma} \cdot \mathsf{poly}(\lambda, n)$*, where* $\gamma < 1$ *is a fixed constant.*

Collusion-succinct: *We say that SKFE is collusion-succinct if the size of the encryption circuit is bounded by* $\mathsf{poly}(n, \lambda, s, \log q)$*, where* q *is the upper bound of issuable functional decryption keys in the security game.*

In this work, we focus on SKFE for all circuits. Below, unless stated otherwise, let the function space of SKFE be all circuits.

2.4 Indistinguishability Obfuscation

We review the definition of indistinguishability obfuscation (IO).

Definition 2.4.1 (Indistinguishability obfuscation) *A PPT algorithm* $\mathsf{i}\mathcal{O}$ *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*

$$\Pr[C'(x) = C(x) : C' \leftarrow \mathsf{i}\mathcal{O}(1^\lambda, C)] = 1 \ .$$

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

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

holds.

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

2.5 Strong Exponentially-Efficient Indistinguishability Obfuscation

We next review the notion of strong exponentially-efficient IO (SXIO) introduced by Lin, Pass, Seth, and Telang [LPST16] and Bitansky et al. [BNPW16].

Definition 2.5.1 (Strong exponentially-efficient indistinguishability obfuscation) *Let $\gamma < 1$ be a constant. A PPT algorithm $\mathsf{sxi}\mathcal{O}$ is a γ-compressing strong exponentially-efficient indistinguishability obfuscator (SXIO) for a circuit class $\{\mathcal{C}\}_{\lambda\in\mathbb{N}}$ if it satisfies the functionality and indistinguishability in Definition 2.4.1 and the following efficiency requirement:*

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

Bitansky et al. [BNPW16] showed the following theorem.

Theorem 2.5.1 ([BNPW16]) *Assuming there exists ϵ-secure collusion-resistant SKFE for all circuits, where $\epsilon(\cdot)$ is a negligible function. Then, for any constant $\gamma < 1$, there exists ϵ-secure γ-compressing SXIO for polynomial-size circuits with logarithmic length input.*

Chapter 3

IO for All Circuits from Collusion-Resistant SKFE

In this chapter, we show how to construct IO for all circuits based on collusion-resistant SKFE. We first give an overview of our construction and techniques.

3.1 Overview

Our basic strategy is to replace PKFE in the construction of Bitansky and Vaikuntanathan [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 how to construct a relaxed variant of puncturable SKFE that is a single-key scheme and weakly-succinct from collusion-resistant SKFE. Moreover, we show that such a relaxed variant of puncturable SKFE is sufficient for constructing IO.

In Section 3.1.1, we give an overview of the construction of Bitansky and Vaikuntanathan [BV15]. Then, in Section 3.1.2, we explain why SKFE must be "puncturable" when we replace PKFE with SKFE in their construction. Finally, we give an overview of how to construct our puncturable SKFE scheme and IO in Section 3.1.3 and Section 3.1.4, respectively.

3.1.1 Construction of IO based on PKFE

The main idea of Bitansky and Vaikuntanathan is to design an obfuscator $\mathsf{i}\mathcal{O}_i$ for circuits with i-bit input from an obfuscator $\mathsf{i}\mathcal{O}_{i-1}$ 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 $\mathsf{i}\mathcal{O}_n$ for circuits with n-bit input since we can easily achieve $\mathsf{i}\mathcal{O}_1$ for circuits with 1-bit input by outputting an entire truth table of a circuit with 1-bit input.

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

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

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 [BS15] 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.

$\mathsf{i}\mathcal{O}_i$ based on $\mathsf{i}\mathcal{O}_{i-1}$ and PKFE works as follows. The construction additionally uses plain secret-key encryption (SKE) to implement the technique used by Brakerski and Segev [BS15]. To obfuscate a circuit C with i-bit input, it first generates a key pair $(\mathsf{PK}_i, \mathsf{MSK}_i)$ of PKFE. Then, using MSK_i, it generates a functional key sk_{C^*} tied to the following circuit C^*. C^* has hardwired two SKE ciphertexts $\mathsf{CT}_0^{\mathsf{ske}}$ and $\mathsf{CT}_1^{\mathsf{ske}}$ of plaintext C under independent keys K_0 and K_1, respectively. C^* expects as an input not only an i-bit string $\boldsymbol{x}_i$ but also an SKE key K_b. On those inputs, C^* first obtains C by decrypting $\mathsf{CT}_b^{\mathsf{ske}}$ by K_b and outputs $U(C, \boldsymbol{x}_i) = C(\boldsymbol{x}_i)$, where $U(\cdot, \cdot)$ is an universal circuit. Finally, the construction obfuscates the following encryption circuit E_{i-1} by $\mathsf{i}\mathcal{O}_{i-1}$. E_{i-1} has hardwired PK_i and K_b. On input $(i-1)$-bit string $\boldsymbol{x}_{i-1}$, it outputs ciphertexts $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1} \| 0, K_b))$ and $\mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1} \| 1, K_b))$, where Enc is the encryption algorithm of PKFE. The resulting obfuscation of C is a tuple $(\mathsf{sk}_{C^*}, \mathsf{i}\mathcal{O}_{i-1}(\mathsf{E}_{i-1}))$. We show the simplified description of circuits C^* and E_{i-1} in Figure 3.1 and 3.2, respectively. 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.

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

Consequently, by using this bit extension construction, the obfuscation of a circuit C with n-bit input consists of n functional keys $\mathsf{sk}_1, \cdots, \mathsf{sk}_n$ each of which is generated under a different master secret key MSK_i, and pair of ciphertexts of 0 and 1 under

Hard-Coded Constants: $\mathsf{CT}_0^{\mathsf{ske}}$, $\mathsf{CT}_1^{\mathsf{ske}}$
Input: $\boldsymbol{x}_i$, K_b
1. Compute $C = \mathsf{D}(K_b, \mathsf{CT}_b^{\mathsf{ske}})$.
2. Return $U(C, \boldsymbol{x}_i)$.

Figure 3.1: Simplified C^*.

Hard-Coded Constants: PK_i, K_b
Input: $\boldsymbol{x}_{i-1} \in \{0,1\}^{i-1}$
1. Compute $\mathsf{CT}_{i,x_i} \xleftarrow{\mathsf{r}} \mathsf{Enc}(\mathsf{PK}_i, (\boldsymbol{x}_{i-1} \| x_i, K_b))$.
2. Output $\mathsf{CT}_{i,0}$ and $\mathsf{CT}_{i,1}$.

Figure 3.2: Simplified E_{i-1}.

PK_1 corresponding to MSK_1. For any $\boldsymbol{x}_n = x_1 \cdots x_n \in \{0,1\}^n$, we can first compute a ciphertext of $\boldsymbol{x}_n$ by repeatedly decrypting a ciphertext of $\boldsymbol{x}_{i-1} = x_1 \cdots x_{i-1}$ by sk_{i-1} and obtaining a ciphertext of $\boldsymbol{x}_i = x_1 \cdots x_i$ for every $i \in \{2, \ldots, n\}$. We can finally obtain $C(\boldsymbol{x}_n)$ by decrypting the ciphertext of $\boldsymbol{x}_n$ by sk_n.

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

We can prove the security of the construction recursively. More precisely, we can prove the security of $\mathsf{i}\mathcal{O}_i$ based on those of $\mathsf{i}\mathcal{O}_{i-1}$, PKFE, and SKE. Note that it is sufficient that PKFE satisfies a mild selective-security to complete the proof. Their security proof relies on the argument of probabilistic IO formalized by Canneti, Lin, Tessaro, and Vaikuntanathan [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.

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

The above security argument might not work if ordinary SKFE is used instead of PKFE. This intuition comes from the impossibility result shown by Barak et al. [BGI+01].

In fact, Bitansky and Vaikuntanathan showed that it is impossible to instantiate their IO by using SKFE. More precisely, they showed that there exists a secure SKFE scheme such that their transformation results in insecure IO if the SKFE scheme is used as the building block. This is why they adopted PKFE as their building block. Therefore, in order to replace PKFE with SKFE in the construction above, we need SKFE whose security holds even when its encryption circuit is publicly available. As one of such primitives, Bitansky and Vaikuntanathan proposed *puncturable SKFE*.

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

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

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

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

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

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

3.1.3 Puncturable SKFE from SKFE

In this work, we show we can construct single-key weakly-succinct puncturable SKFE from collusion-resistant SKFE. More specifically, we show the following two results. First, we show how to construct single-key non-succinct puncturable SKFE based only on one-way functions. In addition, we show that we can transform it into single-key weakly-succinct one using collusion-resistant SKFE. Our formalization of puncturable SKFE is different from that of Bitansky and Vaikuntanathan [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 generates a garbled circuit $\widetilde{U}_m$ with labels $\{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$ by garbling an universal circuit $U(\cdot, m)$ into which m is hardwired. Then, we encrypt $L_{j,\alpha}$ under $K_{j,\alpha}$ and obtain an SKE ciphertext $c_{j,\alpha}$ for every $j \in [s]$ and $\alpha \in \{0,1\}$. The resulting ciphertext of the scheme is $(\widetilde{U}_m, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.

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

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

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

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

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

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

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

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

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

Achieving Weak-Succinctness via Collusion-Succinctness

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

We extend the work by Kitagawa, Nishimaki, and Tanaka [KNT18] that showed how to transform non-succinct PKFE into weakly-succinct one using collusion-resistant SKFE. They accomplished the transformation via a *collusion-succinct* scheme. We try to accommodate their transformation techniques into the context of puncturable SKFE.

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

We first show that we can construct collusion-succinct puncturable SKFE based on single-key non-succinct puncturable SKFE and collusion-resistant SKFE. Then, we transform the collusion-succinct scheme into 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 [BV15] and that proposed by Ananth, Jain, and Sahai [AJS15]. We give an illustration of our construction path in Figure 3.3.

The general picture is similar to that of Kitagawa et al. [KNT18] and we can accomplish the latter transformation based on a known technique, but there is a technical hurdle in the former transformation. The most biggest issue is how to define punctured master secret keys and the punctured encryption algorithm. We show the overview of the former transformation and explain the technical hurdle below.

Construction of collusion-succinct scheme. Our goal of this step is to construct a collusion-succinct scheme, that is, a scheme which supports q functional keys and the size of whose encryption and punctured encryption circuits are sub-linear in q, where q is

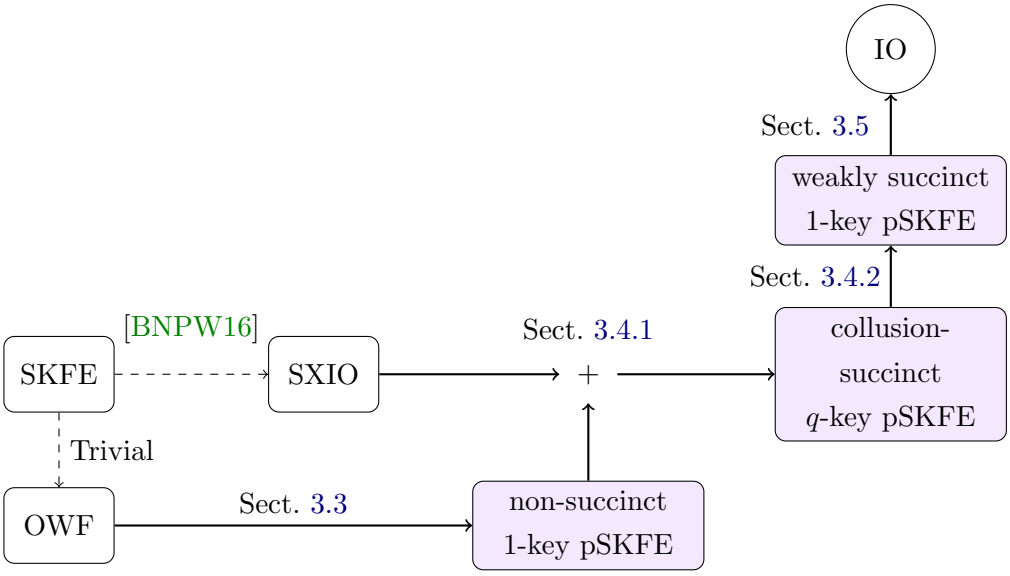

Figure 3.3: Illustration of our construction path. pSKFE denotes puncturable SKFE. Dashed lines denote known or trivial implications. White boxes denote our ingredients or goal. Purple boxes denote our core schemes. A transformation from an object in a rectangle to one in a rectangle incurs only polynomial security loss. A transformation from an object in a rectangle to one in a circle incurs super-polynomial security loss.

an a-priori fixed polynomial. The key tool for achieving this goal is strong exponentially-efficient IO (SXIO) proposed by Lin et al. [LPST16].

SXIO is a relaxed variant of IO. SXIO is required that, given a circuit C with n-bit input, it runs in $2^{\gamma n} \cdot \mathsf{poly}(\lambda, |C|)$-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 [BNPW16].

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

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

Hard-Coded Constants: S, tag, m
Input: $i \in [q]$
1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$.
2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
3. Return $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \mathsf{tag}, m)$.

Figure 3.4: Simplified $\mathsf{E}_{\mathsf{1Key}}$.

we generate q punctured master secret keys $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ 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, BNPW16, KNT18]. Let $\mathsf{sxi}\mathcal{O}$ be SXIO. We set a ciphertext as a circuit computing q ciphertexts obfuscated by $\mathsf{sxi}\mathcal{O}$ instead of setting it as q ciphertexts themselves. Concretely, we obfuscate the following circuit $\mathsf{E}_{\mathsf{1Key}}$ using $\mathsf{sxi}\mathcal{O}$. $\mathsf{E}_{\mathsf{1Key}}$ has hardwired message m, tag tag, and puncturable PRF key S, and on input $i \in [q]$, it first generates MSK_i pseudorandomly from S and i, and then outputs a ciphertext of m under MSK_i and tag. We show the simplified description of $\mathsf{E}_{\mathsf{1Key}}$ in Figure 3.4.

Note that the master secret key of this scheme is now one puncturable PRF key S. In other words, the scheme generates q master secret keys of the single-key scheme from one puncturable PRF key. For the formal description of $\mathsf{E}_{\mathsf{1Key}}$, see Figure 3.7 in Section 3.4.1.

The size of $\mathsf{E}_{\mathsf{1Key}}$ is independent of q since $\mathsf{E}_{\mathsf{1Key}}$ consists of one PRF evaluation and setup and encryption procedure of the single-key scheme.[1] Therefore, the time needed to compute $\mathsf{sxi}\mathcal{O}(\mathsf{E}_{\mathsf{1Key}})$ is bounded by $2^{\gamma \log q} \cdot \mathsf{poly}(\lambda, |m|) = q^\gamma \cdot \mathsf{poly}(\lambda, |m|)$ 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

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

Hard-Coded Constants: S, tag
Input: $i \in [q]$
1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$.
2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
3. Return $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{Punc}(\mathsf{MSK}_i, \mathsf{tag})$.

Figure 3.5: Simplified $\mathsf{P}_{\mathsf{1Key}}$.

Hard-Coded Constants: $\mathsf{MSK}^*\{\mathsf{tag}\}$, tag', m
Input: $i \in [q]$
1. Parse $\mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}}) \leftarrow \mathsf{MSK}^*\{\mathsf{tag}\}$.
2. Compute $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}})(i)$.
3. Return $\mathsf{CT}_i \leftarrow \mathsf{PEnc}(\mathsf{MSK}^*_i\{\mathsf{tag}\}, \mathsf{tag}', m)$.

Figure 3.6: Simplified $\mathsf{PE}_{\mathsf{1Key}}$.

security to that of the building block single-key scheme.

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

Our solution is to set a punctured master secret key as also an obfuscated circuit under SXIO. More precisely, we obfuscate the following circuit $\mathsf{P}_{\mathsf{1Key}}$. $\mathsf{P}_{\mathsf{1Key}}$ has hardwired tag tag and puncturable PRF key S. Note that S is the master secret key thus is the same puncturable PRF key as that hardwired into $\mathsf{E}_{\mathsf{1Key}}$. On input $i \in [q]$, $\mathsf{P}_{\mathsf{1Key}}$ first generates MSK_i pseudorandomly from S and i, and then outputs a punctured master secret key $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ of the single-key scheme. We show the simplified description of $\mathsf{P}_{\mathsf{1Key}}$ in Figure 3.5. For the formal description of $\mathsf{P}_{\mathsf{1Key}}$, see Figure 3.8 in Section 3.4.1.

In addition, we define the punctured encryption algorithm as follows. On input $\mathsf{MSK}^*\{\mathsf{tag}\}$ that is $\mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}})$, tag tag', and message m, the punctured encryption algorithm obfuscates the following circuit $\mathsf{PE}_{\mathsf{1Key}}$ using $\mathsf{sxi}\mathcal{O}$ and outputs the obfuscated circuit. $\mathsf{PE}_{\mathsf{1Key}}$ has hardwired $\mathsf{MSK}^*\{\mathsf{tag}\}$, tag', and m, and on input $i \in [q]$, it first generates the i-th punctured key $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ by feeding i into $\mathsf{MSK}^*\{\mathsf{tag}\} = \mathsf{sxi}\mathcal{O}(\mathsf{PE}_{\mathsf{1Key}})$, and then outputs a ciphertext of m under $\mathsf{MSK}^*_i\{\mathsf{tag}\}$ and tag' using the punctured encryption algorithm of the single-key scheme. If the compression factor of $\mathsf{sxi}\mathcal{O}$ is *sufficiently small*, we ensure that the running time of this punctured encryption algorithm is sub-linear in q. We show the simplified description of $\mathsf{PE}_{\mathsf{1Key}}$ in Figure 3.6. For the formal description of $\mathsf{PE}_{\mathsf{1Key}}$, see Figure 3.9 in Section 3.4.1.

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 $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$, respectively.

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

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

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

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

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

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

Transforming into 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 [BV15] 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.

3.1.4 IO from Puncturable SKFE

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

The construction of IO based on puncturable SKFE is almost the same as that based on PKFE proposed by Bitansky and Vaikuntanathan [BV15]. It does not depend on which functionality preserving property puncturable SKFE satisfies. Recall that, in their construction, a key pair $(\mathsf{PK}_i, \mathsf{MSK}_i)$ of PKFE is generated and the circuit E_{i-1} that has hardwired PK_i is obfuscated at every recursive step. In our construction based on puncturable SKFE, a master secret key MSK_i of puncturable SKFE is generated and E_{i-1} that has hardwired MSK_i is obfuscated at each recursive step. Concretely, we construct E_{i-1} as a circuit that has hardwired MSK_i and an SKE key K, and on $(i-1)$-bit input $\boldsymbol{x}_{i-1}$, it outputs a ciphertext of $(\boldsymbol{x}_{i-1}\|x_i, K)$ for $x_i \in \{0,1\}$ under MSK_i and a tag $\boldsymbol{x}_{i-1}$, that is, $\mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1}\|x_i, K))$ for $x_i \in \{0,1\}$. In the proof, we replace MSK_i hardwired into E_{i-1} with the tuple of a punctured master secret key $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ punctured at $\boldsymbol{j} \in \{0,1\}^{i-1}$ and a ciphertext of $(\boldsymbol{j}\|x_i, K)$ for $x_i \in \{0,1\}$, where $\boldsymbol{j}$ is a string in $\{0,1\}^{i-1}$ that we focus on at that time.

Outline of Security Proof

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

Analogous to IO based on PKFE, we can accomplish this proof recursively. More precisely, we can prove the security of $\mathsf{i}\mathcal{O}_i$ based on those of $\mathsf{i}\mathcal{O}_{i-1}$, 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 $\mathsf{i}\mathcal{O}_i(C_0)$ and $\mathsf{i}\mathcal{O}_i(C_1)$. 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 $\mathsf{i}\mathcal{O}_i(C_0)$. Recall that $\mathsf{i}\mathcal{O}_i(C_0)$ consists of sk_{C^*} and $\mathsf{i}\mathcal{O}_{i-1}(\mathsf{E}_{i-1})$. C^* has hardwired two SKE ciphertexts $\mathsf{CT}_0^{\mathsf{ske}}$ and $\mathsf{CT}_1^{\mathsf{ske}}$ of C_0 under independent keys K_0 and K_1. On i-bit input $\boldsymbol{x}_i$ and SKE key K_b, C^* first obtains C by

decrypting $\mathsf{CT}_b^{\mathsf{ske}}$ by K_b and outputs $C(\boldsymbol{x}_i)$.

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

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

Therefore, the main part of the proof is how we change the circuit E_{i-1} from encrypting K_0 in $\mathcal{H}_1$ to encrypting K_1 in $\mathcal{H}_2$. As mentioned earlier, we accomplish this task by relying on the argument of probabilistic IO formalized by Canneti et al. [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 $\boldsymbol{j} \in \{0,1\}^{i-1}$, it outputs ciphertexts of $(\boldsymbol{j}\|x_i, K_1)$ instead of $(\boldsymbol{j}\|x_i, K_0)$ for $x_i \in \{0,1\}$, where $\boldsymbol{j}$ is the binary representation of j. More precisely, we construct E_{i-1} in $\mathcal{H}_{1,j}$ as follows. E_{i-1} has hardwired MSK_i, K_0, and K_1. On $(i-1)$-bit input $\boldsymbol{x}_{i-1}$,

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

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

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

(1) In the first step, we hardwire ciphertexts of $(\boldsymbol{j}\|x_i, K_0)$ under MSK_i and a tag $\boldsymbol{j}$ for $x_i \in \{0,1\}$ in E_{i-1}. In addition, we replace hardwired MSK_i in E_{i-1} with $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ that is a master secret key punctured at a tag $\boldsymbol{j}$. On $(i-1)$-bit input $\boldsymbol{x}_{i-1}$,

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

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

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

(3) In the final step, we change E_{i-1} so that it does not have hardwired ciphertexts of $(\boldsymbol{j}\|x_i, K_1)$ for $x_i \in \{0,1\}$. Moreover, we change E_{i-1} so that E_{i-1} has hardwired MSK_i and use it to generate the output ciphertexts. This change also does not affect the functionality of E_{i-1}, and thus we can accomplish this step by relying on the security of $\mathsf{i}\mathcal{O}_{i-1}$ 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 $j \in \{0,\ldots,2^{i-1}-1\}$. By combining $\mathcal{H}_1 \sim \mathcal{H}_{1,0}$ and $\mathcal{H}_{1,2^{i-1}} \sim \mathcal{H}_2$, we obtain $\mathcal{H}_1 \sim \mathcal{H}_2$.

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

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

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

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

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

We see that E_{i-1} in $\mathcal{H}_{1,j}$ and $\mathcal{H}^0_{1,j}$ 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 $\mathcal{H}_{1,j} \sim \mathcal{H}_{1,j,0}$, $\mathcal{H}^0_{1,j} \sim \mathcal{H}_{1,j,2^{i-1}}$, and $\mathcal{H}_{1,j,j} \sim \mathcal{H}_{1,j,j+1}$ from the security guarantee of $\mathsf{i}\mathcal{O}_{i-1}$.

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

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

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

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

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

Therefore, we obtain $\mathcal{H}_{1,j} \sim \mathcal{H}^0_{1,j}$ 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] and the case where puncturable SKFE satisfies functionality preserving under puncturing. Our security proof incurs roughly $2^{2\cdot i}$ security loss while the latter proofs incurs 2^i security loss when we prove the security of $\mathsf{i}\mathcal{O}_i$ based on that of $\mathsf{i}\mathcal{O}_{i-1}$. Nevertheless, this difference is not an issue in the sense that if the building block primitives are roughly $2^{\Omega(n^2)}$-secure, we can prove the security of our indistinguishability obfuscator, where n is the input length of circuits to be obfuscated. This requirement is the same as that of Bitansky and Vaikuntanathan.

3.2 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 2.5.1. Thus, we can construct our puncturable SKFE from standard SKFE.

3.2.1 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 work is also a sufficient building block of IO. We state differences between our definition and theirs after describing the syntax and security of our puncturable SKFE.

Definition 3.2.1 (Puncturable secret-key functional encryption) *A puncturable SKFE scheme* pSKFE *is a tuple* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *of six PPT algorithms. Below, let* $\mathcal{M}$*,* $\mathcal{F}$*, and* $\mathcal{T}$ *be the message space, function space, and tag space of* pSKFE*, respectively. In addition, let* q *be a polynomial denoting the upper bound of the number of issuable functional keys.*

- *The setup algorithm* Setup*, given a security parameter* 1^λ*, outputs a master secret key* MSK*.*

- *The key generation algorithm* KG*, given a master secret key* MSK*, function* $f \in \mathcal{F}$*, and an index* $i \in [q]$*, outputs a functional key* sk_f*.*

- *The encryption algorithm* Enc*, given a master secret key* MSK*, a tag* tag*, and a message* $m \in \mathcal{M}$*, outputs a ciphertext* CT*.*

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

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

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

3.2.2 Security

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

Definition 3.2.2 (Secure puncturable SKFE) *Let* $\mathsf{pSKFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *be puncturable SKFE. Below, let* $\mathcal{M}$*,* $\mathcal{F}$*, and* $\mathcal{T}$ *be the message space, function space, and tag space of* pSKFE*, respectively. In addition, let* q *be a polynomial denoting the upper bound of the number of issuable functional keys. We say that* pSKFE *is secure puncturable SKFE if it satisfies the following properties.*

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

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

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

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

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

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

$\mathcal{A}$ is said to be valid if $f_i(m_0) = f_i(m_1)$ holds for every $i \in [q]$ in the above game.

We say that pSKFE *satisfies semantic security at punctured tag if for any valid PPT $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$.*

We further say that pSKFE *satisfies ϵ-semantic security at punctured tag, for some concrete negligible function $\epsilon(\cdot)$, if for any valid PPT $\mathcal{A}$ the advantage $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda)$ is smaller than $\epsilon(\lambda)^{\Omega(1)}$.*

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

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

Definition 3.2.3 (Indistinguishability of functionality under puncturing) *Let* $\mathsf{pSKFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ *be puncturable SKFE whose message space and tag space are $\mathcal{M}$ and $\mathcal{T}$, respectively. We define indistinguishability of functionality game between a challenger and an adversary $\mathcal{A}$ as follows.*

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

   Then, the challenger returns $(\mathsf{MSK}, \mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT})$ *to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$.*

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

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

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

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

Definition 3.2.4 (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.*

3.2.3 Efficiency

We introduce the notion of succinctness for puncturable SKFE.

Definition 3.2.5 (Succinctness) *Let n be the input length of circuits in $\mathcal{F}$, and s the maximum size of circuits contained in $\mathcal{F}$.*

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

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

3.2.4 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 an universal circuit into which the message is hardwired, and then masks labels of the garbled circuit by a string generated by puncturable PRF. A tag fed to the encryption algorithm is used as an input to puncturable PRF. See Section 3.3 for details.

In our construction of IO in Section 3.5, 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 3.4.

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.

In addition to the syntactic differences above, there is a difference about security as stated below.

Functionality under puncturing. We defined indistinguishability of functionality under puncturing in Definition 3.2.3. The reason we introduce the relaxed notion of functionality preserving property is that our weakly-succinct scheme does not satisfy functionality preserving under puncturing in Definition 3.2.2 but the relaxed one. Our *non-succinct* scheme satisfies functionality preserving under puncturing.

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

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

Construction. The scheme consists of the following algorithms.

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

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

$\mathsf{1Key.KG}(\mathsf{MSK}, f)$:

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

$\mathsf{1Key.Enc}(\mathsf{MSK}, \mathsf{tag}, m)$:

- Parse $\{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}$.
- Compute $(\widetilde{U}, \{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, U(\cdot, m))$.
- For every $j \in [s]$ and $\alpha \in \{0,1\}$, compute $R_{j,\alpha} \leftarrow \mathsf{F}(S_{j,\alpha}, \mathsf{tag})$ and $c_{j,\alpha} \leftarrow L_{j,\alpha} \oplus R_{j,\alpha}$.
- Return $\mathsf{CT} \leftarrow (\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.

$\mathsf{1Key.Dec}(\mathsf{sk}_f, \mathsf{tag}, \mathsf{CT})$:

- Parse $(f, \{S_j\}_{j\in[s]}) \leftarrow \mathsf{sk}_f$ and $(\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{CT}$.

- For every $j \in [s]$, compute $R_j \leftarrow \mathsf{F}(S_j, \mathsf{tag})$ and $L_j \leftarrow c_{j,f[j]} \oplus R_j$.
- Return $y \leftarrow \mathsf{Eval}(\widetilde{U}, \{L_j\}_{j\in[s]})$.

$\mathsf{1Key.Punc}(\mathsf{MSK}, \mathsf{tag})$:

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

$\mathsf{1Key.PEnc}(\mathsf{MSK}^*, \mathsf{tag}', m)$

- Parse $\{S^*_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{MSK}^*$.
- Compute $(\widetilde{U}, \{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, U(\cdot, m))$.
- For every $j \in [s]$ and $\alpha \in \{0,1\}$, compute $R_{j,\alpha} \leftarrow \mathsf{F}_{S^*_{j,\alpha}}(\mathsf{tag}')$ and $c_{j,\alpha} \leftarrow L_{j,\alpha} \oplus R_{j,\alpha}$.
- Return $\mathsf{CT} \leftarrow (\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.

Then, we have the following theorem.

Theorem 3.3.1 *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 3.3.1. 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 $\mathsf{1Key.PEnc}(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{tag}', m; r) = \mathsf{1Key.Enc}(\mathsf{MSK}, \mathsf{tag}', m; r)$ for every $m \in \mathcal{M}$, $(\mathsf{tag}, \mathsf{tag}') \in \mathcal{T} \times \mathcal{T}$ such that $\mathsf{tag} \neq \mathsf{tag}'$, randomness r, $\mathsf{MSK} \leftarrow \mathsf{1Key.Setup}(1^\lambda)$, and $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{1Key.Punc}(\mathsf{MSK}, \mathsf{tag})$, 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 $\ell \in \{0, \ldots, 3\}$, let $\mathtt{SUC}_\ell$ 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. We have $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{pSKFE},\mathcal{A}}(\lambda) = 2|\Pr[\mathtt{SUC}_0] - \frac{1}{2}|$. 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 $\mathsf{MSK} \leftarrow \{S_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}$. 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 $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\mathsf{tag} \in \mathcal{T}$, and a function f to the challenger.
3. The challenger computes $(\widetilde{U}, \{L_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, U(\cdot, m_b))$ and $R_{j,\alpha} \leftarrow \mathsf{F}(S_{j,\alpha}, \mathsf{tag})$ and $c_{j,\alpha} \leftarrow L_{j,\alpha} \oplus R_{j,\alpha}$ for every $j \in [s]$ and $\alpha \in \{0,1\}$, and sets $\mathsf{CT} \leftarrow (\widetilde{U}, \{c_{j,\alpha}\}_{j\in[s],\alpha\in\{0,1\}})$.

   Next, the challenger sets $\mathsf{sk}_f \leftarrow (f, \{S_{j,f[j]}\}_{j\in[s]})$.

   Then, the challenger computes $S^*_{j,\alpha}\{\mathsf{tag}\} \leftarrow \mathsf{Punc}_\mathsf{F}(S_{j,\alpha}, \mathsf{tag})$ for every $j \in [s]$ and $\alpha \in \{0,1\}$, and sets $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \{S^*_{j,\alpha}\{\mathsf{tag}\}\}_{j\in[s],\alpha\in\{0,1\}}$.

   The challenger returns $(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT}, \mathsf{sk}_f)$ 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.

We have $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \delta^{\Omega(1)}$ from the pseudorandomness of punctured point of PPRF.

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, $c_{j,1-j[j]}$ is generated as $c_{j,1-f[j]} \leftarrow L_{j,1-f[j]} \oplus R_{j,1-f[j]}$ 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 $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| = 0$.

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

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

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

of $\mathcal{A}$ as

$$\begin{aligned}\frac{1}{2}\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{OneKey},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_3]| \\ &\leq \sum_{\ell=0}^{2} |\Pr[\mathtt{SUC}_\ell] - \Pr[\mathtt{SUC}_{\ell+1}]| \quad . \end{aligned} \tag{3.1}$$

From the above argument, each term of the right side of inequality 3.1 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{OneKey},\mathcal{A}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of $\mathcal{A}$ is arbitrary, OneKey satisfies δ-semantic security at punctured tag. □ (**Theorem 3.3.1**)

3.4 From Non-Succinct Puncturable SKFE 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. [BNPW16] and Kitagawa et al. [KNT18]), and thus we use an obfuscated encryption circuit of the building block scheme by SXIO as a ciphertext of the resulting scheme. This is also the reason the resulting weakly-succinct scheme satisfies only indistinguishability of functionality under puncturing property.

Below, we start with the first step.

3.4.1 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 encryp-

tion algorithm and the punctured encryption algorithm is sub-linear in q. We show the construction below.

Let $\mathsf{OneKey} = (\mathsf{1Key.Setup}, \mathsf{1Key.KG}, \mathsf{1Key.Enc}, \mathsf{1Key.Dec}, \mathsf{1Key.Punc}, \mathsf{1Key.PEnc})$ be puncturable SKFE that we constructed in Section 3.3. Let $\mathsf{sxi}\mathcal{O}$ be SXIO and $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc}_\mathsf{F})$ puncturable PRF. Using OneKey, $\mathsf{sxi}\mathcal{O}$, and PPRF, we construct puncturable SKFE $\mathsf{CollSuc} = (\mathsf{CS.Setup}, \mathsf{CS.KG}, \mathsf{CS.Enc}, \mathsf{CS.Dec}, \mathsf{CS.Punc}, \mathsf{CS.PEnc})$ as follows. We again note that q is a fixed polynomial of λ. Let the tag space of $\mathsf{CollSuc}$ be $\mathcal{T}$. Then, the tag space of OneKey is also $\mathcal{T}$.

Construction. The scheme consists of the following algorithms.

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

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

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

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

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

- Parse $S \leftarrow \mathsf{MSK}$.
- Generate $S_{\mathsf{Enc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and return $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m])$. The circuit $\mathsf{E}_{\mathsf{1Key}}$ is defined in Figure 3.7.

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

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

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

- Parse $S \leftarrow \mathsf{MSK}$.
- Generate $S_{\mathsf{Punc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and compute $\widetilde{\mathsf{P}} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}])$. The circuit $\mathsf{P}_{\mathsf{1Key}}$ is defined in Figure 3.8.
- Return $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \widetilde{\mathsf{P}}$.

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

- Parse $\widetilde{\mathsf{P}} \leftarrow \mathsf{MSK}^*$.

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

Hardwired: Two PRF keys S and S_{Enc}, a tag tag, and a message m.

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

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

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

Figure 3.7: The description of $\mathsf{E}_{\mathsf{1Key}}$.

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

Hardwired: Two PRF keys S and S_{Punc}, and a tag tag.

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

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

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

Figure 3.8: The description of $\mathsf{P}_{\mathsf{1Key}}$.

- Generate $S_{\mathsf{Enc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and return $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{PE}_{\mathsf{1Key}}[\widetilde{\mathsf{P}}, S_{\mathsf{Enc}}, \mathsf{tag}', m])$. The circuit $\mathsf{PE}_{\mathsf{1Key}}$ is defined in Figure 3.9.

Then, we have the following theorem.

Theorem 3.4.1 *Let $\delta(\cdot)$ be some negligible function. Let* OneKey *be δ-secure single-key puncturable SKFE constructed in Section 3.3. Let* sxi$\mathcal{O}$ *be δ-secure γ-compressing SXIO, where γ is a sufficiently small constant such that $\gamma < 1$. Let* PPRF *be δ-secure puncturable PRF. Then,* CollSuc *is (δ, δ)-secure puncturable SKFE with indistinguishability of functionality that is collusion-succinct with compression factor $\hat{\gamma}$, which is a constant smaller than* 1.

The concrete value of $\hat{\gamma}$ is determined in the efficiency analysis in the following proof of Theorem 3.4.1. 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 [BNPW16].

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

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

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

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

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

Figure 3.9: The description of $\mathsf{PE}_{\mathsf{1Key}}$.

Encryption circuit $\mathsf{E}_{\mathsf{1Key}}^{i^*}[S^*, S_{\mathsf{Enc}}^*, \mathsf{tag}, m_0, m_1, \mathsf{CT}_{i^*}](i)$:

Hardwired: Two punctured PRF keys S^* and S_{Enc}^*, a tag tag, two messages m_0 and m_1, and a ciphertext CT_{i^*}.

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

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

1. If $i = i^*$ and $\mathsf{CT}_{i^*} \neq \bot$, return CT_{i^*}.
2. Else, compute as follows:
   - Compute $r_{\mathsf{Setup}}^i \leftarrow \mathsf{F}_{S^*}(i)$ and $r_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_{\mathsf{Enc}}^*}(i)$.
   - Compute $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r_{\mathsf{Setup}}^i)$.
   - If $i \leq i^*$, compute $\mathsf{CT}_i \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_i, \mathsf{tag}, m_1; r_{\mathsf{Enc}})$, and otherwise compute $\mathsf{CT}_i \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_i, \mathsf{tag}, m_0; r_{\mathsf{Enc}})$.
   - Return CT_i.

Figure 3.10: The description of $\mathsf{E}_{\mathsf{1Key}}^{i^*}$. The circuit is defined for every $i^* \in [q]$.

Proof of Theorem 3.4.1. 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 $\mathsf{CollSuc}$.

Padding Parameter. In order to complete this proof, we ensure that the encryption circuits $\mathsf{E}_{\mathsf{1Key}}$, $\mathsf{PE}_{\mathsf{1Key}}$, and $\mathsf{E}_{\mathsf{1Key}}^{i^*}$ for every $i^* \in [q]$ are indistinguishable when we obfuscate them by SXIO. Moreover, the obfuscated $\mathsf{P}_{\mathsf{1Key}}$ and $\mathsf{P}_{\mathsf{1Key}}^{i^*}$ 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 $\mathsf{P}_{\mathsf{1Key}}$ and $\mathsf{P}_{\mathsf{1Key}}^{i^*}$ because the description of $\mathsf{PE}_{\mathsf{1Key}}$ includes obfuscated $\mathsf{P}_{\mathsf{1Key}}$.

To guarantee the indistinguishability of $\mathsf{P}_{\mathsf{1Key}}$ and $\mathsf{P}_{\mathsf{1Key}}^{i^*}$ when we obfuscate them, we

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

Hardwired: Two PRF keys S and S_{Punc}, a tag tag, and a punctured master secret key $\mathsf{MSK}^*_{i^*}$.

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

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

1. If $i = i^*$, return $\mathsf{MSK}^*_{i^*}$.
2. Else, compute as follows:
   - Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_{S^*}(i)$ and $r_{\mathsf{Punc}} \leftarrow \mathsf{F}_{S^*_{\mathsf{Punc}}}(i)$.
   - Compute $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
   - Return $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{1Key.Punc}(\mathsf{MSK}_i, \mathsf{tag}; r_{\mathsf{Punc}})$.

Figure 3.11: The description of $\mathsf{P}^{i^*}_{\mathsf{1Key}}$. The circuit is defined for every $i^* \in [q]$.

need to set

$$\mathsf{pad}_{\mathsf{P}} := \max(|\mathsf{P}_{\mathsf{1Key}}|, |\mathsf{P}^{i^*}_{\mathsf{1Key}}|) \ .$$

Both $\mathsf{P}_{\mathsf{1Key}}$ and $\mathsf{P}^{i^*}_{\mathsf{1Key}}$ 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

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

where $\mathsf{poly}_{\mathsf{P}}$ is some fixed polynomial.

Then, we move on to the analysis of the padding parameter for encryption algorithms. We need to set $\mathsf{pad}_{\mathsf{E}}$ as

$$\mathsf{pad}_{\mathsf{E}} := \max(|\mathsf{E}_{\mathsf{1Key}}|, |\mathsf{PE}_{\mathsf{1Key}}|, |\mathsf{E}^{i^*}_{\mathsf{1Key}}|) \ .$$

$\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{E}^{i^*}_{\mathsf{1Key}}$ 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

$$\max(|\mathsf{E}_{\mathsf{1Key}}|, |\mathsf{E}^{i^*}_{\mathsf{1Key}}|) \leq \mathsf{poly}(\lambda, n, s, \log q) \ . \tag{3.2}$$

In addition, $\mathsf{PE}_{\mathsf{1Key}}$ includes one PRF evaluation over the domain $[q]$, the execution of $\widetilde{\mathsf{P}}$ that is obfuscated P by $\mathsf{sxi}\mathcal{O}$, and the punctured encryption procedure of OneKey. Then, since the non-trivial efficiency of $\mathsf{sxi}\mathcal{O}$, when we obfuscate a circuit C with input

space $[N]$ by $\mathsf{sxi}\mathcal{O}$, we can bound the size of obfuscated C by $N^{\gamma} \cdot |C|^{c} \cdot \mathsf{poly}_{\mathsf{sxi}\mathcal{O}}(\lambda)$, where $\gamma < 1$ and c are constants. Thus, we have

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

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

$$\begin{aligned}|\mathsf{PE}_{\mathsf{1Key}}| &\leq \mathsf{poly}(\lambda, \log q) + q^{\gamma_1} \cdot \mathsf{poly}(\lambda, n, s) + \mathsf{poly}(\lambda, n, s) \\ &\leq q^{\gamma_1} \cdot \mathsf{poly}(\lambda, n, s) \ . \end{aligned} \tag{3.3}$$

Therefore, from inequalities 3.2 and 3.3, we have

$$\mathsf{pad}_{\mathsf{E}} \leq q^{\gamma_1} \cdot \mathsf{poly}_{\mathsf{E}}(\lambda, n, s) \ , \tag{3.4}$$

where $\mathsf{poly}_{\mathsf{E}}$ is some fixed polynomial.

Efficiency. To simplify the efficiency analysis, we assume that we use two different SXIO $\mathsf{sxi}\mathcal{O}$ and $\mathsf{sxi}\mathcal{O}'$. We use $\mathsf{sxi}\mathcal{O}$ to obfuscate $\mathsf{P}_{\mathsf{1Key}}$. We use $\mathsf{sxi}\mathcal{O}'$ to obfuscate $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$.

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

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

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

Then, from inequality 3.4, we can bound the running time of both $\mathsf{CS.Enc}$ and $\mathsf{CS.PEnc}$ by

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

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 $\mathsf{CollSuc}$ is collusion-succinct. From Theorem 2.5.1, 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 $\mathsf{sxi}\mathcal{O}$, 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 $\mathsf{E}_{\mathsf{1Key}}$. A ciphertext output by the punctured encryption algorithm is an obfuscated circuit of $\mathsf{PE}_{\mathsf{1Key}}$. Thus, if we prove that $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$ are functionally equivalent, δ-indistinguishability of functionality under puncturing of $\mathsf{CollSuc}$ holds due to the δ-security of $\mathsf{sxi}\mathcal{O}$.

Note that $\widetilde{\mathsf{P}}$ in $\mathsf{PE}_{\mathsf{1Key}}$ has the exactly same functionality as $\mathsf{P}_{\mathsf{1Key}}$ due to the functionality preserving property of $\mathsf{sxi}\mathcal{O}$. Thus, on input $i \in [q]$, $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$ basically compute the followings:

1. Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$ and $r_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_{\mathsf{Enc}}}(i)$.
2. Compute $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.
3. $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$ respectively computes CT_i as follows:
   - $\mathsf{E}_{\mathsf{1Key}}$ computes $\mathsf{CT}_i \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_i, \mathsf{tag}', m; r_{\mathsf{Enc}})$
   - $\mathsf{PE}_{\mathsf{1Key}}$ computes $\mathsf{CT}_i \leftarrow \mathsf{1Key.PEnc}(\mathsf{MSK}^*_i\{\mathsf{tag}\}, \mathsf{tag}', m; r_{\mathsf{Enc}})$ by using $\mathsf{MSK}^*_i\{\mathsf{tag}\} \leftarrow \mathsf{1Key.Punc}(\mathsf{MSK}_i, \mathsf{tag}; r_{\mathsf{Punc}})$ and $r_{\mathsf{Punc}} \leftarrow \mathsf{F}_{S_{\mathsf{Punc}}}(i)$.
4. Return CT_i.

Recall that OneKey satisfies functionality preserving under puncturing property defined in Definition 3.2.2. Thus, both $\mathsf{E}_{\mathsf{1Key}}$ and $\mathsf{PE}_{\mathsf{1Key}}$ compute the same CT_i as long as $\mathsf{tag}' \neq \mathsf{tag}$ holds and the same S_{Enc} is used in both circuits.

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

Semantic security at punctured tag. Let $\mathcal{A}$ be a valid adversary that attacks the semantic security at punctured tag of $\mathsf{CollSuc}$. We prove it via a sequence of games. Let $\mathtt{SUC}_j$ 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 $\mathsf{CollSuc}$. Then, we have $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{CollSuc},\mathcal{A}}(\lambda) = 2|\Pr[\mathtt{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger generates $S \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and sets $\mathsf{MSK} \leftarrow S$. The challenger also chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. The challenger sends security parameter 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\mathsf{tag} \in \mathcal{T}$, and $\{f_i\}_{i\in[q]}$ to the challenger.
3. The challenger generates $S_{\mathsf{Enc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and computes $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m])$.

Next, for every $i \in [q]$, the challenger computes as follows. The challenger computes $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i)$, $\mathsf{MSK}_i \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$, and $\mathsf{1Key.sk}_{f_i} \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f_i)$. The challenger sets $\mathsf{sk}_{f_i} \leftarrow (i, \mathsf{1Key.sk}_{f_i})$.

Then, the challenger generates $S_{\mathsf{Punc}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\widetilde{\mathsf{P}} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}])$. Then, the challenger sets $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \widetilde{\mathsf{P}}$.

The challenger returns $(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT}, \{\mathsf{sk}_{f_i}\}_{i\in[q]})$ to $\mathcal{A}$.

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

Then, for every $i^* \in [q]$, we define the following games. We define Game $(6,0)$ as the same game as Game 0. Let $\mathtt{SUC}_{(\ell,i^*)}$ denote the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game (ℓ, i^*) for every $\ell \in \{1,\ldots,6\}$ and $i^* \in [q]$.

Game $(1,i^*)$ Same as Game $(6,i^*-1)$ except the followings. The challenger generates $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}^{i^*}_{\mathsf{1Key}}[S^*\{i^*\}, S^*_{\mathsf{Enc}}\{i^*\}, \mathsf{tag}, m_b, m_1, \mathsf{CT}_{i^*}])$, where $\mathsf{CT}_{i^*} \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_{i^*}, \mathsf{tag}, m_b; r^{i^*}_{\mathsf{Enc}})$, $\mathsf{MSK}_{i^*} \leftarrow \mathsf{1Key.Setup}(1^\lambda; r^{i^*}_{\mathsf{Setup}})$, $r^{i^*}_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i^*)$, and $r^{i^*}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_{\mathsf{Enc}}}(i^*)$.

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 $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}^{i^*-1}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot])$. However, we see that $\mathsf{E}^{i^*-1}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot]$ and $\mathsf{E}^{i^*}_{\mathsf{1Key}}[S^*\{i^*\}, S^*_{\mathsf{Enc}}\{i^*\}, \mathsf{tag}, m_b, m_1, \mathsf{CT}_{i^*}]$ have exactly the same functionality. Therefore, by the indistinguishability guarantee of $\mathsf{sxi}\mathcal{O}$, we have $\left|\Pr[\mathtt{SUC}_{(6,i^*-1)}] - \Pr[\mathtt{SUC}_{(1,i^*)}]\right| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

Game $(2,i^*)$ Same as Game $(1,i^*)$ except the followings. The challenger generates $\widetilde{\mathsf{P}} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}^{i^*}_{\mathsf{1Key}}[S^*\{i^*\}, S^*_{\mathsf{Punc}}\{i^*\}, \mathsf{tag}, \mathsf{MSK}^*_{i^*}\{\mathsf{tag}\}])$, where $\mathsf{MSK}^*_{i^*}\{\mathsf{tag}\} \leftarrow \mathsf{1Key.Punc}(\mathsf{MSK}_{i^*}, \mathsf{tag}; r^{i^*}_{\mathsf{Punc}})$.

Similarly to the analysis between Game $(6,i^*-1)$ and $(1,i^*)$, due to the indistinguishability guarantee of $\mathsf{sxi}\mathcal{O}$ and the fact that $\mathsf{P}_{\mathsf{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}]$ and $\mathsf{P}^{i^*}_{\mathsf{1Key}}[S^*\{i^*\}, S^*_{\mathsf{Punc}}\{i^*\}, \mathsf{tag}, \mathsf{MSK}^*_{i^*}\{\mathsf{tag}\}]$ have the same functionality, $\left|\Pr[\mathtt{SUC}_{(1,i^*)}] - \Pr[\mathtt{SUC}_{(2,i^*)}]\right| \leq \delta^{\Omega(1)}$ holds for every $i^* \in [q]$.

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

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

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

In both Game $(3,i^*)$ and $(4,i^*)$, all of MSK_{i^*}, $\mathsf{MSK}^*_{i^*}\{\mathsf{tag}\}$, 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 $\left|\Pr[\mathtt{SUC}_{(3,i^*)}] - \Pr[\mathtt{SUC}_{(4,i^*)}]\right| \leq \delta^{\Omega(1)}$ for every $i^* \in [q]$.

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

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

Game $(6, i^*)$ Same as Game $(5, i^*)$ except that the challenger generates $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}^{i^*}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot])$ and $\widetilde{\mathsf{P}} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{P}_{\mathsf{1Key}}[S, S_{\mathsf{Punc}}, \mathsf{tag}])$.

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

We define one additional game.

Game 7 Same as Game $(6, q)$ except the followings. The challenger generates $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_1])$.

In Game $(6, q)$, CT is generated by $\mathsf{CT} \leftarrow \mathsf{sxi}\mathcal{O}(\mathsf{E}^{q}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot])$. $\mathsf{E}^{q}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot]$ always ignores m_b and outputs a ciphertext of m_1. Therefore, $\mathsf{E}^{q}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_b, m_1, \bot]$ and $\mathsf{E}_{\mathsf{1Key}}[S, S_{\mathsf{Enc}}, \mathsf{tag}, m_1]$ have the same functionality. Thus, we have $\left|\Pr[\mathtt{SUC}_{(6,q)}] - \Pr[\mathtt{SUC}_7]\right| \leq \delta^{\Omega(1)}$ from the indistinguishability guarantee of $\mathsf{sxi}\mathcal{O}$.

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

$$\begin{aligned}
\frac{1}{2}\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{CollSuc},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_7]| \\
&\leq \left|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_{(1,1)}]\right| + \sum_{i^* \in [q]} \sum_{\ell=1}^{5} \left|\Pr[\mathtt{SUC}_{(\ell,i^*)}] - \Pr[\mathtt{SUC}_{(\ell+1,i^*)}]\right| \\
&\qquad + \left|\Pr[\mathtt{SUC}_{(6,q)}] - \Pr[\mathtt{SUC}_7]\right| \quad . \qquad (3.5)
\end{aligned}$$

From the above argument, each term of the right side of inequality 3.5 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{CollSuc},\mathcal{A}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of $\mathcal{A}$ is arbitrary, $\mathsf{CollSuc}$ satisfies δ-semantic security at punctured tag. □ (**Theorem 3.4.1**)

3.4.2 From Collusion-Succinct to Weakly-Succinct

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

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

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

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

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

- Generate $K \xleftarrow{\mathsf{r}} \{0,1\}^{\lambda}$ and $t \leftarrow \{0,1\}^{\lambda}$.
- Compute $\hat{f} \leftarrow \mathsf{RE}(1^{\lambda}, f)$ and decomposed encodings $\hat{f}_1, \cdots \hat{f}_{\mu}$ together with sets of integers $(R_1, \cdots, R_{\mu})$. R_i indicates which bit of a randomness $\hat{f}_i$ depends on for every $i \in [\mu]$. Note that $R_i \subseteq [\rho]$ and $|R_i| = c$ for every $i \in [\mu]$.
- Generate $\mathsf{CT}_i^{\mathsf{ske}} \leftarrow \mathsf{E}(K, 0^{|\hat{f}_i(\cdot,\cdot)|})$, and compute $\mathsf{sk}_{\mathsf{En}_i} \leftarrow \mathsf{CS.KG}(\mathsf{MSK}, \mathsf{En}_{\mathsf{dre}}[\hat{f}_i, R_i, t, \mathsf{CT}_i^{\mathsf{ske}}], i)$ for every $i \in [\mu]$. $\mathsf{En}_{\mathsf{dre}}$ defined in Figure 3.12.
- Return $\mathsf{sk}_f \leftarrow (\mathsf{sk}_{\mathsf{En}_1}, \cdots, \mathsf{sk}_{\mathsf{En}_{\mu}})$.

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

- Generate $S_{\mathsf{encd}} \leftarrow \{0,1\}^{\lambda}$.

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

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

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

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

Figure 3.12: The description of $\mathsf{En}_{\mathsf{dre}}$.

- Return $\mathsf{CT} \leftarrow \mathsf{CS.Enc}(\mathsf{MSK}, \mathsf{tag}, (m, S_{\mathsf{encd}}, \bot))$.

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

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

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

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

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

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

Theorem 3.4.2 *Let $\delta(\cdot)$ be negligible function. Let* CollSuc *be (δ, δ)-secure puncturable SKFE with indistinguishability of functionality that can issue μ functional keys and is collusion-succinct with compression factor γ, where $\gamma < 1$ is a constant. Let* RE, SKE, *and* F *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 3.4.2. We start with analyzing the weak succinctness of WeakSuc, and then move on to the security proof.

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

$$\mu^\gamma \cdot \mathsf{poly}(\lambda, n, \left|\mathsf{En}^i_{\mathsf{dre}}\right|) \leq (s \cdot \mathsf{poly}_{\mathsf{RE}}(\lambda, n))^\gamma \cdot \mathsf{poly}(\lambda, n, \mathsf{poly}_{\mathsf{En}}(\lambda, n, \log s)) \quad \leq s^{\gamma'} \cdot \mathsf{poly}(\lambda, n) \ ,$$

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

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

Semantic security at punctured tag. Let $\mathcal{A}$ be an adversary that attacks the semantic security at punctured tag of $\mathsf{WeakSuc}$. We prove it via sequence of games. Below, for every $\ell \in \{0, \ldots, 4\}$, let $\mathtt{SUC}_\ell$ 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 $\mathsf{WeakSuc}$. Then, we have $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{WeakSuc},\mathcal{A}}(\lambda) = 2|\Pr[\mathtt{SUC}_0] - \frac{1}{2}|$. The detailed description is as follows.

1. The challenger generate $S_{\mathsf{encd}} \leftarrow \{0,1\}^\lambda$ and computes $\mathsf{CT} \leftarrow \mathsf{CS.Enc}(\mathsf{MSK}, \mathsf{tag}, (m_b, S_{\mathsf{encd}}, \bot))$. The challenger sends security parameter 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M} \times \mathcal{M}$, $\mathsf{tag} \in \mathcal{T}$, and a function f to the challenger.
3. The challenger generates $\mathsf{MSK} \leftarrow \mathsf{CS.Setup}(1^\lambda)$. The challenger also chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$.

   Next, the challenger generates $K \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $t \leftarrow \{0,1\}^\lambda$, and computes $\hat{f} \leftarrow \mathsf{RE}(1^\lambda, f)$ and decomposed encodings $\hat{f}_1 \cdots \hat{f}_\mu$ together with sets $(R_1, \cdots, R_\mu)$. Then, the challenger generates $\mathsf{CT}^{\mathsf{ske}}_i \leftarrow \mathsf{E}(K, 0^{|\hat{f}_i(\cdot,\cdot)|})$, and computes $\mathsf{sk}_{\mathsf{En}_i} \leftarrow \mathsf{CS.KG}(\mathsf{MSK}, \mathsf{En}_{\mathsf{dre}}[\hat{f}_i, R_i, t, \mathsf{CT}^{\mathsf{ske}}_i], i)$ for every $i \in [\mu]$. Moreover, the challenger sets $\mathsf{sk}_f \leftarrow (\mathsf{sk}_{\mathsf{En}_1}, \cdots, \mathsf{sk}_{\mathsf{En}_\mu})$.

   Then, the challenger computes $\mathsf{MSK}^*\{\mathsf{tag}\} \leftarrow \mathsf{CS.Punc}(\mathsf{MSK}, \mathsf{tag})$.

   The challenger returns $(\mathsf{MSK}^*\{\mathsf{tag}\}, \mathsf{CT}, \mathsf{sk}_f)$ to $\mathcal{A}$.

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

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

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 $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \delta^{\Omega(1)}$.

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

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

$$\mathsf{En}_{\mathsf{dre}}[\hat{f}_i, R_i, t, \mathsf{CT}_i^{\mathsf{ske}}](m_b, S_{\mathsf{encd}}, \bot) = \hat{f}_i(m_b; r_{R_i}) = \mathsf{En}_{\mathsf{dre}}[\hat{f}_i, R_i, t, \mathsf{CT}_i^{\mathsf{ske}}](\bot, \bot, K).$$

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

Game 3 Same as Game 2 except that the challenger generates r_j as a truly random string for every $j \in [\rho]$.

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

Game 4 Same as Game 3 except that the challenger generates $\{e_i\}_{i\in[\mu]} \leftarrow \mathsf{Sim}(1^\lambda, s, y)$, where 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 $\mathsf{En}_{\mathsf{dre}}$ after encrypted is generated with a truly random string. Therefore, from the security guarantee of RE, we have $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| \leq \delta^{\Omega(1)}$.

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

$$\begin{aligned}
\frac{1}{2}\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{WeakSuc},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_4]| \\
&\leq \sum_{\ell=0}^{3} |\Pr[\mathtt{SUC}_\ell] - \Pr[\mathtt{SUC}_{\ell+1}]| \quad . \qquad (3.6)
\end{aligned}$$

From the above argument, each term of the right side of inequality 3.6 is bounded by $\delta^{\Omega(1)}$. Therefore, we see that $\mathsf{Adv}^{\mathsf{ss}}_{\mathsf{WeakSuc},\mathcal{A}}(\lambda) \leq \delta^{\Omega(1)}$. Since the choice of $\mathcal{A}$ is arbitrary, $\mathsf{WeakSuc}$ satisfies δ-semantic security at punctured tag. □ (**Theorem 3.4.2**)

3.5 Indistinguishability Obfuscation from Puncturable SKFE

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

Theorem 3.5.1 *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 Theorem 2.5.1, 3.3.1, 3.4.1, and 3.4.2, we also obtain the following theorem.

Theorem 3.5.2 *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 3.5.2, we also use δ-secure PRF, puncturable PRF, plain SKE, garbling scheme, and decomposable randomized encoding as building blocks. From Theorem 2.2.1, 2.2.2, 2.2.3, 2.2.4, and 2.2.5, 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 Theorem 3.5.1 and 3.5.2, we obtain the following main theorem.

Theorem 3.5.3 *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 3.5.1 (IO for circuits with input of poly-logarithmic length) *The security loss of our IO construction is exponential in the input length of circuits, but is independent of the size of circuits. Thus, if the input length of circuits is poly-logarithmic in the security parameter, our IO construction incurs only quasi-polynomial security loss regardless of the size of circuits. Therefore, we can obtain IO for circuits of* polynomial size *with input of poly-logarithmic length from* quasi-polynomially secure *collusion-resistant SKFE for all circuits. This is an improvement over the IO construction by Komargodski and Segev [KS17]. 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 3.5.1, we first give the construction of IO based on puncturable SKFE. Then, we analyze its security and efficiency.

3.5.1 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 3.2.4 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 $\mathsf{pSKFE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Punc}, \mathsf{PEnc})$ be a single-key weakly-succinct puncturable SKFE scheme. Let $\mathsf{SKE} = (\mathsf{E}, \mathsf{D})$ be an SKE scheme and $\mathsf{PPRF} = (\mathsf{F}, \mathsf{Punc}_{\mathsf{F}})$ 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 $\mathsf{i}\mathcal{O}$ as follows.

Given a circuit $C : \{0,1\}^n \to \{0,1\}^m$ and a security parameter λ, the obfuscator $\mathsf{i}\mathcal{O}$ first sets the security parameter $\tilde{\lambda}$ for building block schemes as $\tilde{\lambda} = \omega((n^2 + \log \lambda)^{1/\epsilon})$. $\mathsf{i}\mathcal{O}$ 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. $\mathsf{i}\mathcal{O}$ 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. $\mathsf{i}\mathcal{O}$ invokes the following recursive obfuscation procedure $\mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ in order to obfuscate C.

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

- If $i = 1$, return $\widetilde{C}_i \leftarrow (C_i(0), C_i(1))$.
- Else, runs as follows:
  - Generate $K_{i,0}, K_{i,1} \xleftarrow{\mathsf{r}} \{0,1\}^{\tilde{\lambda}}$ and compute $\mathsf{CT}^{\mathsf{ske}}_{i,0} \leftarrow \mathsf{E}(K_{i,0}, C_i)$ and $\mathsf{CT}^{\mathsf{ske}}_{i,1} \leftarrow \mathsf{E}(K_{i,1}, C_i)$.
  - Generate $\mathsf{MSK}_i \leftarrow \mathsf{Setup}(1^{\tilde{\lambda}})$ and compute $\mathsf{sk}_{\mathsf{Ev}_i} \leftarrow \mathsf{KG}(\mathsf{MSK}_i, \mathsf{Ev}_i[\mathsf{CT}^{\mathsf{ske}}_{i,0}, \mathsf{CT}^{\mathsf{ske}}_{i,1}])$. The circuit Ev_i is defined in Figure 3.13.

Evaluation Circuit $\mathsf{Ev}_i[\mathsf{CT}^{\mathsf{ske}}_{i,0}, \mathsf{CT}^{\mathsf{ske}}_{i,1}](\boldsymbol{x}_i, K, \alpha)$

Hardwired: Two ciphertexts $\mathsf{CT}^{\mathsf{ske}}_{i,0}$ and $\mathsf{CT}^{\mathsf{ske}}_{i,1}$.

Input: A string $\boldsymbol{x}_i \in \{0,1\}^i$, a SKE key K, and a bit $\alpha \in \{0,1\}$.

1. Compute $C_i \leftarrow \mathsf{D}(K, \mathsf{CT}^{\mathsf{ske}}_{i,\alpha})$.
2. Return $U(C_i, \boldsymbol{x}_i)$.

Figure 3.13: The description of Ev_i for every $i \in \{2, \ldots, n\}$. In the description, $U(\cdot,\cdot)$ is an universal circuit.

Encryption Circuit $\mathsf{E}_{i-1,\alpha}[\mathsf{MSK}_i, K_i, S_i](\boldsymbol{x}_{i-1})$

Hardwired: A master secret key MSK_i, a SKE key K_i, and a PRF key S_i.

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

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

1. For $x_i \in \{0,1\}$, compute as follows:
   - Compute $r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_i}(\boldsymbol{x}_{i-1}\|x_i)$.
   - Compute $\mathsf{CT}_{i,x_i} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1}\|x_i, K_i, \alpha); r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}})$.
2. Return $(\mathsf{CT}_{i,0}, \mathsf{CT}_{i,1})$.

Figure 3.14: The description of $\mathsf{E}_{i-1,\alpha}$ for every $i \in \{3, \ldots, n\}$ and $\alpha \in \{0,1\}$.

   - Generate $S_i \xleftarrow{\mathsf{r}} \{0,1\}^{\tilde{\lambda}}$ and compute $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{E}_{i-1,0}[\mathsf{MSK}_i, K_{i,0}, S_i])$. The circuit $\mathsf{E}_{i-1,0}$ is defined in Figure 4.4.
   - Return $\widetilde{C}_i \leftarrow (\mathsf{sk}_{\mathsf{Ev}_i}, \widetilde{\mathsf{E}}_{i-1})$.

The corresponding recursive evaluation procedure is as follows. We can evaluate $C(\boldsymbol{x}_n)$ by invoking $\mathsf{rEval}(n, \widetilde{C}, \boldsymbol{x}_n)$, where $\widetilde{C} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ and $\boldsymbol{x}_n \in \{0,1\}^n$.

$\mathsf{rEval}(i, \widetilde{C}_i, \boldsymbol{x}_i)$:

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

Remark 3.5.2 (On the parameter setting of $\tilde{\lambda}$) *In the construction we set the security parameter $\tilde{\lambda}$ for building blocks as $\tilde{\lambda} = \omega\left((n^2 + \log\lambda)^{1/\epsilon}\right)$. 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 when the building block puncturable SKFE scheme satisfies only indistinguishability of functionality under puncturing property. For the detailed security analysis, see Section 3.5.2.

Note that the size of padding for the encryption circuit $\mathsf{E}_{i-1,\alpha}$ is determined in the security analysis of our indistinguishability obfuscator $\mathsf{i}\mathcal{O}$. We need to know the size of padding in order to analyze the efficiency of $\mathsf{i}\mathcal{O}$. Therefore, we first analyze the security of $\mathsf{i}\mathcal{O}$ in Section 3.5.2. Then, we analyze the efficiency of $\mathsf{i}\mathcal{O}$ in Section 3.5.3. We complete the proof of Theorem 3.5.1 by completing the analysis of security and efficiency.

3.5.2 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

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

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

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

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 3.5.1 *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*

$$\delta_i \leq 2^{2(i-1)} \cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \tag{3.7}$$

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

By this lemma, we can estimate δ_n as

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

This inequality shows that we complete the proof of Theorem 3.5.3.

Therefore, if we prove that inequality 3.7 holds for every $i \in \{2, \cdots, n\}$, that is Lemma 3.5.1, we can conclude that our $\mathsf{i}\mathcal{O}$ is a secure indistinguishability obfuscator. In the rest of this section, we prove that inequality 3.7 holds for every $i \in \{2, \cdots, n\}$.

Let $i \in \{2, \ldots, n\}$. 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 $\mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i, C_{i,0})$.

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

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

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

$\mathcal{H}_4$: Same as $\mathcal{H}_3$ except that $\widetilde{\mathsf{E}}_{i-1}$ is generated as $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{E}_{i-1,0}[\mathsf{MSK}_i, K_{i,0}, S_i])$. 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 $\mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i, C_{i,1})$.

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

$$\delta_i \leq \sum_{\ell=0}^{3} \left|\Pr[\mathcal{D}_i(\mathcal{H}_\ell)] - \Pr[\mathcal{D}_i(\mathcal{H}_{\ell+1})]\right| \quad . \tag{3.8}$$

In the following, by estimating each term of the right hand side of inequality 3.8, we prove that inequality 3.7 holds for every $i \in \{2, \cdots, n\}$. We give relations of hybrid experiments in Figure 3.15 and 3.19 in order to see easily the dependences of hybrid experiments.

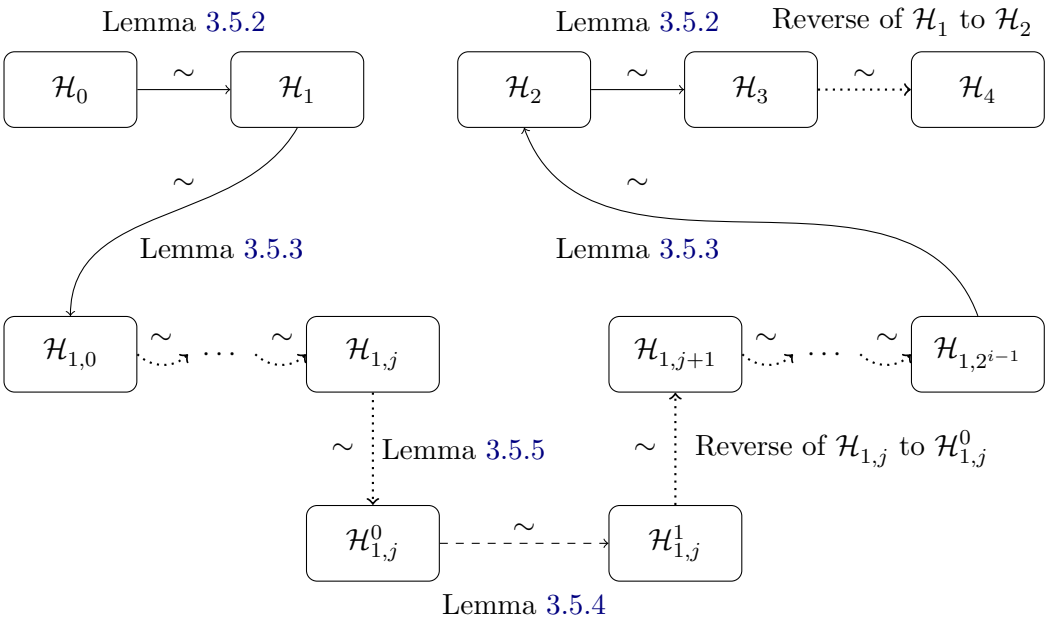

Figure 3.15: Relations of the hybrid experiments from $\mathcal{H}_0$ to $\mathcal{H}_4$ for the security of $\mathsf{ri}\mathcal{O}$. Solid lines denote that the indistinguihability is proven by one step. Dashed lines denote that we use a few hybrid experiments to prove the indistinguishability. Dotted lines denote that we use many hybrid experiments to prove the indistinguishability (Figure 3.19 illustrates those of hybrid experiments for Lemma 3.5.5).

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

Encryption Circuit $\mathsf{E}^{j}_{i-1}[\mathsf{MSK}_i, K_{i,0}, K_{i,1}, S_i](\boldsymbol{x}_{i-1})$

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 $\boldsymbol{x}_{i-1} \in \{0,1\}^{i-1}$.

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

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

Figure 3.16: The description of E^{j}_{i-1}. The red underline is the difference from $\mathsf{E}_{i-1,\alpha}$.

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 $j \in \{0, \cdots, 2^{i-1}\}$. In the following, let $\boldsymbol{j} \in \{0,1\}^{i-1} \cup \{1\|0^{i-1}\}$ be the binary representation of j.

$\mathcal{H}_{1,j}$: In this experiment, $\widetilde{\mathsf{E}}_{i-1}$ is computed as $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{E}^{j}_{i-1}[\mathsf{MSK}_i, K_{i,0}, K_{i,1}, S_i])$. The circuit E^{j}_{i-1} is defined in Figure 3.16.

Then, we have

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

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 size of the above inequality, we need introduce the following additional hybrid experiments for every $j \in \{1, \ldots, 2^{i-1}\}$ since we use a puncturable SKFE scheme.

$\mathcal{H}^0_{1,j}$: $\widetilde{\mathsf{E}}_{i-1}$ is computed as $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{PE}^{j}_{i-1}[\mathsf{MSK}^*_i\{\boldsymbol{j}\}, K_{i,0}, K_{i,1}, S_i\{\boldsymbol{j}\|0, \boldsymbol{j}\|1\}, u_{i,0}, u_{i,1}])$ in this experiment, where $u_{i,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{j}, (\boldsymbol{j}\|b, K_{i,0}, 0); r^{\boldsymbol{j}\|b}_{\mathsf{Enc}})$ and $r^{\boldsymbol{j}\|b}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_i}(\boldsymbol{j}\|b)$ for every $b \in \{0,1\}$. The circuit PE^{j}_{i-1} is defined in Figure 3.17. The other part of this experiment is same as $\mathcal{H}_{1,j}$.

Punctured Encryption Circuit $\mathsf{PE}^{j}_{i-1}[\mathsf{MSK}^{*}_{i}\{\boldsymbol{j}\}, K_{i,0}, K_{i,1}, S^{*}_{i}\{\boldsymbol{j}\|0, \boldsymbol{j}\|1\}, u_{i,0}, u_{i,1}](\boldsymbol{x}_{i-1})$

Hardwired: A punctured master secret key $\mathsf{MSK}^{*}_{i}\{\boldsymbol{j}\}$, two SKE keys $K_{i,0}$ and $K_{i,1}$, a punctured PRF key $S^{*}_{i}\{\boldsymbol{j}\|0, \boldsymbol{j}\|1\}$, and two ciphertexts $u_{i,0}$ and $u_{i,1}$.

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

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

1. If $\boldsymbol{x}_{i-1} = \boldsymbol{j}$, return $(u_{i,0}, u_{i,1})$.
2. Else, set $\alpha \leftarrow 1$ if $\boldsymbol{x}_{i-1} < \boldsymbol{j}$ and $\alpha \leftarrow 0$ otherwise.
3. For $x_i \in \{0,1\}$, compute as follows:
   - Compute $r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S^{*}_{i}\{\boldsymbol{j}\|0,\boldsymbol{j}\|1\}}(\boldsymbol{x}_{i-1}\|x_i)$.
   - Compute $\mathsf{CT}_{i,x_i} \leftarrow \mathsf{PEnc}(\mathsf{MSK}^{*}_{i}\{\boldsymbol{j}\}, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1}\|x_i, K_{i,\alpha}, \alpha); r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}})$.
4. Return $(\mathsf{CT}_{i,0}, \mathsf{CT}_{i,1})$.

Figure 3.17: The description of PE^{j}_{i-1}. Red underlines are differences from E^{j}_{i-1}.

$\mathcal{H}^{1}_{1,j}$: Same as $\mathcal{H}^{0}_{1,j}$ except that $u_{i,b}$ is computed by $u_{i,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}, \boldsymbol{j}, (\boldsymbol{j}\|b, K_{i,1}, 1); r^{\boldsymbol{j}\|b}_{\mathsf{Enc}})$ and $r^{\boldsymbol{j}\|b}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_i}(\boldsymbol{j}\|b)$ 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

$$\begin{aligned}
|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]| \leq{}& \left|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})]\right| \\
&+ \sum_{j=0}^{2^{i-1}-1} \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^{0}_{1,j})]\right| \\
&+ \sum_{j=0}^{2^{i-1}-1} \left|\Pr[\mathcal{D}_i(\mathcal{H}^{0}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^{1}_{1,j})]\right| \\
&+ \sum_{j=0}^{2^{i-1}-1} \left|\Pr[\mathcal{D}_i(\mathcal{H}^{1}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})]\right| \\
&+ \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]\right| \quad . \qquad (3.9)
\end{aligned}$$

In the rest of this section, we estimate each term of the right side of inequality 3.9 by Lemma 3.5.3, 3.5.4, and 3.5.5.

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 3.5.3 $\left|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})]\right| \leq \delta_{i-1}$ *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 3.5.3. 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 $\widetilde{\mathsf{E}}_{i-1}$ is generated. In $\mathcal{H}_1$, $\widetilde{\mathsf{E}}_{i-1}$ is generated by $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{E}_{i-1,0}[\mathsf{MSK}_i, K_{i,0}, S_i])$. On the other hand, in $\mathcal{H}_{1,0}$, $\widetilde{\mathsf{E}}_{i-1}$ is generated by $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{E}^0_{i-1}[\mathsf{MSK}_i, K_{i,0}, K_{i,1}, S_i])$. We can see that circuits obfuscated in each experiment have the same functionality. Moreover, both circuits are padded to the same size $\mathsf{pad}^{i-1}_{\mathsf{E}}$. Therefore, we have $\left|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,0})]\right| \leq \delta_{i-1}$.

Similarly, we also obtain $\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]\right| \leq \delta_{i-1}$. □ (**Lemma 3.5.3**)

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

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

Lemma 3.5.4 *Let $\delta(\tilde{\lambda}) = 2^{-\tilde{\lambda}^\epsilon}$. Let pSKFE satisfy δ-semantic security at punctured tag. Let PPRF be δ-secure. Then, $\left|\Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^1_{1,j})]\right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ holds for every $i \in \{0, \ldots, 2^{i-1}-1\}$.*

Proof of Lemma 3.5.4. In order to use the semantic security at punctured tag of pSKFE, we change the randomness $r^{\boldsymbol{j}\|b}_{\mathsf{Enc}}$ used in $u_{i,b}$ into truly random for every $b \in \{0,1\}$. Thus, we introduce the following intermediate hybrid experiments $\mathcal{H}^{0,\mathsf{rnd}}_{1,j}$ and $\mathcal{H}^{1,\mathsf{rnd}}_{1,j}$ between $\mathcal{H}^0_{1,j}$ and $\mathcal{H}^1_{1,j}$.

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

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

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

Note that the only difference between $\mathcal{H}^{0,\mathsf{rnd}}_{1,j}$ and $\mathcal{H}^{1,\mathsf{rnd}}_{1,j}$ is how $u_{i,b}$ is generated for every $b \in \{0,1\}$. In $\mathcal{H}^{0,\mathsf{rnd}}_{1,j}$, $u_{i,b}$ is generated by $u_{i,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{j}, (\boldsymbol{j}\|b, K_{i,0}, 0))$. On the other hand, in $\mathcal{H}^{1,\mathsf{rnd}}_{1,j}$, $u_{i,b}$ is generated by $u_{i,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{j}, (\boldsymbol{j}\|b, K_{i,1}, 1))$. 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, $\mathsf{CT}^{\mathsf{ske}}_{i,\alpha}$ is a ciphertext of $C_{i,\alpha}$ under the SKE key $K_{i,\alpha}$ for every $\alpha \in \{0,1\}$. Hence, we have

$$\mathsf{Ev}_i[\mathsf{CT}^{\mathsf{ske}}_{i,0}, \mathsf{CT}^{\mathsf{ske}}_{i,1}](\boldsymbol{j}\|b, K_{i,0}, 0) = C_{i,0}(\boldsymbol{j}\|b) = C_{i,1}(\boldsymbol{j}\|b) = \mathsf{Ev}_i[\mathsf{CT}^{\mathsf{ske}}_{i,0}, \mathsf{CT}^{\mathsf{ske}}_{i,1}](\boldsymbol{j}\|b, K_{i,1}, 1),$$

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

Therefore, we obtain $\left|\Pr[\mathcal{D}_i(\mathcal{H}^{0,\mathsf{rnd}}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^{1,\mathsf{rnd}}_{1,j})]\right| \leq 2^{-\tilde{\lambda}^\epsilon}$ from the semantic security at punctured tag of pSKFE.

By the pseudorandomness of PPRF, we have $\left|\Pr[\mathcal{D}_i(\mathcal{H}^{1,\mathsf{rnd}}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^{1}_{1,j})]\right| \leq 2^{-\tilde{\lambda}^\epsilon}$. From these, we see that $\left|\Pr[\mathcal{D}_i(\mathcal{H}^{0}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^{1}_{1,j})]\right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$. □ (**Lemma 3.5.4**)

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

Lemma 3.5.5 *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*

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

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

Proof of Lemma 3.5.5. If the underlying puncturable SKFE satisfies functionality preserving under puncturing property, we can directly estimate $\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})]\right|$ 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 $\mathcal{H}^0_{1,j}$ defined for every $k \in \{0,\dots,2^{i-1}\}$ as follows. Below, let $\boldsymbol{k} \in \{0,1\}^{i-1} \cup \{1\|0^{i-1}\}$ be the binary representation of k.

$\mathcal{H}_{1,j,k}$: $\widetilde{\mathsf{E}}_{i-1}$ is computed as $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{HE}^{j,k}_{i-1}[\mathsf{MSK}_i, \mathsf{MSK}^*_i\{\boldsymbol{j}\}, K_{i,0}, K_{i,1}, S_i, \bot, \bot])$ in this experiment. The circuit $\mathsf{HE}^{j,k}_{i-1}$ is defined in Figure 3.18.

Figure 3.19 illustrates an overview of hybrid experiments from $\mathcal{H}_{1,j-1}$ to $\mathcal{H}^0_{1,j}$.

Then, we have

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

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

$\mathcal{H}^{\mathsf{enc}}_{1,j,k}$: $\widetilde{\mathsf{E}}_{i-1}$ is computed as $\widetilde{\mathsf{E}}_{i-1} \leftarrow \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, i-1, \mathsf{HE}^{j,k}_{i-1}[\mathsf{MSK}_i, \mathsf{MSK}^*_i\{\boldsymbol{j}\}, K_{i,0}, K_{i,1}, S^*_i\{\boldsymbol{k}\|0, \boldsymbol{k}\|1\}, v_{k,0}, v_{k,1}])$ in this experiment, where $v_{k,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{k}, (\boldsymbol{k}\|b, K_{i,\alpha}, \alpha); r^{\boldsymbol{k}\|b}_{\mathsf{Enc}})$ for every $b \in \{0,1\}$.

Hybrid Encryption Circuit $\mathsf{HE}^{j,k}_{i-1}[\mathsf{MSK}_i, \mathsf{MSK}^*_i\{\boldsymbol{j}\}, K_{i,0}, K_{i,1}, S_i, v_{k,0}, v_{k,1}](\boldsymbol{x}_{i-1})$

Hardwired: A master secret key MSK_i, punctured master secret key $\mathsf{MSK}^*_i\{\boldsymbol{j}\}$, two SKE keys $K_{i,0}$ and $K_{i,1}$, PRF key S_i, and two ciphertexts $v_{k,0}$ and $v_{k,1}$.

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

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

1. If $(v_{i,0}, v_{i,1}) \neq (\bot, \bot)$ and $\boldsymbol{x}_{i-1} = \boldsymbol{k}$, return $(v_{k,0}, v_{k,1})$.
2. Else, compute as follows:
   - If $\boldsymbol{x}_{i-1} < \boldsymbol{j}$, set $\alpha \leftarrow 1$, and otherwise, set $\alpha \leftarrow 0$.
   - For $x_i \in \{0,1\}$, compute as follows:
     - Compute $r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}} \leftarrow \mathsf{F}_{S_i}(\boldsymbol{x}_{i-1}\|x_i)$.
     - If $\boldsymbol{x}_{i-1} < \boldsymbol{k}$ and $\boldsymbol{x}_{i-1} \neq \boldsymbol{j}$, then compute $\mathsf{CT}_{i,x_i} \leftarrow \mathsf{PEnc}(\mathsf{MSK}^*_i\{\boldsymbol{j}\}, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1}\|x_i, K_{i,\alpha}, \alpha); r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}})$. Otherwise, compute $\mathsf{CT}_{i,x_i} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{x}_{i-1}, (\boldsymbol{x}_{i-1}\|x_i, K_{i,\alpha}, \alpha); r^{\boldsymbol{x}_{i-1}\|x_i}_{\mathsf{Enc}})$.
   - Return $(\mathsf{CT}_{i,0}, \mathsf{CT}_{i,1})$.

Figure 3.18: The description of $\mathsf{HE}^{j,k}_{i-1}$. Red underlines are differences from E^j_{i-1}.

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

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

$$\begin{aligned}
\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})]\right| \leq{}& \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,0})]\right| \\
&+ \sum_{k\in\{0,\ldots,2^{i-1}-1\}\setminus\{j\}} \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{enc}}_{1,j,k})]\right| \\
&+ \sum_{k\in\{0,\ldots,2^{i-1}-1\}\setminus\{j\}} \left|\Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{enc}}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{penc}}_{1,j,k})]\right| \\
&+ \sum_{k\in\{0,\ldots,2^{i-1}-1\}\setminus\{j\}} \left|\Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{penc}}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k+1})]\right| \\
&+ \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,j+1})]\right| \\
&+ \left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j,2^{i-1}})] - \Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})]\right| \quad . \qquad (3.10)
\end{aligned}$$

In order to bound the right hand side of inequality 3.10, we prove Lemma 3.5.6, 3.5.7, and 3.5.8.

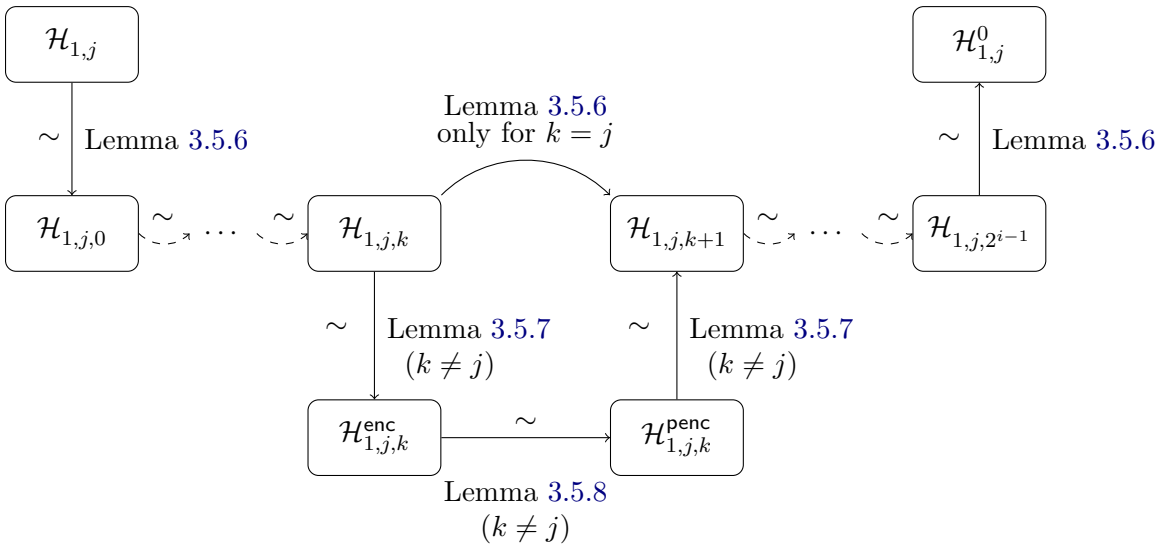

Figure 3.19: Relations of the hybrid experiments from $\mathcal{H}_{1,j-1}$ to $\mathcal{H}^0_{1,j}$ for Lemma 3.5.5. Solid lines denote that the indistinguihability is proven by one step. Dashed lines denote that we use for loop with variable k. Note that, only for $k = j$, we do not need $\mathcal{H}^{\mathsf{enc}}_{1,j,k}$ and $\mathcal{H}^{\mathsf{penc}}_{1,j,k}$ ($\mathcal{H}^{\mathsf{penc}}_{1,j,j}$ is not defined).

Lemma 3.5.6 *It holds that*

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

Lemma 3.5.7 *For $k \in \{0, \ldots, 2^{i-1} - 1\} \setminus \{j\}$, it holds that*

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

Proof of Lemma 3.5.6. The only difference between two experiments in each inequality is how $\widetilde{\mathsf{E}}_{i-1}$ is generated. Thus, we verify that circuits of the same functionality are obfuscated to generate $\widetilde{\mathsf{E}}_{i-1}$ in those two experiments related to each inequality.

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

We next show the second inequality. For $\boldsymbol{x}_{i-1} \leq \boldsymbol{j} - 1$ and $\boldsymbol{x}_{i-1} \geq \boldsymbol{j} + 1$, the behavior of $\mathsf{HE}^{j,j}_{i-1}$ and $\mathsf{HE}^{j,j+1}_{i-1}$ are the same. On input $\boldsymbol{x}_{i-1} = \boldsymbol{j}$, $\mathsf{HE}^{j,k}_{i-1}$ always computes output ciphertexts by using Enc with MSK_i regardless of the value of k, and thus $\mathsf{HE}^{j,j}_{i-1}$ and

$\mathsf{HE}_{i-1}^{j,j+1}$ behave in exactly the same way on input $\boldsymbol{x}_{i-1} = \boldsymbol{j}$. Thus, $\mathsf{HE}_{i-1}^{j,j}$ and $\mathsf{HE}_{i-1}^{j,j+1}$ are functionally equivalent and the second inequality holds due to the security of $\mathsf{ri}\mathcal{O}$.

We finally show the third inequality. On input $\boldsymbol{x}_{i-1}$, if $\boldsymbol{x}_{i-1} = \boldsymbol{j}$, PE_{i-1}^{j} in $\mathcal{H}_{1,j}^{0}$ outputs hardwired $u_{i,0}$ and $u_{i,1}$ that are ciphertexts of $\boldsymbol{j}$ generated by using Enc with MSK_i. Otherwise, it generates ciphertexts of $\boldsymbol{x}_{i-1}$ using PEnc and $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$, and outputs them. On input $\boldsymbol{x}_{i-1}$, $\mathsf{HE}_{i-1}^{j,2^{i-1}}$ in $\mathcal{H}_{1,j,2^{i-1}}^{0}$ computes output ciphertexts by using Enc with MSK_i if $\boldsymbol{x}_{i-1} = \boldsymbol{j}$, and by using PEnc with $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$ otherwise. From this fact, we see that $\mathsf{HE}_{i-1}^{j,2^{i-1}}$ 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 $\mathsf{ri}\mathcal{O}$. □ (**Lemma 3.5.6**)

Proof of Lemma 3.5.7. We first show the first inequality. On input $\boldsymbol{x}_{i-1} \neq \boldsymbol{k}$, $\mathsf{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}$ runs in exactly the same way as $\mathsf{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\mathsf{enc}}$. On input $\boldsymbol{x}_{i-1} = \boldsymbol{k}$, $\mathsf{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}$ generate ciphertexts of k by using Enc with MSK_i, and outputs them. On input $\boldsymbol{x}_{i-1} = \boldsymbol{k}$, $\mathsf{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\mathsf{enc}}$ outputs hardwired $v_{k,0}$ and $v_{k,1}$ that are ciphertexts of $\boldsymbol{k}$ generated by using Enc with MSK_i. Thus, these circuits are functionally equivalent and the inequality holds due to the security of $\mathsf{ri}\mathcal{O}$.

We next show the second inequality. In $\mathcal{H}_{1,j,k}^{\mathsf{penc}}$, $\widetilde{\mathsf{E}}_{i-1}$ is generated by obfuscating $\mathsf{HE}_{i-1}^{j,k}$ that has hardwired ciphertexts of $\boldsymbol{k}$, that is $v_{k,0}$ and $v_{k,1}$ generated by using PEnc with $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$. In $\mathcal{H}_{1,j,k+1}$, $\widetilde{\mathsf{E}}_{i-1}$ is generated by obfuscating $\mathsf{HE}_{i-1}^{j,k+1}$ that does not have hardwired ciphertexts. On input $\boldsymbol{x}_{i-1} \neq \boldsymbol{k}$, these two circuits runs in exactly the same way. On input $\boldsymbol{x}_{i-1} = \boldsymbol{k}$, $\mathsf{HE}_{i-1}^{j,k}$ in $\mathcal{H}_{1,j,k}^{\mathsf{penc}}$ outputs hardwired $v_{k,0}$ and $v_{k,1}$. On input $\boldsymbol{x}_{i-1} = \boldsymbol{k}$, $\mathsf{HE}_{i-1}^{j,k+1}$ in $\mathcal{H}_{1,j,k+1}$ generates ciphertexts of $\boldsymbol{k}$ by using PEnc with $\mathsf{MSK}_i^*\{\boldsymbol{j}\}$, and outputs them. Thus, two circuits are functionally equivalent and the second inequality holds due to the security of $\mathsf{ri}\mathcal{O}$. □ (**Lemma 3.5.7**)

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

Lemma 3.5.8 *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 $\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\mathsf{enc}})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j,k}^{\mathsf{penc}})]\right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ for every $j \in \{0, \ldots, 2^{i-1} - 1\} \setminus \{j\}$.*

Proof of Lemma 3.5.8. Let k be any integer in $\{0, \ldots, 2^{i-1} - 1\} \setminus \{j\}$. In order to use the indistinguishability of functionality under puncturing of pSKFE, we change the randomness $r_{\mathsf{Enc}}^{\boldsymbol{k}\|b}$ used in $v_{k,b}$ into truly random for every $b \in \{0, 1\}$. Thus, we introduce the following intermediate hybrid experiments $\mathcal{H}_{1,j,k}^{\mathsf{enc,rnd}}$ and $\mathcal{H}_{1,j,k}^{\mathsf{penc,rnd}}$ between $\mathcal{H}_{1,j,k}^{\mathsf{enc}}$ and $\mathcal{H}_{1,j,k}^{\mathsf{penc}}$.

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

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

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

Note that the only difference between $\mathcal{H}^{\mathsf{enc,rnd}}_{1,j,k}$ and $\mathcal{H}^{\mathsf{penc,rnd}}_{1,j,k}$ is how $v_{k,b}$ is generated for every $b \in \{0,1\}$. In $\mathcal{H}^{\mathsf{enc,rnd}}_{1,j,k}$, $v_{k,b}$ is generated by $v_{k,b} \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, \boldsymbol{k}, (\boldsymbol{k}\|b, K_{i,\alpha}, \alpha))$. On the other hand, in $\mathcal{H}^{\mathsf{penc,rnd}}_{1,j,k}$, $v_{k,b}$ is generated by $v_{k,b} \leftarrow \mathsf{PEnc}(\mathsf{MSK}^*_i\{\boldsymbol{j}\}, \boldsymbol{k}, (\boldsymbol{k}\|b, K_{i,\alpha}, \alpha))$. 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 $\left|\Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{enc,rnd}}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{penc,rnd}}_{1,j,k})]\right| \leq 2^{-\tilde{\lambda}^\epsilon}$.

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

From the above, we obtain $\left|\Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{enc}}_{1,j,k})] - \Pr[\mathcal{D}_i(\mathcal{H}^{\mathsf{penc}}_{1,j,k})]\right| \leq 2^{-\Omega(\tilde{\lambda}^\epsilon)}$ for every $k \in \{0,\ldots,2^{i-1}-1\} \setminus \{j\}$. □ (**Lemma 3.5.8**)

From inequality 3.10, and Lemma 3.5.6, 3.5.7, and 3.5.8, it holds that

$$\begin{aligned}\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})]\right| &\leq 3\cdot\delta_{i-1} + 2(2^{i-1}-1)\cdot\delta_{i-1} + (2^{i-1}-1)\cdot 2^{-\Omega(\tilde{\lambda}^\epsilon)} \\ &\leq 2^{i-1}\cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \ . \end{aligned} \tag{3.11}$$

This completes the estimation of $\left|\Pr[\mathcal{D}_i(\mathcal{H}_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}^0_{1,j})]\right|$ for every $j \in \{0,\ldots,2^{i-1}-1\}$, that is, completes the proof of Lemma 3.5.5. □ (**Lemma 3.5.5**)

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

$$\left|\Pr[\mathcal{D}_i(\mathcal{H}^1_{1,j})] - \Pr[\mathcal{D}_i(\mathcal{H}_{1,j+1})]\right| \leq 2^{i-1}\cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \tag{3.12}$$

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

From inequality 3.11 and 3.12, inequality 3.9, and Lemma 3.5.3 and 3.5.4, we have

$$\begin{aligned}|\Pr[\mathcal{D}_i(\mathcal{H}_1)] - \Pr[\mathcal{D}_i(\mathcal{H}_2)]| &\leq 2\cdot\delta_{i-1} + 2\cdot 2^{i-1}\cdot 2^{i-1}\cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) + 2^{i-1}\cdot 2^{-\Omega(\tilde{\lambda}^\epsilon)} \\ &\leq 2^{2(i-1)}\cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \ . \end{aligned} \tag{3.13}$$

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

$$|\Pr[\mathcal{D}_i(\mathcal{H}_3)] - \Pr[\mathcal{D}_i(\mathcal{H}_4)]| \leq 2^{2(i-1)}\cdot O(\delta_{i-1} + 2^{-\Omega(\tilde{\lambda}^\epsilon)}) \ . \tag{3.14}$$

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 3.13 and 3.14, Lemma 3.5.2, and inequality 3.8, we obtain inequality 3.7 in Lemma 3.5.1.

This completes the security analysis for $\mathsf{i}\mathcal{O}(1^\lambda, C) = \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ in Section 3.5.1.

3.5.3 Efficiency Analysis

In this section, we prove that $\mathsf{i}\mathcal{O}(1^\lambda, C) = \mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ 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 $\mathsf{i}\mathcal{O}$. When we invoke $\mathsf{ri}\mathcal{O}(1^{\tilde{\lambda}}, n, C)$ for some circuit C with n-bit input, $\mathsf{i}\mathcal{O}$ generates total $n-1$ master secret keys $\mathsf{MSK}_2, \cdots, \mathsf{MSK}_n$. In other words, $\mathsf{i}\mathcal{O}$ 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 $\mathsf{i}\mathcal{O}$. 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 $\mathsf{Enc}(\mathsf{MSK}_i, \cdot)$. Moreover, let PEnc_i denote the punctured encryption circuit PEnc with respect to MSK_i^*, that is $\mathsf{PEnc}(\mathsf{MSK}_i^*, \cdot)$, 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 $\mathsf{pad}_{\mathsf{E}}^{i-1}$ of encryption circuits for every $i \in \{3, \ldots, n\}$. Then, using the bound of $\mathsf{pad}_{\mathsf{E}}^{i-1}$, we complete the efficiency analysis of $\mathsf{i}\mathcal{O}$.

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

$$\mathsf{pad}_{\mathsf{E}}^{i-1} := \max(|\mathsf{E}_{i-1,\alpha}|, \left|\mathsf{PE}_{i-1}^j\right|, \left|\mathsf{E}_{i-1}^j\right|, \left|\mathsf{HE}_{i-1}^{j,k}\right|) \ .$$

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

$$\mathsf{pad}_{\mathsf{E}}^{i-1} \leq |\mathsf{Enc}_i| \cdot \mathsf{poly}_{\mathsf{pad}}(\tilde{\lambda}, n) \ , \tag{3.15}$$

where $\mathsf{poly}_{\mathsf{pad}}$ is a fixed polynomial.

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

First, for every $i \in \{2, \ldots, n\}$, 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 $i \in \{2, \ldots, n\}$ and circuit C_i with i-bit input, let $\mathsf{Ev}_i[C_i]$ denote the evaluation circuit Ev_i defined in Figure 3.13 into which C_i is hardwired after encrypted by SKE. In addition, for every $i \in \{2, \ldots, n\}$, let E_{i-1} denote the encryption circuit $\mathsf{E}_{i-1,0}[\mathsf{MSK}_i, K_i, S_i]$ defined in Figure 4.4.

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

From inequality 3.15, for every $i \in \{3, \ldots, n\}$, we have

$$|\mathsf{E}_{i-1}| \leq |\mathsf{Enc}_i| \cdot \mathsf{poly}_{\mathsf{pad}}(\tilde{\lambda}, n) \ . \tag{3.16}$$

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

$$|\mathsf{Ev}_i[C_i]| \leq |C_i| \cdot \log |C_i| \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) \leq |C_i|^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) \ , \tag{3.17}$$

where $\mathsf{poly}_{\mathsf{Ev}}$ is a fixed polynomial and $\gamma_1 < 1$ is an arbitrary constant. By combining inequalities 3.16 and 3.17, for every $i \in \{3, \ldots, n\}$, we obtain

$$\begin{aligned} |\mathsf{Ev}_{i-1}[\mathsf{E}_{i-1}]| &\leq |\mathsf{E}_{i-1}|^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) \\ &\leq \left(|\mathsf{Enc}_i| \cdot \mathsf{poly}_{\mathsf{pad}}(\tilde{\lambda}, n)\right)^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) \\ &\leq |\mathsf{Enc}_i|^{1+\gamma_1} \cdot \mathsf{poly}_1(\tilde{\lambda}, n) \ , \end{aligned} \tag{3.18}$$

where poly_1 is some fixed polynomial.

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

$$s_{i-1} \leq |\mathsf{Enc}_i|^{1+\gamma_1} \cdot \mathsf{poly}_1(\tilde{\lambda}, n) \ . \tag{3.19}$$

Moreover, from inequality 3.17, we have

$$s_n \leq |C|^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) = s^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n) \ . \tag{3.20}$$

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

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

$$\begin{aligned} |\mathsf{Enc}_i| &\leq s_i^{\gamma} \cdot \mathsf{poly}(\tilde{\lambda}, n + \tilde{\lambda} + 1) \\ &\leq s_i^{\gamma} \cdot \mathsf{poly}_{\mathsf{E}}(\tilde{\lambda}, n) \ , \end{aligned} \tag{3.21}$$

where $\gamma < 1$ is some constant and $\mathsf{poly}_{\mathsf{E}}$ denotes some fixed polynomial. By substituting inequality 3.19 into 3.21, for every $i \in \{3, \ldots, n\}$, we obtain

$$\begin{aligned} |\mathsf{Enc}_{i-1}| &\leq \left(|\mathsf{Enc}_i|^{1+\gamma_1} \cdot \mathsf{poly}_1(\tilde{\lambda}, n)\right)^{\gamma} \cdot \mathsf{poly}_{\mathsf{E}}(\tilde{\lambda}, n) \\ &\leq |\mathsf{Enc}_i|^{\gamma_2} \cdot \mathsf{poly}_2(\tilde{\lambda}, n) \ , \end{aligned} \tag{3.22}$$

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 3.20, we also have

$$\begin{aligned} |\mathsf{Enc}_n| &\leq \left(s^{1+\gamma_1} \cdot \mathsf{poly}_{\mathsf{Ev}}(\tilde{\lambda}, n)\right)^{\gamma} \cdot \mathsf{poly}_{\mathsf{E}}(\tilde{\lambda}, n) \\ &\leq s^{\gamma_2} \cdot \mathsf{poly}_3(\tilde{\lambda}, n) \ , \end{aligned} \tag{3.23}$$

where poly_3 is some fixed polynomial.

Then, from inequalities 3.22 and 3.23, for every $i \in \{2, \ldots, n\}$, it holds that

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

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 $i \in \{2, \ldots, n\}$.

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

We complete the analysis of security and efficiency. Thus, we complete the proof of Theorem 3.5.1 and prove that we can construct IO for all circuits solely from SKFE for all circuits.

Chapter 4

Collusion-Resistant SKFE from Succinct SKFE

In this chapter, we show how to construct collusion-resistant SKFE based on succinct one. We first give a high-level overview of our techniques to accomplish the task.

4.1 Overview

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, BKS16, GS16, LM16], but we cannot directly use such techniques as we see below.

4.1.1 Re-encryption Techniques in The Public-Key Setting

It was shown that single-key weakly succinct PKFE implies collusion-resistant PKFE by Garg and Srinivasan [GS16] and Li and Micciancio [LM16]. Their techniques are different, but the core idea seems to be the same. Both techniques use functional decryption keys for a re-encryption function that outputs a ciphertext under a different encryption key. Below, we give more details of the technique by Li and Micciancio since it is our starting point. In the following, we call a functional encryption scheme that can securely issue q functional keys q-key scheme.

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

two instances of it via the PRODUCT construction, we can construct a λ^2-key PKFE scheme, where λ is the security parameter.

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

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

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

The PRODUCT construction involves *the chaining of re-encryption by functional decryption keys* used in many previous works [AJ15, BV15, BKS16, GS16]. This technique causes several difficulties when we adopt the PRODUCT construction in the SKFE setting. This is also the reason the building block single-key PKFE scheme must satisfy (weak) succinctness property.

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

(2×2)-key scheme from 2-key scheme.

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

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

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

$, \mathcal{E}[\mathsf{MPK}_i], i)$ and $\mathsf{sk}_f^{(i,j)} \leftarrow \mathsf{KG}(\mathsf{MSK}_i, f, j)$, where $\mathcal{E}$ is a re-encryption circuit described below. A functional key is $(\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]}, \mathsf{sk}_f^{(i,j)})$.

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

Decryption: First, obtain $\mathsf{ct}_{\mathsf{post}}$ by decrypting $\mathsf{ct}_{\mathsf{pre}}$ using $\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]}$, that is, $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Dec}(\mathsf{sk}^i_{\mathcal{E}[\mathsf{MPK}_i]}, \mathsf{ct}_{\mathsf{pre}})$. Then, compute $f(m) \leftarrow \mathsf{Dec}(\mathsf{sk}_f^{(i,j)}, \mathsf{ct}_{\mathsf{post}})$.

Re-encryption circuit $\mathcal{E}[\mathsf{MPK}_i]$ used in the above description has a master public key MPK_i hardwired and runs as follows. $\mathcal{E}[\mathsf{MPK}_i]$ takes as an input a message m and outputs $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$.

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

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

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

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

4.1.2 Techniques in Multi-Input SKFE

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

Brakerski, Komargodski, and Segev [BKS16] 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 [BS15].

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

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

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

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

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

4.1.3 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 [ABSV15] showed that a hybrid encryption construction is useful in designing adaptively secure functional encryption from selectively secure one without any additional assumption. In fact, Brakerski et al. [BKS16] also used a hybrid encryption construction to achieve an input aggregation mechanism for multi-input SKFE.

In this study, we propose a new hybrid encryption construction for functional encryp-

tion 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 $\mathsf{1CT}$, which is constructed only from one-way functions. The notable features of $\mathsf{1CT}$ are as follows.

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

$\mathsf{1CT}$ 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.[2] 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 $\mathsf{1CT}$ 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 by Ananth et al. [ABSV15].[3] We use $\mathsf{1CT}$ as data encapsulation mechanism and a q-key weakly succinct SKFE scheme SKFE as key encapsulation mechanism. In our hybrid construction, the encryption algorithm of SKFE encrypts only short values (concretely, a one-time master secret key of $\mathsf{1CT}$), which are independent of the length of a message to be encrypted. A one-time encryption key (that is short and fixed length) of $\mathsf{1CT}$ is encrypted by SKFE.

That is, by this hybrid construction, a real message part is shifted onto $\mathsf{1CT}$, whose ciphertext has the full succinctness property. In other words, we can separate the blow-up due to recursion from nested-ciphertext-embedding part. Therefore, we call our new hybrid construction technique "size-shifting". See Section 4.5.1 for more detailed intuition.

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

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

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

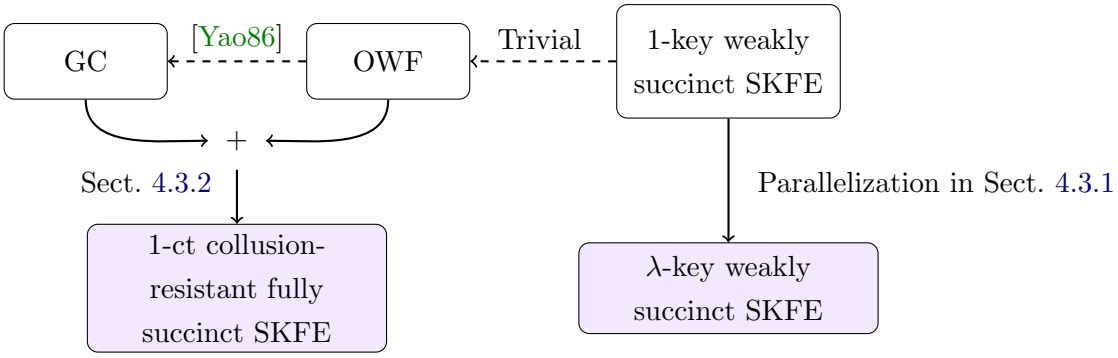

Figure 4.1: An illustration of our building blocks. Purple boxes denote our core schemes used in our iterated construction in Figure 4.2.

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

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

The size-shifting procedure is "sandwiched" by each our new PRODUCT construction. As a result, we can reduce the blow-up of the encryption time after k iterations to $k \cdot \lambda^{O(1)}$ if the underlying single-key scheme is weakly-succinct while the naive k iterated applications of our new PRODUCT construction incurs $\lambda^{2^{O(k)}}$ size blow-up. Therefore, we can iterate our new PRODUCT construction $\omega(1)$ times via the size-shifting and construct a collusion-resistant SKFE scheme based only on a single-key (weakly) succinct SKFE scheme.[4]

Figure 4.1 illustrates how to construct our building blocks. An illustration of our sandwiched size-shifting procedure is described in Figure 4.2.

4.2 Index Based Secret-Key Functional Encryption

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 PKFE scheme supports through the index based variant syntax. We introduce the syntax of index based variant SKFE here. We

[4]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.

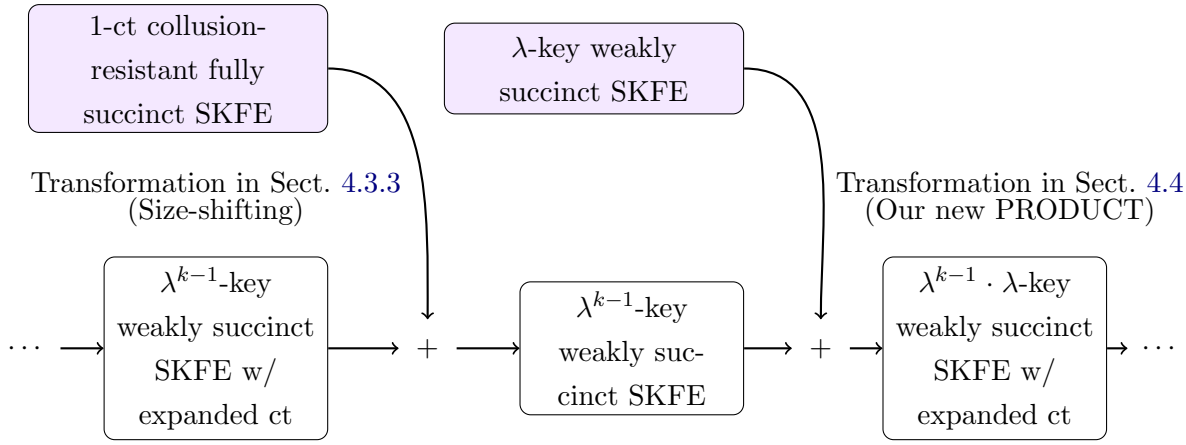

Figure 4.2: An illustration of our iteration technique, in which the size-shifting procedure is sandwiched. For k-th iteration, first, we apply the size-shifting procedure to a λ^{k-1}-key weakly-succinct SKFE scheme with expanded ciphertexts incurred by nested-ciphertext-embedding (the result of $(k-1)$-th iteration). Second, we apply our new PRODUCT construction to increase the number of issuable keys.

call it *index based secret-key functional encryption (iSKFE)*. The index based definition is required to use the strategy similar to that of Li and Micciancio.

In fact, for a single-key scheme, an iSKFE scheme is also an ordinary 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 4.2.1 for more details.

Definition 4.2.1 (Index based secret-key functional encryption) *An iSKFE scheme* iSKFE *is a four tuple* $(\mathsf{Setup}, \mathsf{iKG}, \mathsf{Enc}, \mathsf{Dec})$ *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* $\tilde{m} \in \{\bot\} \cup \mathcal{M}$*.*

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

Next, we introduce selective-message function privacy for iSKFE. We start the transformation in this chapter from selective-message function private single-key SKFE. As noted before, a message private SKFE scheme can be transformed into function private one without any additional assumption [BS15], and thus we can start the transformation from a message private SKFE scheme.

Definition 4.2.2 (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.*

1. *The challenger generates a master secret key* $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ *and chooses a challenge bit* $b \xleftarrow{\mathsf{r}} \{0,1\}$. *The challenger sends security parameter* 1^λ *to* $\mathcal{A}$.

2. $\mathcal{A}$ *sends* $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell \in [p]}$ *to the challenger, where* p *is an a-priori unbounded polynomial of* λ. *The challenger generates ciphertexts* $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b^\ell)(\ell \in [p])$ *and sends them to* $\mathcal{A}$.

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

3. *For a key query* $(i, f_0, f_1) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$ *from* $\mathcal{A}$, *the challenger generates* $sk_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f_b, i)$, *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.*

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

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

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

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* $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{A}}(\lambda) < \epsilon(\lambda)^{\Omega(1)}$.

Below, we say that an iSKFE scheme is $(\mathsf{poly}, \epsilon)$-selective-message function private if the size of its index space q is super-polynomial. We also call such a scheme collusion-resistant iSKFE. Note that collusion-resistance for functional encryption generally does not require function privacy.

Remark 4.2.1 (Transforming iSKFE into SKFE) *The goal of this chapter is to construct collusion-resistant SKFE based on single-key SKFE. To achieve this, we construct* $(\mathsf{poly}, \epsilon)$*-selective-message function private SKFE based only on* $(1, \epsilon')$*-selective-message function private SKFE for some negligible functions* ϵ *and* ϵ'*. In order to accomplish 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* $(\frac{1}{S})^{\Omega(1)}$*. Therefore, the resulting SKFE is* $(\mathsf{poly}, \frac{1}{S} + \delta)$*-selective-message function private. In Section 4.5.4, we formally show that this transformation works.*

Next, we define the succinctness for iSKFE.

Definition 4.2.3 (Succinctness) *Let* s *and* n *be the maximum size and input length of functions contained in* $\mathcal{F}$*, respectively.*

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

Weakly succinct: *We say that iSKFE is weakly succinct if the size of the encryption circuit is bounded by* $s^{\gamma} \cdot \mathsf{poly}(\lambda, n)$*, where* $\gamma < 1$ *is a fixed constant.*

Collusion-succinct: *We say that iSKFE is collusion succinct if the size of the encryption circuit is bounded by* $\mathsf{poly}(n, \lambda, s, \log q)$*, where* q *is the size of the index space.*

4.3 Basic Tools for Transformation

In this section, we introduce some basic constructions we use in the chapter.

4.3.1 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. The construction is as follows.

Let $\mathsf{1Key} = (\mathsf{1Key.Setup}, \mathsf{1Key.KG}, \mathsf{1Key.Enc}, \mathsf{1Key.Dec})$ be a single-key SKFE scheme. We construct an iSKFE scheme $\mathsf{Parallel}_q = (\mathsf{Para}_q.\mathsf{Setup}, \mathsf{Para}_q.\mathsf{iKG}, \mathsf{Para}_q.\mathsf{Enc}, \mathsf{Para}_q.\mathsf{Dec})$

as follows. Note again that q is a fixed polynomial of λ. Let the function space of $\mathsf{1Key}$ be $\mathcal{F}$. The function space of $\mathsf{Parallel}_q$ is also $\mathcal{F}$.

Construction. The scheme consists of the following algorithms.

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

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

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

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

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

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

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

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

We have the following theorem.

Theorem 4.3.1 *Let* $\mathsf{1Key}$ *be* $(1, \delta)$*-selective-message function private SKFE. Then,* $\mathsf{Parallel}_q$ *is* (q, δ)*-selective-message function private iSKFE.*

Proof of Theorem 4.3.1. The correctness of this construction directly follows from that of $\mathsf{1Key}$.

Efficiency. Before proving the security, we estimate the efficiency of $\mathsf{1Key}$.

Let $\mathsf{1Key}.t_{\mathsf{Enc}}$ and $\mathsf{1Key}.t_{\mathsf{KG}}$ be bounds of the running time of $\mathsf{1Key.Enc}$ and $\mathsf{1Key.KG}$. In addition, let $\mathsf{Para}_q.t_{\mathsf{Enc}}$ and $\mathsf{Para}_q.t_{\mathsf{iKG}}$ be bounds of the running time of $\mathsf{Para}_q.\mathsf{Enc}$ and $\mathsf{Para}_q.\mathsf{iKG}$. Then, we have

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

Especially, if $\mathsf{1Key}$ is (weakly) succinct and we set $q := \lambda$, then $\mathsf{Parallel}_q$ is also (weakly) succinct and we have

$$\mathsf{Para}_q.t_{\mathsf{Enc}}(\lambda, n, s) = \lambda \cdot \mathsf{1Key}.t_{\mathsf{Enc}}(\lambda, n, s) = s^{\gamma} \cdot \mathsf{poly}_{\mathsf{Para}}(\lambda, n), \tag{4.1}$$

where $\gamma < 1$ is a constant and $\mathsf{poly}_{\mathsf{Para}}$ is a fixed polynomial. Note that the encryption algorithm of $\mathsf{Parallel}_\lambda$ runs that of $\mathsf{1Key}$ λ times. Therefore, $\mathsf{Para}.t_{\mathsf{Enc}}$ is λ times bigger than the encryption time of $\mathsf{1Key}$, but the factor λ is absorbed in $\mathsf{poly}_{\mathsf{Para}}(\lambda, n)$, and thus we can bound $\mathsf{Para}.t^{(k)}_{\mathsf{Enc}}$ by inequality 4.1. At the concrete instantiation in Section 4.5, we set $q := \lambda$ and use this bound.

Security. We assume that the advantage of any adversary attacking $\mathsf{1Key}$ is bounded by ϵ_{1Key}. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{Parallel}_q$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Parallel}_q,\mathcal{A}}(\lambda) \leq q \cdot \epsilon_{\mathsf{1Key}}. \tag{4.2}$$

This means that if $\mathsf{1Key}$ is δ-secure, then so is $\mathsf{Parallel}_q$. Below, we prove the above inequality 4.2.

Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{1Key}$.

1. On input security parameter 1^λ, $\mathcal{B}$ sends it to $\mathcal{A}$. $\mathcal{B}$ chooses $k^* \xleftarrow{\mathsf{r}} [q]$ and for every $k \in [q] \setminus \{k^*\}$, generates $\mathsf{MSK}_k \leftarrow \mathsf{1Key.Setup}(1^\lambda)$.

2. When, $\mathcal{A}$ sends $\left\{(m^\ell_0, m^\ell_1)\right\}_{\ell\in[q]}$, $\mathcal{B}$ computes as follows.

   - For every $k < k^*$ and $\ell \in [p]$, $\mathcal{B}$ computes $\mathsf{CT}^{(\ell)}_k \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_k, m^\ell_0)$.
   - $\mathcal{B}$ sends $\left\{(m^\ell_0, m^\ell_1)\right\}_{\ell\in[p]}$ to the challenger, and obtains the answer $\left\{\mathsf{CT}^{(\ell)}_{k^*}\right\}_{\ell\in[p]}$.
   - For every $k > k^*$ and $\ell \in [p]$, $\mathcal{B}$ computes $\mathsf{CT}^{(\ell)}_k \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_k, m^\ell_1)$.

   $\mathcal{B}$ sets $\mathsf{CT}^{(\ell)} \leftarrow \left\{\mathsf{CT}^{(\ell)}_k\right\}_{k\in[q]}$ for every $\ell \in [p]$, and returns $\left\{\mathsf{CT}^{(\ell)}\right\}_{\ell\in[p]}$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a key query $(i, f_0, f_1) \in [q] \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ responds as follows.

   - If $i < k^*$, $\mathcal{B}$ computes $\mathsf{1Key.sk}^i_f \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f_0)$, and returns $\mathsf{sk}^i_f \leftarrow (i, \mathsf{1Key.sk}^i_f)$ to $\mathcal{A}$.
   - If $i = k^*$, $\mathcal{B}$ first queries (f_0, f_1) to the challenger, and obtains the answer $\mathsf{1Key.sk}_f$. Then, $\mathcal{B}$ returns $\mathsf{sk}^{k^*}_f \leftarrow (k^*, \mathsf{1Key.sk}_f)$ to $\mathcal{A}$.
   - If $i > k^*$, $\mathcal{B}$ computes $\mathsf{1Key.sk}^i_f \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f_1)$, and returns $\mathsf{sk}^i_f \leftarrow (i, \mathsf{1Key.sk}^i_f)$ to $\mathcal{A}$.

4. When $\mathcal{A}$ terminates with output b', $\mathcal{B}$ outputs $\beta' = b'$.

Let β be the challenge bit between the challenger and $\mathcal{B}$. Since $\mathcal{A}$ is a valid adversary, for every $\ell \in [p]$ and key query (i, f_0, f_1), $f_0(m_0^\ell) = f_1(m_1^\ell)$ holds. In addition, since $\mathcal{A}$ makes 1 key query under the index k^* at most, $\mathcal{B}$ makes 1 key query at most. Therefore, $\mathcal{B}$ is a valid adversary for $\mathsf{1Key}$, and we have

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

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 $\Pr[b' = 1|k^* = Q \wedge \beta = 0] = \Pr[b' = 1|k^* = Q+1 \wedge \beta = 1]$ for every $Q \in [q-1]$. Therefore, we also have

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

When $k^* = 1$ and $\beta = 1$, $\mathcal{B}$ perfectly simulates the selective-message function privacy game when the challenge bit is 1 for $\mathcal{A}$. On the other hand, when $k^* = q$ and $\beta = 0$, $\mathcal{B}$ perfectly simulates the selective-message function privacy game when of the challenge bit is 0 for $\mathcal{A}$. Therefore, we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Parallel}_q,\mathcal{A}}(\lambda) = |\Pr[b' = 1|k^* = 1 \wedge \beta = 1] - \Pr[b' = 1|k^* = q \wedge \beta = 0]|$, and thus we obtain $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Parallel}_q,\mathcal{A}}(\lambda) = q \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1Key},\mathcal{B}}(\lambda) \leq q \cdot \epsilon_{\mathsf{1Key}}$. Therefore, inequality 4.2 holds. □ (**Theorem 4.3.1**)

4.3.2 Single-Ciphertext Collusion-Resistant Fully Succinct SKFE

Next, we show how to construct a succinct SKFE scheme $\mathsf{1CT}$ 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 $\mathsf{1CT}$ 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 that proposed by Sahai and Seyalioglu [SS10].

Let $n = |m|$. Let $\mathsf{GC} = (\mathsf{Garble}, \mathsf{Eval})$ be a garbling scheme, $\mathsf{SKE} = (\mathsf{E}, \mathsf{D})$ an SKE scheme, and $\{\mathsf{F}_S : \{0, \cdots, n\} \times \{0,1\} \to \{0,1\}^n \mid S \in \{0,1\}^\lambda\}$ a PRF. Using GC ,SKE, and F, we construct an SKFE scheme $\mathsf{1CT} = (\mathsf{1CT.Setup}, \mathsf{1CT.KG}, \mathsf{1CT.Enc}, \mathsf{1CT.Dec})$ as follows.

Construction. The scheme consists of the following algorithms.

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

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

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

- Parse $S \leftarrow \mathsf{MSK}$.
- Compute $K_{j,\alpha} \leftarrow \mathsf{F}_S(j\|\alpha)$ for every $j \in [n]$ and $\alpha \in \{0,1\}$, and $R \leftarrow \mathsf{F}_S(0\|0)$[5].
- Compute $(\widetilde{C}, \{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Garble}(1^\lambda, C_f)$, where C_f is a circuit computing f.
- For every $j \in [n]$, compute $c_{j,0} \leftarrow \mathsf{E}(K_{j,0}, L_{j,R[j]})$ and $c_{j,1} \leftarrow \mathsf{E}(K_{j,1}, L_{j,1-R[j]})$.
- Return $\mathsf{sk}_f \leftarrow (\widetilde{C}, \{c_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}})$.

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

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

$\mathsf{1CT.Dec}(\mathsf{sk}_f, \mathsf{CT})$:

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

We have the following theorem.

Theorem 4.3.2 *Let* GC *be* δ*-secure garbling scheme,* SKE *be* δ*-secure SKE, and* F *be* δ*-secure PRF. Then,* $\mathsf{1CT}$ *is single-ciphertext* (poly, δ)*-selective-message function private SKFE.*

Proof of Theorem 4.3.2. We first prove the correctness of $\mathsf{1CT}$.

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 $\mathsf{1CT}$ directly follows from those of building blocks.

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

Efficiency. Before proving the security, we estimate the efficiency of $\mathsf{1CT}$.

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

$$\begin{aligned} |\mathsf{MSK}| = \lambda,\ \mathsf{1CT}.\ell^{\mathsf{Enc}}(\lambda, n) &= (\lambda+1)n, \\ \mathsf{1CT}.t_{\mathsf{Enc}}(\lambda, n) &= n \cdot \mathsf{poly}_{\mathsf{1CT}}(\lambda, \log n), \\ \mathsf{1CT}.t_{\mathsf{KG}}(\lambda, n, s) &= t_{\mathsf{F}} + t_{\mathsf{Garble}}(\lambda, |f|) + 2 \cdot n \cdot t_{\mathsf{E}} \\ &\leq \mathsf{poly}(\lambda, \log n) + |f| \cdot \mathsf{poly}(\lambda) + n \cdot \mathsf{poly}(\lambda) \\ &\leq (|f| + n) \cdot \mathsf{poly}(\lambda), \end{aligned} \tag{4.3}$$

where $\mathsf{poly}_{\mathsf{1CT}}$ is a fixed polynomial.

The master secret-key length is $|\mathsf{MSK}| = \lambda$ 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 assume that the advantage of any adversary attacking GC, SKE, and F is bounded by ϵ_{GC}, ϵ_{SKE}, and ϵ_{F}, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{1CT}$. Moreover, we assume that $\mathcal{A}$ makes q key queries at most, where q is a polynomial of λ. Then, it holds that

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{A}}(\lambda) \leq 2(q \cdot \epsilon_{\mathsf{GC}} + \epsilon_{\mathsf{SKE}} + \epsilon_{\mathsf{F}}). \tag{4.4}$$

This means that if all of GC, SKE, and F are δ-secure, then so is $\mathsf{1CT}$. Below, we prove this via a sequence of games. First, consider the following sequence of games. For $h = 0, \cdots, 4$, let $\mathtt{SUC}_i$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit, that is, $b = b'$ occurs in Game i.

Game 0 This is the selective-message function privacy game regarding $\mathsf{1CT}$.

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

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

We have $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \epsilon_{\mathsf{F}}$ by the security of F.

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

We have $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \epsilon_{\mathsf{SKE}}$ by the security of SKE.

Game 3 Same as Game 2 except that when $\mathcal{A}$ makes a key query (f_0, f_1), the challenger computes $(\widetilde{C}, \{L_j\}_{j\in[n]}) \leftarrow \mathsf{Sim}(1^\lambda, s, y)$, 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 $c_{j,x[j]} \leftarrow \mathsf{E}(K_{j,x[j]}, L_j)$ for every $j \in [n]$.

We have $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| \leq q \cdot \epsilon_{\mathsf{GC}}$ by the security of GC.

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

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

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

$$\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{A}}(\lambda) = |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \leq \sum_{h=0}^{3} |\Pr[\mathtt{SUC}_h] - \Pr[\mathtt{SUC}_{h+1}]|. \tag{4.5}$$

From inequality 4.5 and the above arguments, we obtain inequality 4.4. This completes the proof. □ (**Theorem 4.3.2**)

4.3.3 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 4.3.2, 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 $\mathsf{iSKFE} = (\mathsf{Setup}, \mathsf{iKG}, \mathsf{Enc}, \mathsf{Dec})$ be an iSKFE scheme whose index space is $\mathcal{I}$. Let $\mathsf{1CT} = (\mathsf{1CT.Setup}, \mathsf{1CT.KG}, \mathsf{1CT.Enc}, \mathsf{1CT.Dec})$ be an SKFE scheme. Let $\{\mathsf{F}_S : \mathcal{I} \to \mathcal{R} | S \in \{0,1\}^\lambda\}$ be a PRF, where $\mathcal{R}$ is the randomness space of $\mathsf{1CT.KG}$. We construct an iSKFE scheme $\mathsf{HYBRD} = (\mathsf{Hyb.Setup}, \mathsf{Hyb.iKG}, \mathsf{Hyb.Enc}, \mathsf{Hyb.Dec})$ as follows. Then, the index space of HYBRD is the same as 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 4.3.

Construction. The scheme consists of the following algorithms.

$\mathsf{Hyb.Setup}(1^\lambda)$:

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

$\mathsf{Hyb.iKG}(\mathsf{MSK}, f, i)$:

- Compute $\mathsf{sk}_G \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f, \bot, \bot, i], i)$. The circuit G is defined in Figure 4.3.
- Return $\mathsf{sk}_f \leftarrow \mathsf{sk}_G$.

$\mathsf{Hyb.Enc}(\mathsf{MSK}, m)$:

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

$\mathsf{Hyb.Dec}(\mathsf{sk}_f, \mathsf{Hyb.CT})$:

- Parse $\mathsf{sk}_G \leftarrow \mathsf{sk}_f$ and $(\mathsf{CT}, \mathsf{1CT.CT}) \leftarrow \mathsf{Hyb.CT}$.
- Compute $\mathsf{1CT.sk}_f \leftarrow \mathsf{Dec}(\mathsf{sk}_G, \mathsf{CT})$.

Key generation circuit $G[f_0, f_1, u, i](\mathsf{1CT.MSK}, S, \alpha)$

Hardwired: functions f_0 and f_1, functional key u, and index i.

Input: master secret key $\mathsf{1CT.MSK}$, PRF key S, and bit $\alpha \in \{0,1\}$.

1. If $\mathsf{1CT.MSK} = \bot$, return u.
2. Else compute $r_i \leftarrow \mathsf{F}_S(i)$ and return $\mathsf{1CT.sk}_f \leftarrow \mathsf{1CT.KG}(\mathsf{1CT.MSK}, f_\alpha; r_i)$.

Figure 4.3: Construction of a key generation circuit G.

- Return $y \leftarrow \mathsf{1CT.Dec}(\mathsf{1CT.sk}_f, \mathsf{1CT.CT})$.

We have the following theorem.

Theorem 4.3.3 *Let* iSKFE *be* (q, δ)*-selective-message function private iSKFE, where* q *is a fixed function of* λ*. Let* $\mathsf{1CT}$ *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 4.5, we set q as a super-polynomial of λ.

Proof of Theorem 4.3.3. 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 4.5.

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

$$\begin{aligned} |G| &= 2\,|f| + |\mathsf{1CT.sk}_f| + |i| + |\mathsf{1CT.KG}| + |\mathsf{F}| \\ &\leq 2\,|f| + (|f| + |m|) \cdot \mathsf{poly}(\lambda) + \log q + (|f| + |m|) \cdot \mathsf{poly}(\lambda) + \mathsf{poly}(\lambda, \log q) \\ &\leq (|f| + |m|) \cdot \mathsf{poly}_G(\lambda, \log q) \ , \end{aligned} \tag{4.6}$$

where poly_G is some fixed polynomial. The second inequality holds due to the efficiency of $\mathsf{1CT}$ in Section 4.3.2. Note that $\mathsf{1CT.sk}_f$ is output by $\mathsf{1CT.KG}$ thus $|\mathsf{1CT.sk}_f|$ is bounded by $|\mathsf{1CT.KG}|$.

Security. We assume that the advantage of any adversary attacking iSKFE, $\mathsf{1CT}$, and F is bounded by ϵ, ϵ_{1CT}, and ϵ_{F}, 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

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) \leq 2\left((2p+1)\cdot\epsilon + p\cdot\epsilon_{\mathsf{1CT}} + p\cdot\epsilon_{\mathsf{F}}\right). \tag{4.7}$$

This means that if all of iSKFE, $\mathsf{1CT}$, and F are δ-secure, then so is HYBRD. Below, we prove this via 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 $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[p]}$ to the challenger. Next, the challenger generates $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$ and chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. Then, for every $\ell \in [p]$, the challenger generates $\mathsf{1CT.MSK}^{(\ell)} \leftarrow \mathsf{1CT.Setup}(1^\lambda)$, $S^{(\ell)} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\mathsf{1CT.MSK}^{(\ell)}, S^{(\ell)}, 0))$. Finally, the challenger generates $\mathsf{1CT.CT}^{(\ell)} \leftarrow \mathsf{1CT.Enc}(\mathsf{1CT.MSK}^{(\ell)}, m_b^\ell)$ for every $\ell \in [p]$, and returns $\left\{(\mathsf{CT}^{(\ell)}, \mathsf{1CT.CT}^{(\ell)})\right\}_{\ell\in[p]}$ 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 $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, \bot, \bot, i], i)$ to $\mathcal{A}$.

Final phase $\mathcal{A}$ output b'.

For every $\ell^* \in [p]$, we define the following games. We define Game $(5,0)$ as the same game as Game 0.

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

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 $r_i^{(\ell^*)} \leftarrow \mathsf{F}_{S^{(\ell^*)}}(i)$ and $u_i \leftarrow \mathsf{1CT.KG}(\mathsf{1CT.MSK}^{(\ell^*)}, f_b^i; r_i^{(\ell^*)})$, and returns $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, u_i, i], i)$.

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

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

Game $(5, \ell^*)$ Same as Game $(4, \ell^*)$ except the following. The challenger generates $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (1\mathsf{CT}.\mathsf{MSK}^{(\ell^*)}, S^{(\ell^*)}, 1))$. 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 $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, \bot, i], i)$.

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 $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[\bot, f_1^i, \bot, i], i)$. In this game, the challenger generates $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (1\mathsf{CT}.\mathsf{MSK}^{(\ell)}, S^{(\ell)}, 1))$ for every $\ell \in [p]$.

Let $\mathtt{SUC}_0$ and $\mathtt{SUC}_6$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game 0 and 6, respectively. Similarly, for every $h \in \{1, \cdots, 5\}$ and $\ell^* \in [p]$, let $\mathtt{SUC}_{(h,\ell^*)}$ 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 $|\Pr[\mathtt{SUC}_6] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\
&\leq \sum_{\ell^* \in [p]} |\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \\
&\quad + \sum_{\ell^* \in [p]} \sum_{h=1}^{4} |\Pr[\mathtt{SUC}_{(h,\ell^*)}] - \Pr[\mathtt{SUC}_{(h+1,\ell^*)}]| \\
&\quad + |\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]| \qquad (4.8)
\end{aligned}$$

Below, we estimate each term on the right side of inequality 4.8.

Lemma 4.3.1 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \leq \epsilon$.

Proof of Lemma 4.3.1. Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks iSKFE.

1. On input security parameter 1^λ, $\mathcal{B}$ chooses $b \xleftarrow{r} \{0, 1\}$ and sends 1^λ to $\mathcal{A}$.
2. When $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell \in [q]}$, $\mathcal{B}$ sets $\left\{(M_0^\ell, M_1^\ell)\right\}_{\ell \in [p]}$ as follows.
   - For every $\ell < \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (1\mathsf{CT}.\mathsf{MSK}^{(\ell)}, S^{(\ell)}, 1)$.
   - $\mathcal{B}$ sets $M_0^{\ell^*} = (1\mathsf{CT}.\mathsf{MSK}^{(\ell^*)}, S^{(\ell^*)}, 0)$ and $M_1^{\ell^*} = (\bot, \bot, 0)$.
   - For every $\ell > \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (1\mathsf{CT}.\mathsf{MSK}^{(\ell)}, S^{(\ell)}, 0)$.

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

3. 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 $u_i \leftarrow \mathsf{1CT.KG}(\mathsf{MSK}^{(\ell^*)}, f_b^i; r_i^{(\ell^*)})$, where $r_i^{(\ell^*)} \leftarrow \mathsf{F}_{S^{(\ell^*)}}(i)$. Then, $\mathcal{B}$ queries $(i, G[f_b^i, f_1^i, \bot, i], G[f_b^i, f_1^i, u_i, i])$ to the challenger and returns the answer to $\mathcal{A}$.

4. 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 $\mathcal{A}$, $G[f_b^i, f_1^i, \bot, i](M_0^\ell) = G[f_b^i, f_1^i, u_i, i](M_1^\ell)$ holds if $f_0^i(m_0^\ell) = f_1^i(m_0^\ell)$ holds. Moreover, we have

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

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 $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. $\mathcal{B}$ perfectly simulates Game $(5, \ell^* - 1)$ if $\beta = 0$. On the other hand, $\mathcal{B}$ perfectly simulates Game $(1, \ell^*)$ if $\beta = 1$. Moreover, $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds in guessing the value of b. Therefore, we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]|$ thus $|\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \leq \epsilon$ holds. □ (**Lemma 4.3.1**)

Lemma 4.3.2 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(1,\ell^*)}] - \Pr[\mathtt{SUC}_{(2,\ell^*)}]| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus omitted.

Lemma 4.3.3 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{1CT}}$.

Proof of Lemma 4.3.3. Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{1CT}$.

1. On input security parameter 1^λ, $\mathcal{B}$ chooses $b \xleftarrow{r} \{0, 1\}$ and sends 1^λ to $\mathcal{A}$.
2. When $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[q]}$, $\mathcal{B}$ first computes $\left\{\mathsf{CT}^{(\ell)}\right\}_{\ell\in[p]}$ and $\left\{\mathsf{1CT.CT}^{(\ell)}\right\}_{\ell\in[p]}$ as follows.
   - For every $\ell < \ell^*$, $\mathcal{B}$ computes $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\mathsf{1CT.MSK}^{(\ell)}, S^{(\ell)}, 1))$ and $\mathsf{1CT.CT}^{(\ell)} \leftarrow \mathsf{1CT.Enc}(\mathsf{1CT.MSK}^{(\ell)}, m_1^\ell)$.

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

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

3. 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 $\mathsf{1CT.sk}_f^i$. Then, $\mathcal{B}$ generates $\mathsf{sk}_G^i \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, \mathsf{1CT.sk}_f^i], i)$ and returns it to $\mathcal{A}$.

4. 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 $\mathsf{1CT}$, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. 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 $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]|$ thus $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{1CT}}$ holds. □ (**Lemma 4.3.3**)

Lemma 4.3.4 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(3,\ell^*)}] - \Pr[\mathtt{SUC}_{(4,\ell^*)}]| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus omitted.

Lemma 4.3.5 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(4,\ell^*)}] - \Pr[\mathtt{SUC}_{(5,\ell^*)}]| \leq \epsilon$.

The proof is almost the same as that of Lemma 4.3.1 thus is omitted.

Lemma 4.3.6 $|\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]| \leq \epsilon$.

Proof of Lemma 4.3.6. The only difference between Game $(5, p)$ and 6 is how $\mathsf{sk}_G^{(i)}$ is generated for every $i \in [q]$. In Game $(5, p)$, it is generated as $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, \bot, i], i)$. On the other hand, in Game 6, it is generated as $\mathsf{sk}_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[\bot, f_1^i, \bot, i], i)$. Here, in both games, for every $\ell \in [p]$, $\mathsf{CT}^{(\ell)}$ is generated as $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\mathsf{1CT.MSK}^{(\ell)}, S^{(\ell)}, 1))$. Then, for every $i \in [q]$ and $\ell \in [p]$, we have

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

This is because f_b^i is ignored in the left hand side. Therefore, we can construct an adversary attacking iSKFE whose advantage is $|\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]|$ thus $|\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]| \leq \epsilon$ holds. □ (**Lemma 4.3.6**)

From inequality 4.8 and lemmas 4.3.1 to 4.3.6, inequality 4.7 holds. This completes the proof. □ (**Theorem 4.3.3**)

4.4 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 4.1, 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. [BKS16] in the context of multi-input SKFE.

Let $\mathsf{Root} = (\mathsf{Rt.Setup}, \mathsf{Rt.iKG}, \mathsf{Rt.Enc}, \mathsf{Rt.Dec})$ and $\mathsf{Leaf} = (\mathsf{Lf.Setup}, \mathsf{Lf.iKG}, \mathsf{Lf.Enc}, \mathsf{Lf.Dec})$ be iSKFE schemes. We assume that the index spaces of Root and Leaf are $\mathcal{I}_{\mathsf{Rt}}$ and $\mathcal{I}_{\mathsf{Lf}}$, respectively. Let $\{\mathsf{F}_S : \mathcal{I}_{\mathsf{Rt}} \times [2] \to \{0,1\}^\lambda \mid S \in \{0,1\}^\lambda\}$ and $\{\mathsf{F}'_S : \{0,1\}^\lambda \to \{0,1\}^\lambda \mid S \in \{0,1\}^\lambda\}$ be PRFs. Then, using Root, Leaf, F, and F', we construct an iSKFE scheme $\mathsf{PRDCT} = (\mathsf{Prd.Setup}, \mathsf{Prd.iKG}, \mathsf{Prd.Enc}, \mathsf{Prd.Dec})$ as follows. Note that the index space of PRDCT is $\mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}}$. Moreover, if the function space of Leaf is $\mathcal{F}$, then that of PRDCT is also $\mathcal{F}$. We assume that Root supports sufficiently large function class that particularly includes the encryption circuit e described in Figure 4.4. In addition, we assume that all of randomness spaces of $\mathsf{Lf.Setup}$, $\mathsf{Lf.iKG}$, and $\mathsf{Lf.Enc}$ are $\{0,1\}^\lambda$.

Construction. The scheme consists of the following algorithms.

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

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

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

- Parse $(\mathsf{Rt.MSK}, S) \leftarrow \mathsf{MSK}$.
- Compute $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i\|0)$, $S_i \leftarrow \mathsf{F}_S(i\|1)$, and $r^i_{\mathsf{iKG}} \leftarrow \mathsf{F}_S(i\|2)$.
- Generate $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$.

Encryption circuit $e[\mathsf{Lf.MSK}_i, S_i, \alpha](m_0, m_1, t, u)$:

Hardwired: master secret key $\mathsf{Lf.MSK}_i$, PRF key S_i, and bit $\alpha \in \{0,1\}$.

Input: messages m_0 and m_1, tag t, and ciphertext u.

1. If $\mathsf{Lf.MSK}_i = \bot$, return u.
2. Else, compute $r_{\mathsf{Enc}} \leftarrow \mathsf{F}'_{S_i}(t)$.
3. Return $\mathsf{Lf.CT}_i \leftarrow \mathsf{Lf.Enc}(\mathsf{Lf.MSK}_i, m_\alpha; r_{\mathsf{Enc}})$.

Figure 4.4: Construction of an encryption circuit e.

- Compute $\mathsf{Rt.sk}_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\mathsf{Lf.MSK}_i, S_i, 0], i; r^i_{\mathsf{iKG}})$ and $\mathsf{Lf}.sk_f^{i,j} \leftarrow \mathsf{Lf.iKG}(\mathsf{Lf.MSK}_i, f, j)$.
- Return $sk_f \leftarrow (\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j})$.

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

- Parse $(\mathsf{Rt.MSK}, S) \leftarrow \mathsf{MSK}$.
- Generate $t \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$.
- Compute $\mathsf{Rt.CT} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m, \bot, t, \bot))$.
- Return $\mathsf{CT} \leftarrow \mathsf{Rt.CT}$.

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

- Parse $(\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j}) \leftarrow sk_f$ and $\mathsf{Rt.CT} \leftarrow \mathsf{CT}$.
- Compute $\mathsf{Lf.CT} \leftarrow \mathsf{Rt.Dec}(\mathsf{Rt}.sk_{e_i}, \mathsf{Rt.CT})$.
- Return $y \leftarrow \mathsf{Lf.Dec}(\mathsf{Lf}.sk_f^{i,j}, \mathsf{Lf.CT})$.

Let $|\mathcal{I}_{\mathsf{Rt}}| = q_{\mathsf{Rt}}$ and $|\mathcal{I}_{\mathsf{Lf}}| = q_{\mathsf{Lf}}$. Then, we have the following theorem.

Theorem 4.4.1 *Let* Root *be* $(q_{\mathsf{Rt}}, \delta)$*-selective-message function private iSKFE, and* Leaf $(q_{\mathsf{Lf}}, \delta)$*-selective-message function private iSKFE. Let* F *and* F′ *be* δ*-secure PRF. Then,* PRDCT *is* $(q_{\mathsf{Rt}} \cdot q_{\mathsf{Lf}}, \delta)$*-selective-message function private iSKFE.*

Proof of Theorem 4.4.1. We first prove the correctness of PRDCT.

Correctness. In the construction, we use q_{Rt} instances of Leaf and thus q_{Rt} master secret keys are generated in Prd.iKG. In addition, we let Rt.iKG release the same functional key $\mathsf{Rt}.sk_{e_i}$ for the same index $i \in \mathcal{I}_{\mathsf{Rt}}$ since Root can release only q_{Rt} functional keys. In order to ensure that only q_{Rt} master secret keys $\{\mathsf{Lf.MSK}_i\}_{i \in [q_{\mathsf{Rt}}]}$ and functional keys

$\{\mathsf{Rt}.sk_{e_i}\}_{i\in[q_{\mathsf{Rt}}]}$ are generated, we manage them as one PRF key S. Then, if we decrypt $\mathsf{Rt.CT}$ by $\mathsf{Rt}.sk_{e_i}$, it is re-encrypted to a ciphertext under the master secret key $\mathsf{Lf.MSK}_i$. Thus, the correctness of PRDCT follows from those of Root and Leaf.

Efficiency. We next analyze the efficiency of PRDCT.

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

$$\begin{aligned}|e| = |\mathsf{Lf.MSK}| + \lambda + 1 + |\mathsf{Lf.Enc}| + |\mathsf{F}'| &\leq 2\,|\mathsf{Lf.Enc}| + \mathsf{poly}(\lambda)\\ &\leq |\mathsf{Lf.Enc}| \cdot \mathsf{poly}_e(\lambda),\end{aligned} \tag{4.9}$$

where poly_e is a fixed polynomial.

Let $\mathsf{Rt}.t_{\mathsf{Enc}}$ and $\mathsf{Rt}.t_{\mathsf{iKG}}$ be bounds of the running time of $\mathsf{Rt.Enc}$ and $\mathsf{Rt.iKG}$. Let $\mathsf{Lf}.t_{\mathsf{Enc}}$ and $\mathsf{Lf}.t_{\mathsf{iKG}}$ be bounds of the running time of $\mathsf{Rt.Enc}$ and $\mathsf{Rt.iKG}$. Let $\mathsf{Prd}.t_{\mathsf{Enc}}$ and $\mathsf{Prd}.t_{\mathsf{iKG}}$ be bounds of the running time of $\mathsf{Prd.Enc}$ and $\mathsf{Prd.iKG}$. Note that the length of a ciphertext output by $\mathsf{Lf.Enc}$ is bounded by its running time $\mathsf{Lf}.t_{\mathsf{Enc}}$. Then, we have

$$\begin{aligned}\mathsf{Prd}.t_{\mathsf{Enc}}(\lambda, n, s) &= \mathsf{Rt}.t_{\mathsf{Enc}}\left(\lambda, 2n + \lambda + \mathsf{Lf}.t_{\mathsf{Enc}}(\lambda, n, s), |e|\right),\\ \mathsf{Prd}.t_{\mathsf{iKG}}(\lambda, n, s) &= \mathsf{Rt}.t_{\mathsf{iKG}}\left(\lambda, 2n + \lambda + \mathsf{Lf}.t_{\mathsf{Enc}}(\lambda, n, s), |e|\right) + \mathsf{Lf}.t_{\mathsf{iKG}}(\lambda, n, s).\end{aligned}$$

Security. We assume that the advantage of any adversary attacking Root, Leaf, F, and F' is bounded by ϵ_{Rt}, ϵ_{Lf}, ϵ_{F}, and ϵ_{F}, 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 $\{i_k\}_{k\in[q]} \in \mathcal{I}^q_{\mathsf{Rt}}$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) \leq 2\left((2q+2)\cdot\epsilon_{\mathsf{Rt}} + q\cdot\epsilon_{\mathsf{Lf}} + (2q+1)\epsilon_{\mathsf{F}}\right). \tag{4.10}$$

This means that if all of Root, Leaf, F, and F' are δ-secure, then so does PRDCT. Below, we prove this via a sequence of games.

Below, we prove equality 4.10 via a sequence of games. 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.

1. The challenger sends security parameter 1^λ to $\mathcal{A}$.
2. $\mathcal{A}$ sends $\left\{(m^\ell_0, m^\ell_1)\right\}_{\ell\in[p]}$ to the challenger. Next, the challenger generates $\mathsf{Rt.MSK} \leftarrow \mathsf{Rt.Setup}(1^\lambda)$ and $S \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. Then, the challenger generates $t^{(\ell)} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ and $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m^\ell_b, \bot, t^{(\ell)}, \bot))$ for every $\ell \in [p]$, and returns $\left\{(\mathsf{Rt.CT}^{(\ell)})\right\}_{\ell\in[p]}$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, the challenger first generates $r^i_{\mathsf{Setup}} \leftarrow \mathsf{F}_S(i\|0)$, $S_i \leftarrow \mathsf{F}_S(i\|1)$, $r^i_{\mathsf{iKG}} \leftarrow \mathsf{F}_S(i\|2)$, and $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$. Then the challenger computes $\mathsf{Rt.sk}_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\mathsf{Lf.MSK}_i, S_i, 0], i; r^i_{\mathsf{iKG}})$ and $\mathsf{Lf}.sk_f^{i,j} \leftarrow \mathsf{Lf.iKG}(\mathsf{Lf.MSK}_i, f_b^{i,j}, j)$, and return $(\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j})$ to $\mathcal{A}$.
4. $\mathcal{A}$ 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 $\mathsf{Lf.MSK}_i$, a PRF key S_i, and a functional key $\mathsf{Rt}.sk_{e_i}$. 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 $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$ in $\mathcal{L}$. If so, the challenger responds to the key query using $(\mathsf{Lf.MSK}_i, S_i)$. Otherwise, the challenger generates r^i_{Setup}, S_i, and r^i_{iKG} as truly random strings, and generates $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$ and $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{MSK}, e[\mathsf{Lf.MSK}_i, S_i, 0], i)$. Then, the challenger responds to the key query, and finally adds the entry $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$ to $\mathcal{L}$.

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

For every $k^* \in [q]$, we define the following games. We define Game $(7, 0)$ as the same game as Game 2.

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 $\mathsf{Lf.MSK}^* \leftarrow \mathsf{Lf.Setup}(1^\lambda)$ and $S^* \leftarrow \{0,1\}^\lambda$. Then, for every $\ell \in [p]$, the challenger generates $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)}))$, where $u^{*(\ell)} \leftarrow \mathsf{Lf.Enc}(\mathsf{Lf.MSK}^*, m_b^\ell; r^{*(\ell)}_{\mathsf{Enc}})$ and $r^{*(\ell)}_{\mathsf{Enc}} \leftarrow \mathsf{F}'_{S^*}(t^{(\ell)})$.

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 $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$, the challenger responds to the query by using them. Otherwise, the challenger first generates $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda)$, $S_i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{MSK}, e[\mathsf{Lf.MSK}_i, S_i, 1], i)$. 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 $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$, the challenger responds to the query by using them. Otherwise, the challenger first

sets $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.MSK}^*$, $S_i \leftarrow S^*$, and $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}_i, e[\bot, \bot, 0], i)$. 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 $r_{\mathsf{Enc}}^{(i^*,\ell)}$ as a truly random string for every $\ell \in [p]$.

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

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

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

We define one additional game.

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

For every $h \in \{0, 1, 2, 8\}$, let $\mathtt{SUC}_h$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game h. Similarly, for every $h \in \{3, \cdots, 7\}$ and $k^* \in [q]$, let $\mathtt{SUC}_{(h,k^*)}$ 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 $|\Pr[\mathtt{SUC}_8] - \frac{1}{2}| = 0$. Then, we can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| + |\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \\
&\quad + \sum_{k^* \in [q]} |\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]| \\
&\quad + \sum_{k^* \in [q]} \sum_{h=3}^{6} |\Pr[\mathtt{SUC}_{(h,k^*)}] - \Pr[\mathtt{SUC}_{(h+1,k^*)}]| \\
&\quad + |\Pr[\mathtt{SUC}_{(7,q)}] - \Pr[\mathtt{SUC}_8]| \qquad (4.11)
\end{aligned}$$

Below, we estimate each term on the right side of inequality 4.11.

Lemma 4.4.1 $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \epsilon_{\mathsf{F}}$.

The proof of it is straightforward thus omitted.

Lemma 4.4.2 $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \epsilon_{\mathsf{Rt}}$.

Proof of Lemma 4.4.2. Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Root.

1. On input security parameter 1^λ, $\mathcal{B}$ chooses $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends 1^λ to $\mathcal{A}$.
2. When $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[q]}$, $\mathcal{B}$ sends $\left\{(m_b^\ell, \bot, t^{(\ell)}, \bot), (m_b^\ell, m_1^\ell, t^{(\ell)}, \bot)\right\}_{\ell\in[p]}$ to the challenger and returns the answer $\left\{\mathsf{Rt.CT}^{(\ell)}\right\}_{\ell\in[p]}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ first checks whether there is an entry whose first component is i in $\mathcal{L}$. If so, using the entry $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$, $\mathcal{B}$ responds to the key query. Otherwise, $\mathcal{B}$ first generates $r^i_{\mathsf{Setup}}, S_i, r^i_{\mathsf{iKG}} \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and computes $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda; r^i_{\mathsf{Setup}})$. Then, $\mathcal{B}$ queries $(i, e[\mathsf{Lf.MSK}_i, S_i, 0], e[\mathsf{Lf.MSK}_i, S_i, 0])$ to the challenger as a key query, and obtains the answer $\mathsf{Rt}.sk_{e_i}$. Next, $\mathcal{B}$ generates $\mathsf{Lf}.sk_f^{i,j} \leftarrow \mathsf{Lf.iKG}(\mathsf{Lf.MSK}_i, f_b^{i,j}; j)$ and returns $(\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j})$ to $\mathcal{A}$. Finally, $\mathcal{B}$ adds $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$ to $\mathcal{L}$.
4. 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 $e[\mathsf{Lf.MSK}_i, S_i, 0]$ ignores second component of the input, and thus for every $\ell \in [p]$ and every key query $i \in \mathcal{I}_{\mathsf{Rt}}$, we have $e[\mathsf{Lf.MSK}_i, S_i, 0](m_b^\ell, \bot, t^{(\ell)}, \bot) = e[\mathsf{Lf.MSK}_i, S_i, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, \bot)$. 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 Root, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Root},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. 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. Thus, we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Root},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]|$ and $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \epsilon_{\mathsf{Rt}}$ holds. □ (**Lemma 4.4.2**)

Lemma 4.4.3 *For every* $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]| \leq \epsilon_{\mathsf{Rt}}$.

Proof of Lemma 4.4.3. Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Root.

1. On input security parameter 1^λ, $\mathcal{B}$ chooses $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends 1^λ to $\mathcal{A}$.

2. When $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[q]}$, $\mathcal{B}$ generates $u^{*(\ell)} \leftarrow \mathsf{Lf.Enc}(\mathsf{Lf.MSK}^*, m_b^\ell; r_{\mathsf{Enc}}^{*(\ell)})$ for every $\ell \in [p]$, where $r_{\mathsf{Enc}}^{*(\ell)} \leftarrow \mathsf{F}_{S^*}(t^{(\ell)})$. $\mathcal{B}$ sends $\left\{(m_b^\ell, m_1^\ell, t^{(\ell)}, \bot), (m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)})\right\}_{\ell\in[p]}$ to the challenger and returns the answer $\left\{\mathsf{Rt.CT}^{(\ell)}\right\}_{\ell\in[p]}$ to $\mathcal{A}$.

3. When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ responds as follows.

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

4. 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

$$e[\mathsf{Lf.MSK}_i, S_i, \alpha](m_b^\ell, m_1^\ell, t^{(\ell)}, \bot) = e[\mathsf{Lf.MSK}_i, S_i, \alpha](m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)})$$

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

$$\begin{aligned} e[\mathsf{Lf.MSK}^*, S^*, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, \bot) &= \mathsf{Lf.Enc}(\mathsf{Lf.MSK}^*, m_b^\ell; r_{\mathsf{Enc}}^{*(\ell)}) \\ &= u^{*(\ell)} \\ &= e[\bot, \bot, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, u^{*(\ell)}) \end{aligned}$$

for every $\ell \in [p]$. Therefore, $\mathcal{B}$ is a valid adversary for Root, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Root},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. 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 $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Root},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]|$, and thus $|\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]| \leq \epsilon_{\mathsf{Rt}}$ holds. □ (**Lemma 4.4.3**)

Lemma 4.4.4 *For every* $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(3,k^*)}] - \Pr[\mathtt{SUC}_{(4,k^*)}]| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus omitted.

Lemma 4.4.5 *For every* $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(4,k^*)}] - \Pr[\mathtt{SUC}_{(5,k^*)}]| \leq \epsilon_{\mathsf{Lf}}$.

Proof of Lemma 4.4.5. Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Leaf.

1. On input security parameter 1^λ, $\mathcal{B}$ chooses $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends 1^λ to $\mathcal{A}$.
2. When $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell\in[q]}$, $\mathcal{B}$ sends $\left\{(m_b^\ell, m_1^\ell)\right\}_{\ell\in[p]}$ to the challenger and obtains the answer $\left\{\mathsf{Lf.CT}^{(\ell)}\right\}_{\ell\in[p]}$. $\mathcal{B}$ computes $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \mathsf{Lf.CT}^{(\ell)}))$, and returns $\left\{\mathsf{Rt.CT}^{(\ell)}\right\}_{\ell\in[p]}$ to $\mathcal{A}$.
3. When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ responds as follows.
   - In the case $|\mathcal{L}| < k^* - 1$, if there is an entry of the form $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$, $\mathcal{B}$ responds using them. Otherwise, $\mathcal{B}$ first generates $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda)$, $S_i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\mathsf{Lf.MSK}_i, S_i, 1], i)$. 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 $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$, $\mathcal{B}$ 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 $\mathsf{Lf}.sk_f^{i,j}$. Then, $\mathcal{B}$ generates $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\perp, \perp, 0], i)$, and returns $(\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j})$. Finally, $\mathcal{B}$ sets $i^* = i$ and adds $(i^*, \perp, \perp, \mathsf{Rt}.sk_{e_i})$ to $\mathcal{L}$.
   - In the case $|\mathcal{L}| > k^* - 1$, if there is an entry of the form $(i, \mathsf{Lf.MSK}_i, S_i, \mathsf{Rt}.sk_{e_i})$ 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 $\mathsf{Lf}.sk_f^{i,j}$, and returns $(\mathsf{Rt}.sk_{e_i}, \mathsf{Lf}.sk_f^{i,j})$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ first generates $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda)$, $S_i \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$, and $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\mathsf{Lf.MSK}_i, S_i, 0], i)$. Then, $\mathcal{B}$ responds using them, and adds them to $\mathcal{L}$.
4. 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 $j \in \mathcal{I}_{\mathsf{Lf}}$ at most. Therefore, $\mathcal{B}$ is a valid adversary for Leaf, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Leaf},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. 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 $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Leaf},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_{(4,k^*)}] - \Pr[\mathtt{SUC}_{(5,k^*)}]|$, and thus $|\Pr[\mathtt{SUC}_{(4,k^*)}] - \Pr[\mathtt{SUC}_{(5,k^*)}]| \leq \epsilon_{\mathsf{Lf}}$ holds. □ (**Lemma 4.4.5**)

Lemma 4.4.6 *For every* $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(5,k^*)}] - \Pr[\mathtt{SUC}_{(6,k^*)}]| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus omitted.

Lemma 4.4.7 *For every* $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(6,k^*)}] - \Pr[\mathtt{SUC}_{(7,k^*)}]| \leq \epsilon$.

The proof is almost the same as that of Lemma 4.4.3 thus is omitted.

Lemma 4.4.8 $|\Pr[\mathtt{SUC}_{(7,p)}] - \Pr[\mathtt{SUC}_8]| \leq \epsilon_{\mathsf{Rt}}$.

Proof of Lemma 4.4.8. The only difference between Game $(7, q)$ and 8 is how $\mathsf{Rt.CT}^{(\ell)}$ is generated for every $\ell \in [p]$. In Game $(7, q)$, it is generated as $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \bot))$. On the other hand, in Game 8, it is generated as $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (\bot, m_1^\ell, t^{(\ell)}, \bot))$. Here, in both games, for every $i \in \mathcal{I}_{\mathsf{Rt}}$, $\mathsf{Rt}.sk_{e_i}$ is generated as $\mathsf{Rt}.sk_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{MSK}, e[\mathsf{Lf.MSK}_i, S_i, 1], i)$, or not generated. Then, for every $i \in \mathcal{I}_{\mathsf{Rt}}$ and $\ell \in [p]$, we have

$$e[\mathsf{Lf.MSK}_i, S_i, 1](m_b^\ell, m_1^\ell, t^{(\ell)}, \bot) = e[\mathsf{Lf.MSK}_i, S_i, 1](\bot, m_1^\ell, t^{(\ell)}, \bot).$$

This is because m_b^ℓ is ignored in the left hand side. Therefore, we can construct an adversary attacking Root whose advantage is $|\Pr[\mathtt{SUC}_{(7,q)}] - \Pr[\mathtt{SUC}_8]|$, and thus $|\Pr[\mathtt{SUC}_{(7,q)}] - \Pr[\mathtt{SUC}_8]| \leq \epsilon$ holds. □ (**Lemma 4.4.8**)

From inequality 4.11 and lemmas 4.4.1 to 4.4.8, inequality 4.10 holds. This completes the proof. □ (**Theorem 4.4.1**)

4.5 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 an iSKFE scheme the size of whose index space is $\lambda^{\omega(1)}$ then transform the iSKFE scheme into a standard SKFE scheme (i.e., stateless scheme).

We basically expand the index space by using our new PRODUCT construction introduced in Section 4.4 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 $\mathsf{1CT}$ introduced in Section 4.3.3 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.

4.5.1 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 $\mathsf{Parallel}_\lambda$ from the underlying single-key SKFE scheme. This is done by simply running λ instances of the single-key scheme as in Section 4.3.1. For simplicity, we assume that the underlying single-key scheme is fully succinct, and the encryption time of $\mathsf{Parallel}_\lambda$ is bounded by $|m|^c + O(\lambda^c)$, 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 $\mathsf{Parallel}_\lambda$ using our PRODUCT construction in Section 4.4. The encryption time of PRDCT_2 is roughly

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

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 $\mathsf{1CT}$ be SKFE constructed in Section 4.3.2. For simplicity, we suppose that we can bound the encryption time of $\mathsf{1CT}$ 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 $|m|^{c^2} + O\left(\lambda^{c^2}\right)$ and a fresh instance of $\mathsf{1CT}$ via the hybrid construction in Section 4.3.3.

Recall that the length of a master secret key of $\mathsf{1CT}$ 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

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

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 Root and *a fresh instance* of $\mathsf{Parallel}_\lambda$ as Leaf. In this case, a ciphertext of $\mathsf{Parallel}_\lambda$ is embedded into a ciphertext of HYBRD_2 in the security proof. Therefore, the encryption time of PRDCT_3 is roughly

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

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 $|m|^{c^2} + O\left(\lambda^{c^2}\right)$ no matter how many times we iterate the construction. Of course, the term $O\left(\lambda^{c^2}\right)$ 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 4.5.1 (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* Root *and* a fresh instance *of* $\mathsf{Parallel}_\lambda$ *as* Leaf*. 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 Root *and* Leaf *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* Root *and* Leaf *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.

4.5.2 Construction of Collusion-Resistant iSKFE

In order to precisely define our collusion-resistant iSKFE, we introduce some notations. We let

$$\langle \mathsf{iSKFE}_{\mathsf{Rt}}, \mathsf{iSKFE}_{\mathsf{Lf}} \rangle_{\mathsf{product}} = \mathsf{iSKFE}$$

denote that an iSKFE scheme iSKFE is constructed from our new PRODUCT construction in Section 4.4 by using $\mathsf{iSKFE}_{\mathsf{Rt}}$ as Root and $\mathsf{iSKFE}_{\mathsf{Lf}}$ as Leaf. Moreover, we let

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

denote that an SKFE scheme SKFE' is constructed from our proposed hybrid encryption construction introduced in Section 4.3.3 by using iSKFE as a building block iSKFE scheme together with $\mathsf{1CT}$.

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

First, we construct an iSKFE scheme $\mathsf{Parallel}_\lambda$ by applying the parallel construction introduced in Section 4.3.1 that the number of parallelization is λ. That is, we set $q = \lambda$ and use $\mathsf{1Key}$ as a building block in the construction introduced in Section 4.3.1. Note that the index space of $\mathsf{Parallel}_\lambda$ is $[\lambda]$.

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

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

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 is HYBRD_η, where $\eta = \omega(1)$. HYBRD_η is based solely on $\mathsf{1Key}$, and the following theorem holds.

Theorem 4.5.1 *Assuming there exists* $(1,\delta)$*-selective-message function private SKFE that is weakly-succinct, where* $\delta(\lambda) = \lambda^{-\zeta}$ *and* $\zeta = \omega(1)$*. Then, 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*.*

4.5.3 Analysis for Security Bound and Efficiency

In this section, we formally analyze the security and efficiency of HYBRD_η and prove Theorem 4.5.1.

Proof of Theorem 4.5.1. We start with the security bound analysis and then move to the efficiency analysis.

Security bound analysis. We assume that the advantages of any adversary attacking $\mathsf{Parallel}_\lambda$, $\mathsf{1CT}$, and F is bounded by ϵ_{Para}, ϵ_{1CT}, and ϵ_{F}, respectively. For every $k \in [\eta]$, let $\epsilon_{\mathsf{Hyb}_k}$ and $\epsilon_{\mathsf{Prd}_k}$ be the upper bounds of the advantage of any adversary attacking HYBRD_k and PRDCT_k, respectively. Then, from inequalities 4.7 and 4.10, for every $k \in [\eta-1]$, we have

$$\begin{aligned}
\epsilon_{\mathsf{Hyb}_{k+1}} &\le 2\left((2p+1)\epsilon_{\mathsf{Prd}_{k+1}} + p\cdot\epsilon_{\mathsf{1CT}} + p\cdot\epsilon_{\mathsf{F}}\right)\\
&\le 4(2p+1)\left((2q+2)\epsilon_{\mathsf{Hyb}_k} + q\cdot\epsilon_{\mathsf{Para}} + (2q+1)\epsilon_{\mathsf{F}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}}\right)\\
&\le 8(2p+1)(q+1)\cdot\epsilon_{\mathsf{Hyb}_k} + 8(2p+1)(q+1)\left(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}}\right),
\end{aligned}$$

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

$$\epsilon_{\mathsf{Hyb}_{k+1}} \le Q\cdot\epsilon_{\mathsf{Hyb}_k} + Q\cdot\left(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}}\right).$$

Therefore, it holds that

$$\begin{aligned}
\epsilon_{\mathsf{Hyb}_\eta} &\le Q^{\eta-1}\cdot\epsilon_{\mathsf{Hyb}_1} + \left(\sum_{k=0}^{\eta-2} Q^k\right)\cdot Q(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}})\\
&\le Q^{\eta-1}\cdot 2\left((p+1)\cdot\epsilon_{\mathsf{Para}} + p\cdot\epsilon_{\mathsf{1CT}} + p\cdot\epsilon_{\mathsf{F}}\right)\\
&\qquad + (\eta-2)\cdot Q^{\eta-2}\cdot(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}})\\
&\le (\eta-2)\cdot Q^{\eta-1}\cdot 3(p+1)\cdot(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}}).
\end{aligned}$$

Since p and q are polynomials of λ, and $Q = 8(2p+1)(q+1)$, we get

$$\epsilon_{\mathsf{Hyb}_\eta} \le \lambda^{O(\eta)}\cdot(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{F}}). \tag{4.12}$$

Our assumption is that $\mathsf{1Key}$ is a $(1,\delta)$-selective-message function private SKFE scheme, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, from Theorem 4.3.1, $\mathsf{Parallel}_\lambda$ 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 $\mathsf{1CT}$ since it can be constructed based only on δ-secure one-way functions as we show in Section 4.3.2. 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

$$\epsilon_{\mathsf{Hyb}_\eta} \leq \lambda^{O(\eta)} \cdot \delta^{\Omega(1)} = \lambda^{O(\zeta^{1/2})} \cdot (\lambda^{-\zeta})^{\Omega(1)} \leq (\lambda^{-\zeta})^{\Omega(1)} = \delta^{\Omega(1)}.$$

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 4.2.1, 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 $\mathrm{loglog}\lambda$ when transforming quasi-polynomially (resp. sub-exponentially) secure single-key SKFE scheme. See Section 4.5.4 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 $\mathsf{Parallel}_\lambda$ and $\mathsf{1CT}$ when constructing PRDCT_k and HYBRD_k, respectively. In the following, we denote these two schemes as $\mathsf{Parallel}_\lambda^{(k)}$ and $\mathsf{1CT}^{(k)}$ to emphasize that they are instances of $\mathsf{Parallel}_\lambda$ and $\mathsf{1CT}$ 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 $\mathsf{Parallel}_\lambda$ and $\mathsf{1CT}$ 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 $\eta = O(\mathrm{polylog}(\lambda))$ (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 $\mathsf{Parallel}_\lambda^{(k)}$ and $\mathsf{1CT}^{(k)}$ 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 $\mathsf{Parallel}_\lambda^{(k)}$ for every $k \in [\eta]$.

Before analysis, we bound the size of the key generation circuit G and encryption circuit e defined in Figure 4.3 and 4.4, respectively. Below, let $G[f]$ denote the key generation circuit $G[f, \bot, \bot, i]$, 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 $e[\mathsf{Enc}]$ denote the encryption circuit computing Enc (and one PRF evaluation). From the bounds 4.6 and 4.9 that we show in Section 4.3.3 and 4.4, we can bound the size of G and e as,

$$|G[f]| \leq (|f| + n_f) \cdot \mathsf{poly}_G(\lambda, \log q), \tag{4.13}$$

$$|e[\mathsf{Enc}]| \leq |\mathsf{Enc}| \cdot \mathsf{poly}_e(\lambda), \tag{4.14}$$

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 $\mathsf{Para}.t_{\mathsf{Enc}}^{(k)}$ be the bound of the running time of the encryption algorithm of $\mathsf{Parallel}_\lambda^{(k)}$. Since $\mathsf{1Key}$ is weakly succinct, from the bound 4.1 in Section 4.3.1, for every $k \in [\eta]$, $\mathsf{Para}.t_{\mathsf{Enc}}^{(k)}$ is bounded as

$$\mathsf{Para}.t_{\mathsf{Enc}}^{(k)} \leq |f|^\gamma \cdot \mathsf{poly}_{\mathsf{Para}}(\lambda, n_f), \tag{4.15}$$

where f is a function supported by $\mathsf{Parallel}_\lambda^{(k)}$, n_f is the input length of f, $\gamma < 1$ is a constant, and $\mathsf{poly}_{\mathsf{Para}}$ is a fixed polynomial. As mentioned above, for every $k \in [\eta]$, $\mathsf{Parallel}_\lambda^{(k)}$ denotes different instances of the same scheme $\mathsf{Parallel}_\lambda$ and thus $\mathsf{poly}_{\mathsf{Para}}$ is independent of k.

For every $k \in \{2, \cdots, \eta\}$, $\mathsf{Parallel}_\lambda^{(k-1)}$ has to generate a functional key tied to the circuit $G[e[\mathsf{Para}.\mathsf{Enc}^{(k)}]]$. Therefore, from 4.15, we have

$$\mathsf{Para}.t_{\mathsf{Enc}}^{(k-1)} \leq |G[e[\mathsf{Para}.\mathsf{Enc}^{(k)}]]|^\gamma \cdot \mathsf{poly}_{\mathsf{Para}}(\lambda, 2\lambda + 1). \tag{4.16}$$

Here, the input length of the circuit $e[\mathsf{Para}.\mathsf{Enc}^{(k)}]$ is bounded by $2(2\lambda+1)+\lambda+|\mathsf{Para}.\mathsf{Enc}^{(k)}|$ since the length of a ciphertext output by $\mathsf{Para}.\mathsf{Enc}^{(k)}$ is bounded by the size of $\mathsf{Para}.\mathsf{Enc}^{(k)}$. Then, from 4.13 and 4.14, for every $k \in [\eta]$, we can estimate $|G[e[\mathsf{Para}.\mathsf{Enc}^{(k)}]]|$ as

$$\begin{aligned}
|G[e[\mathsf{Para}.\mathsf{Enc}^{(k)}]]| &\leq \Big(|e[\mathsf{Para}.\mathsf{Enc}^{(k)}]| + (2(2\lambda+1) + \lambda + |\mathsf{Para}.\mathsf{Enc}^{(k)}|)\Big) \cdot \mathsf{poly}_G(\lambda, \log \lambda^k) \\
&= \Big(|\mathsf{Para}.\mathsf{Enc}^{(k)}| \cdot \mathsf{poly}_e(\lambda) + ((2(2\lambda+1) + \lambda + |\mathsf{Para}.\mathsf{Enc}^{(k)}|)\Big) \cdot \\
&\quad \mathsf{poly}_G(\lambda, \log \lambda^k) \\
&\leq |\mathsf{Para}.\mathsf{Enc}^{(k)}| \cdot \mathsf{poly}_1(\lambda, k) \leq |\mathsf{Para}.\mathsf{Enc}^{(k)}| \cdot \mathsf{poly}_1(\lambda, \eta),
\end{aligned} \tag{4.17}$$

where poly_1 is a fixed polynomial. Thus, from 4.16 and 4.17, for every $k \in \{2, \cdots, \eta\}$, we have

$$\begin{aligned}
\mathsf{Para}.t_{\mathsf{Enc}}^{(k-1)} &\leq \Big(|\mathsf{Para}.\mathsf{Enc}^{(k)}| \cdot \mathsf{poly}_1(\lambda, \eta)\Big)^\gamma \cdot \mathsf{poly}_{\mathsf{Para}}(\lambda, 2\lambda + 1) \\
&\leq |\mathsf{Para}.\mathsf{Enc}^{(k)}|^\gamma \cdot \mathsf{poly}_2(\lambda, \eta) = (\mathsf{Para}.t_{\mathsf{Enc}}^{(k)})^\gamma \cdot \mathsf{poly}_2(\lambda, \eta),
\end{aligned} \tag{4.18}$$

where poly_2 is also a fixed polynomial. In the last equality, we consider $|\mathsf{Para.Enc}^{(k)}|$ equals its running time.

In addition, $\mathsf{Parallel}_\lambda^{(\eta)}$ has to release a functional key tied to a circuit G in which a function supported by HYBRD_η is hardwired. Therefore, by using 4.13 and 4.15 again, we can bound $\mathsf{Para}.t_{\mathsf{Enc}}^{(\eta)}$ as

$$\begin{aligned}\mathsf{Para}.t_{\mathsf{Enc}}^{(\eta)} &\leq ((s+n)\cdot \mathsf{poly}_G(\lambda, \log \lambda^\eta))^\gamma \cdot \mathsf{poly}_{\mathsf{Para}}(\lambda, 2\lambda+1) \\ &\leq s^\gamma \cdot \mathsf{poly}_3(\lambda, n, \eta), \end{aligned} \tag{4.19}$$

where poly_3 is also a fixed polynomial.

From 4.18 and 4.19, for every $k \in [\eta]$, it holds that

$$\begin{aligned}\mathsf{Para}.t_{\mathsf{Enc}}^{(k)} &\leq (s^\gamma \cdot \mathsf{poly}_3(\lambda, n, \eta))^{\gamma^{\eta-k}} \cdot \prod_{j=0}^{\eta-k-1} \mathsf{poly}_2(\lambda, \eta)^{\gamma^j} \\ &\leq s^\gamma \cdot \mathsf{poly}_3(\lambda, n, \eta) \cdot \mathsf{poly}_2(\lambda, \eta)^{\frac{1}{1-\gamma}} \leq s^\gamma \cdot \mathsf{poly}_4(\lambda, n, \eta), \end{aligned} \tag{4.20}$$

where poly_4 is a fixed polynomial. The second inequality comes from the fact $\gamma < 1$. Thus, we see that the encryption algorithm of $\mathsf{Parallel}_\lambda^{(k)}$ for every $k \in [\eta]$ runs in polynomial of λ, n, η and s.

Next, we analyze the encryption time of $\mathsf{1CT}^{(k)}$ for every $k \in [\eta]$. For every $k \in [\eta]$, let $\mathsf{1CT}.t_{\mathsf{Enc}}^{(k)}$ be the bound of the running time of the encryption algorithm of $\mathsf{1CT}^{(k)}$. For every $k \in [\eta]$, $\mathsf{1CT}^{(k)}$ generates a functional key tied to the circuit $e[\mathsf{Para.Enc}^{(k)}]$. From 4.20, the input length of $e[\mathsf{Para.Enc}^{(k)}]$ is bounded by

$$\begin{aligned}2(2\lambda+1) + \lambda + \left|\mathsf{Para.Enc}^{(k)}\right| &\leq 2(2\lambda+1) + \lambda + s^\gamma \cdot \mathsf{poly}_4(\lambda, n, \eta) \\ &\leq s^\gamma \cdot \mathsf{poly}_5(\lambda, n, \eta),\end{aligned}$$

where poly_5 is a fixed polynomial. Then, from the quasi-linear efficiency of $\mathsf{1CT}$ that we show as 4.3 in Section 4.3.2, for every $k \in [\eta]$, we have

$$\begin{aligned}\mathsf{1CT}.t_{\mathsf{Enc}}^{(k)} &\leq (s^\gamma \cdot \mathsf{poly}_5(\lambda, n, \eta)) \cdot \mathsf{poly}_{\mathsf{1CT}}(\lambda, \log(s^\gamma \cdot \mathsf{poly}_5(\lambda, n))) \\ &\leq s^{\gamma'} \cdot \mathsf{poly}_6(\lambda, n, \eta), \end{aligned} \tag{4.21}$$

where $\mathsf{poly}_{\mathsf{1CT}}$ 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 $\mathsf{1CT}^{(k)}$ runs in polynomial of λ, n, η, and s.

From 4.20 and 4.21, the encryption time of HYBRD_η is bounded by

$$\eta \cdot \left(s^\gamma \cdot \mathsf{poly}_5(\lambda, n, \eta) + s^{\gamma'} \cdot \mathsf{poly}_6(\lambda, n, \eta)\right) \leq s^{\gamma'} \cdot \mathsf{poly}_7(\lambda, n, \eta), \tag{4.22}$$

where poly_7 is a fixed polynomial.

As stated earlier, η is at most sub-linear in λ in the construction. Thus, 4.22 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, 4.22 means that HYBRD_η is weakly succinct.[6]

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. □ (**Theorem 4.5.1**)

4.5.4 Converting iSKFE into SKFE

Finally, we show how to transform iSKFE into SKFE. We provide the overview of this transformation in Remark 4.2.1 in Section 4.2. Here, we give the formal description of the transformation and prove its security. Let $\mathsf{iSKFE} = (\mathsf{Setup}, \mathsf{iKG}, \mathsf{Enc}, \mathsf{Dec})$ be an iSKFE scheme whose index space is $\mathcal{I}$. We construct an SKFE scheme $\mathsf{SKFE} = (\mathsf{Setup}', \mathsf{KG}, \mathsf{Enc}', \mathsf{Dec}')$ as follows. Setup', Enc', and Dec' are exactly the same as Setup, Enc, and Dec, respectively. We define KG as follows.

$\mathsf{KG}(\mathsf{MSK}, f)$:

- Generate $i \xleftarrow{r} \mathcal{I}$ and return $sk_f \leftarrow \mathsf{iKG}(\mathsf{MSK}, f, i)$.

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 4.5.2 *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 4.5.2. 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

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) \leq 2\left(\frac{q(q-1)}{S} + \frac{\epsilon}{2}\right). \tag{4.23}$$

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 4.23. Consider the following two games.

[6] Analogously, we see that if the underlying single-key SKFE is succinct, then so does HYBRD_η.

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 $\mathcal{L}$ which stores an index $i \in \mathcal{I}$. When $\mathcal{A}$ sends a function f as a key query, the challenger first generates an index $i \xleftarrow{r} \mathcal{I}$, and checks whether the list $\mathcal{L}$ contains the index i or not. If so, the challenger generates $i' \xleftarrow{r} \mathcal{I} \setminus \mathcal{L}$ and returns $sk_f \leftarrow \mathsf{iKG}(\mathsf{MSK}, f, i')$ to $\mathcal{A}$. Otherwise, the challenger returns $sk_f \leftarrow \mathsf{iKG}(\mathsf{MSK}, f, i)$ to $\mathcal{A}$ and adds i to $\mathcal{L}$.

Let $\mathtt{SUC}_0$ (resp. $\mathtt{SUC}_1$) be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game 0 (resp. Game 1). Let $\mathsf{Collision}$ 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 $\mathsf{Collision}$ occurs. Therefore, we have $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \Pr[\mathsf{Collision}]$. In addition, $\Pr[\mathsf{Collision}]$ is bounded by $\frac{q(q-1)}{S}$ using the union bound. Thus, $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \frac{q(q-1)}{S}$ holds.

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 $2|\Pr[\mathtt{SUC}_1] - \frac{1}{2}|$. Thus, $|\Pr[\mathtt{SUC}_1] - \frac{1}{2}| \leq \frac{\epsilon}{2}$.

Then, we can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}
\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\
&\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| + |\Pr[\mathtt{SUC}_1] - \frac{1}{2}| && (4.24) \\
&\leq \frac{q(q-1)}{S} + \frac{\epsilon}{2}. && (4.25)
\end{aligned}$$

From the inequality 4.25, we obtain inequality 4.23. □ (**Theorem 4.5.2**)

4.5.5 From Single-Key SKFE to Collusion-Resistant SKFE

We combine the results that we proved through Section 4.5 to give our main result. First, we combine Theorem 4.5.1 and 4.5.2, and obtain the following main theorem.

Theorem 4.5.3 *Assuming there exists $(1, \delta)$-selective-message function private SKFE for all circuits that is weakly-succinct, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, there exists (poly, δ')-selective-message function private SKFE for all circuits, where $\delta'(\lambda) = \lambda^{-\zeta^{1/2}}$.*[7]

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

Theorem 4.5.3 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 4.5.4 *Assuming there exists $(1, \delta)$-selective-message function private SKFE for all circuits 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 all circuits, where $\delta'(\lambda) = 2^{-\lambda^{\gamma/2}}$.*

Therefore, by combining Theorem 4.5.4 with Theorem 3.5.3, we obtain the following corollary.

Corollary 4.5.1 *Assuming there exist sub-exponentially secure single-key weakly-succinct SKFE for all circuits. Then, there exists IO for all circuits.*

4.6 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 weakly-selective-message message private SKFE scheme into selective-message function private one with polynomial security loss.

By accommodating these upgrades into Theorem 4.5.3 and 4.5.4, we obtain the following corollary.

Corollary 4.6.1 *If there exists quasi-polynomially (resp. sub-exponentially) secure single-key SKFE that is weakly-selective-message 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.

4.6.1 From Weakly-Succinct to Succinct

Ananth et al. [AJS15] proved that we can transform a collusion-resistant SKFE scheme into a succinct one. By using this result and Theorem 4.5.3, 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 4.5.2 for $\eta = O(1)$.

By setting $\eta = O(1)$ in the construction of HYBRD_η, from the inequality 4.12, we can construct an iSKFE scheme supporting λ^η functional keys with polynomial security loss. Let $q = \lambda^\eta$. Then, from the inequality 4.22, 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 4.5.3.

Theorem 4.6.1 *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 as we show in Section 3.4.2, which is originally used by Bitansky and Vaikuntanathan [BV15, Proposition IV.1]. Formally, the following theorem holds.

Theorem 4.6.2 *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.*

Below, we show the actual transformation.

We construct a single-key SKFE scheme $\mathsf{SCNCT} = (\mathsf{SCT.Setup}, \mathsf{SCT.KG}, \mathsf{SCT.Enc}, \mathsf{SCT.Dec})$ based on the following building blocks. Let s and n be the maximum size and input length of functions supported by SCNCT, respectively. Let RE be a c-local decomposable randomized encoding, where c is a constant. We suppose that the number of decomposed encodings and the length of randomness of RE are μ and ρ, respectively. Then, both μ and ρ are bounded by $s \cdot \mathsf{poly}_{\mathsf{RE}}(\lambda, n)$, where $\mathsf{poly}_{\mathsf{RE}}$ is a fixed polynomial. Let $\mathsf{iSKFE} = (\mathsf{Setup}, \mathsf{iKG}, \mathsf{Enc}, \mathsf{Dec})$ be an iSKFE scheme whose index space is $[\mu]$. Let $\{\mathsf{F}_K(\cdot) : \{0,1\}^\lambda \times [\rho] \to \{0,1\} | K \in \{0,1\}^\lambda\}$ be a PRF. The construction of SCNCT is as follows.

Construction. The scheme consists of the following algorithms.

$\mathsf{SCT.Setup}(1^\lambda)$:

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

$\mathsf{SCT.KG}(\mathsf{MSK}, f)$:

- Generate $t \leftarrow \{0,1\}^\lambda$.

Decomposable randomized encoding circuit $D_{\mathsf{re}}[\hat{f}_{0,i}, \hat{f}_{1,i}, S_{0,i}, S_{1,i}, t, u_i](m, K, \alpha)$

Hardwired: decomposed functions $\hat{f}_{0,i}$ and $\hat{f}_{1,i}$, sets $S_{0,i}$ and $S_{1,i}$, tag t, and functional key u_i.
Input: message m, PRF key K, and bit α.

1. If $K = \perp$, return u_i.
2. Else for $j \in S_{i,\alpha}$, compute $r_j \leftarrow \mathsf{F}_K(t\|j)$, set $r_{S_{i,\alpha}} \leftarrow \{r_j\}_{j \in S_{i,\alpha}}$, where r_j is the j-th bit of $r_{S_{i,\alpha}}$.
3. Return $e_i \leftarrow \hat{f}_{i,\alpha}(x; r_{S_{i,\alpha}})$.

Figure 4.5: Construction of decomposable randomized encoding circuit D_{re}.

- Compute decomposed f, that is, $(\hat{f}_1, \cdots, \hat{f}_\mu)$ together with $(S_1, \cdots, S_\mu)$ where $S_i \subseteq [\rho]$ and $|S_i| = c$.
- Compute $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\hat{f}_i, \perp, S_i, \perp, t, \perp], i)$. The circuit D_{re} is defined in Figure 4.5.
- Return $\mathsf{sk}_f \leftarrow (\mathsf{sk}_{f_1}, \cdots, \mathsf{sk}_{f_\mu})$.

$\mathsf{SCT.Enc}(\mathsf{MSK}, m)$:

- Generate $K \leftarrow \{0,1\}^\lambda$.
- Return $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m, K, 0))$.

$\mathsf{SCT.Dec}(\mathsf{sk}_f, \mathsf{CT})$:

- Parse $(\mathsf{sk}_{f_1}, \cdots, \mathsf{sk}_{f_\mu}) \leftarrow \mathsf{sk}_f$.
- For every $i \in [\mu]$, compute $e_i \leftarrow \mathsf{Dec}(\mathsf{sk}_{f_i}, \mathsf{CT})$.
- Decode y from $(e_1, \cdots, e_\mu)$.
- Return y.

The correctness of SCNCT directly follows from that of iSKFE. Then, we have the following theorem.

Theorem 4.6.3 *Let* iSKFE *be a* (μ, δ)*-selective-message function private iSKFE scheme that is collusion succinct. Let* RE *a be* δ*-secure decomposable randomized encoding. Let* F *be a* δ*-secure PRF. Then,* SCNCT *is a* $(1, \delta)$*-selective-message function private SKFE that is succinct.*

Proof of Theorem 4.6.3. We start with analyzing the succinctness of SCNCT, and then move on to the security proof.

Succinctness. Let D_i be the circuit $D_{\mathsf{re}}[\hat{f}_{i,0}, \hat{f}_{i,1}, S_{i,0}, S_{i,1}, t, u_i]$. D_i includes at most c times PRF evaluation on the domain $\{0,1\}^\lambda \times [\rho]$ and single evaluation of $\hat{f}_{i,0}$ or $\hat{f}_{i,1}$. $\left|\hat{f}_{i,0}\right|$ and $\left|\hat{f}_{i,1}\right|$ are independent of $|f|$, and the size of $S_{i,0}, S_{i,1}, t$, and u_i are bounded by $O(\lambda)$ from the decomposability of RE. Therefore, the size of D_i is bounded by $\mathsf{poly}_D(\lambda, n, \log s)$, where poly_D is a polynomial. Since iSKFE is collusion succinct, the encryption time of SCNCT is bounded by

$$\begin{aligned}\mathsf{poly}(\lambda, n, |D_i|, \log \mu) &\leq \mathsf{poly}(\lambda, n, \mathsf{poly}_D(\lambda, n, \log s), \log(s \cdot \mathsf{poly}_{\mathsf{RE}}(\lambda, n))) \\ &\leq \mathsf{poly}'(\lambda, n, \log s),\end{aligned}$$

where poly and poly' are polynomials. This implies that SCNCT is succinct.

Security proof. We assume that the advantages of any adversary attacking iSKFE, RE, and F are bounded by ϵ, ϵ_{RE}, and ϵ_{F}, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of SCNCT. We assume that $\mathcal{A}$ makes p encryption queries at most, where p is a polynomial of λ. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{SCNCT},\mathcal{A}}(\lambda) \leq 2\left((2p+1)\cdot\epsilon + 2p\cdot\epsilon_{\mathsf{RE}} + 2p\cdot\epsilon_{\mathsf{F}}\right). \tag{4.26}$$

This means that if all of iSKFE, RE, and F are δ-secure, then so is SCNCT. Below, we prove the above inequality 4.26 via the following sequence of games.

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

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

Key query $\mathcal{A}$ can make only one key query (f_0, f_1). When $\mathcal{A}$ makes the query, the challenger first generates $t \leftarrow \{0,1\}^\lambda$ and computes decomposed f_b, that is, $(\hat{f}_{b,1}, \cdots, \hat{f}_{b,\mu})$ together with $(S_{b,1}, \cdots, S_{b,\mu})$. Then, the challenger computes $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\hat{f}_{b,i}, \bot, S_{b,i}, \bot, t, \bot], i)$ for every $i \in [\mu]$, and returns $\mathsf{sk}_f \leftarrow (\mathsf{sk}_{f_1}, \cdots, \mathsf{sk}_{f_\mu})$ to $\mathcal{A}$.

Final phase $\mathcal{A}$ output b'.

For every $\ell^* \in [p]$, we define the following games. We define Game $(5, \ell^* - 1)$ as the same game as Game 0.

Game $(1, \ell^*)$ Same as Game $(5, \ell^* - 1)$ except the following. The challenger generates $\left\{\mathsf{CT}^{(\ell)}\right\}_{\ell \in [p]}$ as follows.

- For every $\ell < \ell^* - 1$, the challenger generates $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m_1^{(\ell)}, K^{(\ell)}, 1))$.
- The challenger generates $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\bot, \bot, 0))$.
- For every $\ell > \ell^*$, the challenger generates $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m_b^{(\ell)}, K^{(\ell)}, 0))$.

In addtion, for the key query (f_0, f_1), the challenger responds as follows. First, the challenger generates $t \leftarrow \{0,1\}^\lambda$ and for $\alpha \in \{0,1\}$, computes decomposed f_α, that is, $(\hat{f}_{\alpha,1}, \cdots, \hat{f}_{\alpha,\mu})$ together with $(S_{\alpha,1}, \cdots, S_{\alpha,\mu})$. Then, the challenger computes $r_j^{(\ell^*)} \leftarrow \mathsf{F}_{K^{(\ell^*)}}(t \| j)$ for every $j \in [\rho]$. Next, for every $i \in [\mu]$, the challenger sets $r_{S_{i,b}}^{(\ell^*)} \leftarrow \{r_j^{(\ell^*)}\}_{j \in S_{i,b}}$, and computes $u_i^{(\ell^*)} \leftarrow \hat{f}_{i,b}(m_b^{\ell^*}; r_{S_{i,b}}^{(\ell^*)})$ and $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}], i)$. Finally, the challenger returns $sk_f \leftarrow (sk_{f_1}, \cdots, sk_{f_\mu})$ to $\mathcal{A}$.

Gam $(2, \ell^*)$ Same as Game $(1, \ell^*)$ except that for every $j \in [\rho]$, $r_j^{(\ell^*)}$ is generated as a truly random string.

Game $(3, \ell^*)$ Same as Game $(2, \ell^*)$ except that for every $i \in [\mu]$, the challenger generates $u_i^{(\ell^*)} \leftarrow \mathsf{Sim}(1^\lambda, s, y)$, where $s = |f_0| = |f_1|$ and $y = f_0(m_0^{\ell^*}) = f_1(m_1^{\ell^*})$. Here, Sim is a simulator for RE.

Game $(4, \ell^*)$ Same as Game $(3, \ell^*)$ except that for every $i \in [\mu]$, the challenger generates $u_i^{(\ell^*)} \leftarrow \hat{f}_{i,1}(m_1^{\ell^*}; r_{S_{i,1}})$.

Game $(5, \ell^*)$ Same as Game $(4, \ell^*)$ except that for every $j \in [\rho]$, the challenger generates $r_j^{(\ell^*)} \leftarrow \mathsf{F}_{K^{(\ell^*)}}(t \| j)$.

Game $(6, \ell^*)$ Same as Game $(5, \ell^*)$ except that the challenger generates $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m_1^{(\ell^*)}, K^{(\ell^*)}, 1))$. In addition, for every $i \in [\mu]$, the challenger generates $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot], i)$.

We define one additional game.

Game 7 Same as Game $(6, p)$ except that for every $i \in [\mu]$, the challenger computes $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\bot, \hat{f}_{1,i}, \bot, S_{1,i}, t, \bot], i)$. Note that in this game, for every $\ell \in [p]$, the challenger generates $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m_1^{\ell}, K^{(\ell)}, 1))$.

Let $\mathtt{SUC}_0$ and $\mathtt{SUC}_7$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit b in Game 0 and 7, respectively. Similarly, for every $h \in \{1, \cdots, 6\}$ and $\ell^* \in [p]$, let $\mathtt{SUC}_{(h,\ell^*)}$ be the event that $\mathcal{A}$ succeeds in guessing b in Game (h, ℓ^*).

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

$$\begin{aligned}\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\leq \sum_{\ell^* \in [p]} |\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \\ &\quad + \sum_{\ell^* \in [p]} \sum_{h=1}^{5} |\Pr[\mathtt{SUC}_{(h,\ell^*)}] - \Pr[\mathtt{SUC}_{(h+1,\ell^*)}]| \\ &\quad + |\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7]|. \end{aligned} \tag{4.27}$$

Below, we estimate each term on the right side of inequality 4.27.

Lemma 4.6.1 *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \leq \epsilon$.*

Proof of Lemma 4.6.1. 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 $b \xleftarrow{\mathsf{r}} \{0,1\}$. When, $\mathcal{A}$ sends $\left\{(m_0^\ell, m_1^\ell)\right\}_{\ell \in [q]}$, $\mathcal{B}$ sets $\left\{(M_0^\ell, M_1^\ell)\right\}_{\ell \in [p]}$ as follows.

- For every $\ell < \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (m_1^\ell, K^{(\ell)}, 1)$.
- $\mathcal{B}$ sets $M_0^{\ell^*} = (m_b^{\ell^*}, K^{(\ell^*)}, 0)$ and $M_1^{\ell^*} = (\bot, \bot, 0)$.
- For every $\ell > \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (m_b^\ell, K^{(\ell)}, 0)$.

Then, $\mathcal{B}$ sends $\left\{(M_0^\ell, M_1^\ell)\right\}_{\ell \in [p]}$ to the challenger and returns the answer $\left\{\mathsf{CT}^{(\ell)}\right\}_{\ell \in [p]}$ to $\mathcal{A}$.

Key queries For the key query (f_0, f_1), $\mathcal{B}$ first generates $t \leftarrow \{0,1\}^\lambda$. Then, for $\alpha \in \{0,1\}$, $\mathcal{B}$ computes decomposed f_α, that is, $(\hat{f}_{\alpha,1}, \cdots, \hat{f}_{\alpha,\mu})$ together with $(S_{\alpha,1}, \cdots, S_{\alpha,\mu})$. Then, $\mathcal{B}$ computes $r_j^{(\ell^*)} \leftarrow \mathsf{F}_{K^{(\ell^*)}}(t \| j)$ for every $j \in [\rho]$. Next, for every $i \in [\mu]$ and, the challenger sets $r_{S_{\alpha,i}}^{(\ell^*)} \leftarrow \{r_j^{(\ell^*)}\}_{j \in S_{\alpha,i}}$ for $\alpha \in \{0,1\}$, and computes $u_i^{(\ell^*)} \leftarrow \hat{f}_{b,i}(m_b^{\ell^*}; r_{S_{b,i}}^{(\ell^*)})$. Then, for every $i \in [\mu]$, $\mathcal{B}$ queries $(i, D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot], D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}])$ to the challenger and obtains the answer sk_{f_i}. $\mathcal{B}$ returns $sk_f \leftarrow (sk_{f_1}, \cdots, sk_{f_\mu})$ 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 $i \in [\mu]$ and $\ell \in [p]$, we have

$$D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot](m_b^\ell, K^{(\ell)}, \alpha) = D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}](m_b^\ell, K^{(\ell)}, \alpha),$$

for every $b, \alpha \in \{0,1\}$ if $K^{(\ell)} \neq \bot$ since $u_i^{(\ell^*)}$ is ignored in the right hand side. In addition, it holds that

$$\begin{aligned} D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot](m_b^{\ell^*}, K^{(\ell^*)}, 0) &= \hat{f}_{b,i}(m_b^{\ell^*}; r_{S_{b,i}}^{(\ell^*)}) \\ &= u_i^{(\ell^*)} = D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}](\bot, \bot, 0). \end{aligned}$$

Moreover, $\mathcal{B}$ makes μ key queries at most, and each of them is under different index $i \in [\mu]$. Therefore, $\mathcal{B}$ is a valid adversary for iSKFE, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. We see that $\mathcal{B}$ perfectly simulates Game $(6, \ell^* - 1)$ if $\beta = 0$. On the other hand, $\mathcal{B}$ perfectly simulates Game $(1, \ell^*)$ if $\beta = 1$. Moreover, $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ succeeds in guessing the value of b. Therefore, we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]|$, and thus $|\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \leq \epsilon$ holds. □ (**Lemma 4.6.1**)

Lemma 4.6.2 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(1,\ell^*)}] - \Pr[\mathtt{SUC}_{(2,\ell^*)}]| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus is omitted.

Lemma 4.6.3 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{RE}}$.

Proof of Lemma 4.6.3. In both of Game $(2, \ell^*)$ and Game $(3, \ell^*)$, $\left\{u_i^{(\ell^*)}\right\}_{i\in[\mu]}$ is generated using truly random strings $\left\{r_j^{(\ell^*)}\right\}_{j\in[\rho]}$. In addition, if $\left\{u_i^{(\ell^*)}\right\}_{i\in[\mu]}$ is given, the actual value of $\left\{r_j^{(\ell^*)}\right\}_{j\in[\rho]}$ is not needed for simulating both games. The only difference between two games is that $\left\{u_i^{(\ell^*)}\right\}_{i\in[\mu]}$ is computed as a real encoding of $(f_b, m_b^{\ell^*})$ in Game $(2, \ell^*)$ whereas it is computed as a simulated encoding in Game $(3, \ell^*)$. Therefore, from the security of RE, we have $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{RE}}$. □ (**Lemma 4.6.3**)

Lemma 4.6.4 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(3,\ell^*)}] - \Pr[\mathtt{SUC}_{(4,\ell^*)}]| \leq \epsilon_{\mathsf{RE}}$.

The proof is almost the same as that of Lemma 4.6.3 thus is omitted.

Lemma 4.6.5 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(4,\ell^*)}] - \mathtt{SUC}_{(5,\ell^*)}| \leq \epsilon_{\mathsf{F}}$.

The proof is straightforward thus is omitted.

Lemma 4.6.6 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(5,\ell^*)}] - \Pr[\mathtt{SUC}_{(6,\ell^*)}]| \leq \epsilon$.

The proof is almost the same as that of Lemma 4.6.1 thus is omitted.

Lemma 4.6.7 *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7|] \leq \epsilon$.

Proof of Lemma 4.6.7. The only difference between Game $(6,p)$ and 7 is how sk_{f_i} is generated for every $i \in [q]$. In Game $(6,p)$, sk_{f_i} is generated as $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot], i)$. On the other hand, in Game 7, it is generated as $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\bot, \hat{f}_{1,i}, \bot, S_{1,i}, t, \bot], i)$. Here, in both games, for every $\ell \in [p]$, $\mathsf{CT}^{(\ell)}$ is generated as $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (m_1^\ell, K^{(\ell)}, 1))$. Then, for every $i \in [q]$ and $\ell \in [p]$, we have

$$D_{\mathsf{re}}[\hat{f}_{b,i}, \hat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot](m_1^\ell, K^{(\ell)}, 1) = D_{\mathsf{re}}[\bot, \hat{f}_{1,i}, \bot, S_{1,i}, t, \bot](m_1^\ell, K^{(\ell)}, 1).$$

This is because $\hat{f}_{b,i}$ and $S_{b,i}$ are ignored in the left hand side. Therefore, we can construct an adversary attacking iSKFE whose advantage is $|\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7]|$, and thus $|\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7]| \leq \epsilon$ holds. □ (**Lemma 4.6.7**)

From inequality 4.27 and lemmas 4.6.1 to 4.6.7, inequality 4.26 holds. This completes the proof. □ (**Theorem 4.6.3**)

Note that the existence of δ-secure iSKFE scheme implies that of δ-secure decomposable randomized encoding and PRF since they are constructed from δ-secure one-way functions. Thus, from Theorem 4.6.1 and 4.6.3, we obtain Theorem 4.6.2. We again stress that this result incurs only polynomial security loss. □ (**Theorem 4.6.2**)

4.6.2 From Weakly-Selective Secure to Selective Secure

We can transform a weakly-selective-message message private SKFE scheme into a selective-message function private one.

Theorem 4.6.4 *If there exists a $(1,\delta)$-weakly-selective-message 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 weakly-selective-message message private SKFE into selective-message message private one.

Theorem 4.6.5 ([KNT18]) *If there exists a $(1,\delta)$-weakly-selective-message 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 4.6.5 by the result of Kitagawa et al. [KNT18]. Their construction does not directly use underlying SKFE and first transform it into SXIO. SXIO that is sufficient for their construction can be based on single-key weakly-succinct SKFE [BNPW16]. In the construction of SXIO, we can observe that it is sufficient that the

underlying SKFE satisfies weakly-selective-message message privacy though this fact is not explicitly stated. Thus, we obtain Theorem 4.6.5.

In addition, by Theorem 2.3.1 shown by Brakerski and Segev [BS15], 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 weakly-selective-message 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.

[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.

[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.

[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 14*, pages 646–658. ACM Press, November 2014.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.

[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.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. http://eprint.iacr.org/2015/730.

[AS15] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 191–209. IEEE Computer Society Press, October 2015.

[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.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[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.

[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.

[BGL+15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 439–448. ACM Press, June 2015.

[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.

[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 12*, pages 784–796. ACM Press, October 2012.

[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, *EURO-*

CRYPT 2016, Part II, volume 9666 of *LNCS*, pages 852–880. Springer, Heidelberg, May 2016.

[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.

[BNPW16] Nir Bitansky, Ryo Nishimaki, Alain Passelègue, and Daniel Wichs. From cryptomania to obfustopia through secret-key functional encryption. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 391–418. Springer, Heidelberg, October / November 2016.

[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.

[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.

[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.

[BS15] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 306–324. Springer, Heidelberg, March 2015.

[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.

[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.

[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.

[CGH+15] Jean-Sébastien Coron, Craig Gentry, Shai Halevi, Tancrède 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.

[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.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 429–437. ACM Press, June 2015.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016.

[CLLT17] Jean-Sébastien Coron, Moon Sung Lee, Tancrède 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.

[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.

[FRS16] Rex Fernando, Peter M. R. Rasmussen, and Amit Sahai. Preventing CLT attacks on obfuscation with linear overhead. Cryptology ePrint Archive, Report 2016/1070, 2016. http://eprint.iacr.org/2016/1070.

[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.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[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.

[GPSZ17] Sanjam Garg, Omkant Pandey, Akshayaram Srinivasan, and Mark Zhandry. Breaking the sub-exponential barrier in obfustopia. 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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[KNT18] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple and generic constructions of succinct functional encryption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 187–217. Springer, Heidelberg, March 2018.

[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 13*, pages 669–684. ACM Press, November 2013.

[KS17] Ilan Komargodski and Gil Segev. From minicrypt to obfustopia via private-key functional encryption. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 122–151. Springer, Heidelberg, April / May 2017.

[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.

[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.

[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.

[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.

[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.

[LR88] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM J. Comput.*, 17(2):373–386, 1988.

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017.

[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.

[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.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556.

[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.

[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 10*, pages 463–472. ACM Press, October 2010.

[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.

[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.

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). In Ashok K. Chandra, Detlef Wotschke, Emily P. Friedman, and Michael A. Harrison, editors, *8th ACM STOC*, pages 196–203. ACM, 1976.

[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.

[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.