# From Single-Key to Collusion-Resistant Secret-Key Functional Encryption by Leveraging Succinctness

Fuyuki Kitagawa[*][1] Ryo Nishimaki [2] Keisuke Tanaka [1]

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

**Abstract**

We show how to construct secret-key functional encryption (SKFE) supporting unbounded polynomially many functional decryption keys, that is, collusion-resistant SKFE solely from SKFE supporting only one functional decryption key. The underlying single-key SKFE needs to be weakly succinct, that is, the size of its encryption circuit is sub-linear in the size of functions.

We show we can transform any quasi-polynomially secure single-key weakly-succinct SKFE into quasi-polynomially secure collusion-resistant one. In addition, if the underlying single-key SKFE is sub-exponentially secure, then so does the resulting scheme in our construction.

Some recent results show the power and usefulness of collusion-resistant SKFE. From our result, we see that succinct SKFE is also a powerful and useful primitive. In particular, by combining our result and the result by Kitagawa, Nishimaki, and Tanaka (ePrint 2017), we can obtain indistinguishability obfuscation from sub-exponentially secure weakly succinct SKFE that supports only a single functional decryption key.

**Keywords:** Secret-key functional encryption, Collusion-resistance, Succinctness, Obfuscation.

[*]This work was done while the author was visiting NTT Secure Platform Laboratories as a summer internship student.

## Contents

# 1 Introduction

## 1.1 Background

**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 $\mathsf{MSK}$ can generate a functional decryption key $sk_f$ for a function $f$ belonging to a function family $\mathcal{F}$. By decrypting a ciphertext $ct_x$ of a message $x$ using $sk_f$, a holder of $sk_f$ can learn only a value $f(x)$. No information about $x$ except $f(x)$ is revealed from $ct_x$.

Due to the ability to generate functional decryption keys, functional encryption enables us to construct a cryptographic system with fine-grained access control. Various applications of functional encryption have been considered until today. It is known that not only public-key functional encryption (PKFE) but also secret-key functional encryption (SKFE) is useful in many application settings such as mining large datasets. In order to use functional encryption in practical situations, we need functional encryption satisfying the following two important notions, that is, *collusion-resistance* and *succinctness*.

**Collusion-resistance and succinctness.** The number of functional decryption keys that can be released is an important measure of secure functional encryption. If a functional encryption scheme can securely release only a limited number of functional decryption keys, then systems based on the functional encryption scheme are not flexible enough. A functional encryption scheme having such a limitation is called *bounded collusion-resistant*, or $q$-key scheme if the number of issuable key $q$ is specified. In particular, a scheme supporting only one functional decryption key is called a single-key scheme. Obviously, it is preferable that a functional encryption scheme does not have such a limitation and can securely release unbounded polynomially many functional decryption keys. Such functional encryption is called *collusion-resistant*.

The running time of the encryption algorithm is also an important measure of functional encryption. In many constructions proposed so far, the running time of the encryption algorithm depends on not only the length of messages to be encrypted but also the size of functions supported by the scheme. This dependence on the size of functions is undesirable since it precludes many applications of functional encryption such as delegation of computation. In the setting of delegation, the time for encrypting data should be less than that for computing functions on the data. Namely, the dependence on the size of functions should be as low as possible to decrease the encryption time of functional encryption. Functional encryption is called *succinct* if the dependence is logarithmic, and is called *weakly-succinct* if the dependence is sub-linear.

**Relation between two properties.** It seems to be difficult to construct functional encryption satisfying either one of collusion-resistance or succinctness under standard assumptions in both the secret-key and public-key settings.[1] All existing collusion-resistant or succinct schemes are based on strong assumptions such as indistinguishability obfuscation (IO), cryptographic multi-linear maps [GGH$^+$13, Wat15, GGHZ16]. Although many cryptographers have been trying to achieve collusion-resistant or succinct functional encryption under standard assumptions, nobody succeeds until today.

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

Such a transformation is already known for PKFE. Ananth, Jain, and Sahai [AJS15] showed how to construct collusion-resistant and succinct PKFE from collusion-resistant one. In addition, Garg and

[1] On the other hand, It is known that bounded collusion-resistant and non-succinct functional encryption can be realized under standard cryptographic assumptions such as one-way function or public-key encryption [GVW12].

Srinivasan [GS16] and Li and Micciancio [LM16] showed a transformation from single-key weakly-succinct PKFE to collusion-resistant one with polynomial security loss.[2] The resulting scheme of the transformation proposed by Garg and Srinivasan is succinct even if the building block scheme is only weakly-succinct. The transformation proposed by Li and Micciancio preserves succinctness of the building block scheme. From these results, collusion-resistance and succinctness are equivalent in PKFE.

On the other hand, the situation is different in SKFE. While we know how to construct collusion-resistant and succinct schemes from collusion-resistant 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*.

SKFE is useful enough to construct cryptographic systems with fine-grained access control though it is weaker than PKFE. Moreover, it is non-trivial whether techniques in the public-key setting can be applied in the secret-key setting since PKFE is stronger than SKFE. Thus, the major open question is:

*Is it possible to transform single-key weakly-succinct SKFE into collusion-resistant and succinct SKFE?*

In fact, the above question is partially solved. If we additionally assume the learning with errors (LWE) assumption or the existence of identity-based encryption, we can transform single-key weakly-succinct SKFE into collusion-resistant one by combining some previous results [LPST16, BNPW16, GS16, LM16, KNT17b]. However, the transformation is done through PKFE, and thus it seems to involve more overhead than necessary. It is important to clarify whether we can transformation SKFE more directly and efficiently.

In addition, solving the above question without assuming additional public-key primitives is having a major impact on the study of the complexity of SKFE.

**Complexity of SKFE.** Asharov and Segev [AS15] showed that SKFE does not imply plain public-key encryption via black-box reductions. This separation result gave us the impression that SKFE might be essentially equivalent to one-way functions, that is, a Minicrypt primitive [Imp95]. However, some results have recently shown that this is not the case if SKFE is used in a non-black-box manner.

Bitansky, Nishimaki, Passelègue, and Wichs [BNPW16] showed that the combination of sub-exponentially secure collusion-resistant SKFE and (almost) 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. They also 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. 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. Recently, Kitagawa, Nishimaki, and Tanaka [KNT17a] subsequently showed that IO for all polynomial size circuits can be constructed from sub-exponentially secure collusion-resistant SKFE for all circuits.

These results show that SKFE is a strong cryptographic primitive beyond Minicrypt if we consider non-black-box reductions. However, one natural question arises for this situation. All of the above results assume collusion-resistant SKFE as a building block. Thus, while we see that collusion-resistant SKFE is outside Minicrypt, it is still open whether succinct SKFE is also a strong cryptographic primitive beyond Minicrypt.

[2] Before their results, it was known that a single-key weakly succinct PKFE scheme implies a collusion-resistant and succinct one via IO [GGH$^+$13, Wat15, BV15] though it incurs sub-exponential security loss.

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

## 1.2 Our Results

Based on the above backgrounds, in this work, we investigate the relationship between the succinctness and the number of functional decryption keys of SKFE. More specifically, we show the following result.

**Theorem 1.1 (Informal).** *Assume that there exists quasi-polynomially (resp. sub-exponentially) secure single-key weakly-succinct SKFE for all circuits. Then, there also exists quasi-polynomially (resp. sub-exponentially) secure collusion-resistant SKFE for all circuits.*

We note that our transformation incurs quasi-polynomial security loss. However, we can transform any quasi-polynomially secure single-key weakly-succinct SKFE into quasi-polynomially secure collusion-resistant one, if we know the security bound of the underlying single-key SKFE. Moreover, 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 1.1, 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.

As stated above, previous results [LPST16, BNPW16, GS16, LM16, KNT17b] show that if we additionally assume the LWE assumption or identity-based encryption, both of which imply public-key encryption, we can transform succinct SKFE into collusion-resistant one. We note that our transformation is more direct and with less assumptions than that consisting of previous results.

In order to perform the transformation using previous results, we first transform succinct SKFE into succinct PKFE by assuming the LWE assumption or identity-based encryption [LPST16, BNPW16, KNT17b]. Then, we transform the succinct PKFE into collusion-resistant one [GS16, LM16].

Our transformation from single-key weakly-succinct SKFE to collusion-resistant one is direct similarly to the transformation for PKFE. In other words, our transformation avoids a path via intermediate PKFE. Thus, the transformation relying on the LWE assumption or identity-based encryption incurs more blow-up due to the transformation from SKFE into PKFE than ours.

**Additional feature of our transformation.** While the above our main result incurs quasi-polynomial security loss, our transformation technique 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 the upgrade of succinctness by the straightforward combination of Theorem 1.1 and the result of Ananth *et al.* [AJS15] incurs quasi-polynomial security loss.

Moreover, we show how to transform *weakly*-selective-secure[3] SKFE that is weakly-succinct into a selectively-secure one preserving weak-succinctness property by using some existing results [BNPW16,

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

KNT17b].

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 main transformation. Namely, by applying these upgrades before our main 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.

Note that, in the PKFE setting, such additional features are obtained in the transformation by Garg and Srinivasan [GS16], but those are not in the transformation by Li and Micciancio [LM16].

**Application to IO constructions.** Our result has applications to constructions of IO. In our result, if the underlying single-key scheme is sub-exponentially secure, then so does the resulting collusion-resistant one.[4] Therefore, by combining Theorem 1.1 and the result by Kitagawa *et al.* [KNT17a], we obtain the following corollary stating that single-key weakly-succinct SKFE itself is powerful enough to yield IO.

**Corollary 1.2 (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 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 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.

## 1.3 Technical Overview

In this section, we give a high-level overview of our technique for increasing the number of functional decryption keys that an SKFE scheme supports. The basic idea behind our proposed construction is that we combine multiple instances of a functional encryption scheme and use functional decryption keys tied to a function that outputs a re-encrypted ciphertext under a different encryption key. Several re-encryption techniques have been studied in the context of functional encryption [AJ15, BV15, BKS16, GS16, LM16], but we cannot directly use such techniques as we see below.

**First attempt: Applying re-encryption techniques in the public-key setting.** It is natural to try using the techniques in the public-key setting. In particular, it was shown that single-key weakly succinct PKFE implies collusion-resistant PKFE by Garg and Srinivasan [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.

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

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

[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.

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

By applying the PRODUCT construction $k$ times iteratively, we can construct a $\lambda^k$-key PKFE scheme from a $\lambda$-key PKFE scheme. Note that we can in turn construct a $\lambda$-key PKFE scheme by simply running $\lambda$ instances of a single-key PKFE scheme in parallel. Moreover, if the underlying single-key scheme is weakly succinct, the running time of the $\lambda^k$-key scheme depends only on $k$ instead of $\lambda^k$. 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 why the building block single-key PKFE scheme must satisfy (weak) succinctness property.

We now look closer at the technique of Li and Micciancio to see difficulties in the SKFE setting. Let $\mathsf{PKFE}$ 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.

**$(2 \times 2)$-key scheme from $2$-key scheme.**

**Setup:** Generates $\mathsf{PKFE}$ key pairs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $(\mathsf{MPK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{Setup}(1^\lambda)$ for $i \in [2]$. $\mathsf{MPK}$ 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.[5]

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

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

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

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

```
Hard-Coded Constants: $\mathsf{MPK}_i$.          // Description of (simplified) $\mathcal{E}$
Input: $m$
    1. Return $\mathsf{ct}_{\mathsf{post}} \leftarrow \mathsf{Enc}(\mathsf{MPK}_i, m)$.
```

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

Using the master secret-key $\mathsf{MSK}_1$, we can generate two functional keys $\mathsf{sk}_{f_1}^{1,1}, \mathsf{sk}_{f_2}^{1,2}$ since $\mathsf{PKFE}$ is a 2-key scheme. Similarly, we can generate two functional keys using $\mathsf{MSK}_2$. Moreover, since $\mathsf{MSK}$ 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 $\mathsf{MSK}$ at the functional key generation step. By these combinations, we can generate $2 \times 2$ 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 $\lambda$-key scheme $k$ times iteratively, we can obtain a $\lambda^k$-key scheme. Note again that we can construct a $\lambda$-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 $\mathsf{sk}_f$ might leak information about $f$. In the PKFE setting, this issue does not arise since an encryption key is publicly available.

**Second attempt: Applying techniques in a different context of SKFE.** To solve the above issue, we try using a technique in the secret-key setting but in a different context from the collusion-resistance.

Brakerski, Komargodski, and Segev [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 $\mathsf{MSK}^*$ from the re-encryption circuit $\mathcal{E}$ when we reduce the security of the resulting scheme to that of the underlying scheme corresponding to $\mathsf{MSK}^*$.

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.

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

A hybrid encryption methodology is used in many encryption schemes. In particular, Ananth, Brakerski, Segev, and Vaikuntanathan [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 encryption to *reduce the encryption time* of a functional encryption scheme without any additional assumption. Our key tool is a single-ciphertext collusion-resistant SKFE scheme called $\mathsf{1CT}$, 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.[6] 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 proposed by Ananth *et al.* [ABSV15].[7] We use $\mathsf{1CT}$ as data encapsulation mechanism and a $q$-key weakly succinct SKFE scheme $\mathsf{SKFE}$ as key encapsulation mechanism. In our hybrid construction, the encryption algorithm of $\mathsf{SKFE}$ encrypts only short values (concretely, a one-time master secret-key of $\mathsf{1CT}$), which are independent of the length of a message to be encrypted. A one-time encryption key (short and fixed length) of $\mathsf{1CT}$ is encrypted by $\mathsf{SKFE}$.

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 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 $\mathsf{SKFE}$ for a function $G$, which generates a functional key of $\mathsf{1CT}$ for $f$. A simplified description of $G$ is below. Due to the third property of $\mathsf{1CT}$, the hybrid construction preserves weak

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

succinctness.

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

[7]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.

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 $\mathsf{SKFE}$ and $\mathsf{1CT}$. Therefore, since $\mathsf{1CT}$ is collusion-resistant, if $\mathsf{SKFE}$ supports $q$ functional keys, then so does the resulting scheme, where $q$ is any fixed polynomial of $\lambda$.

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.[8]

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

Figure 1 illustrates how to construct our building blocks. An illustration of our sandwiched size-shifting procedure is described in Figure 2.

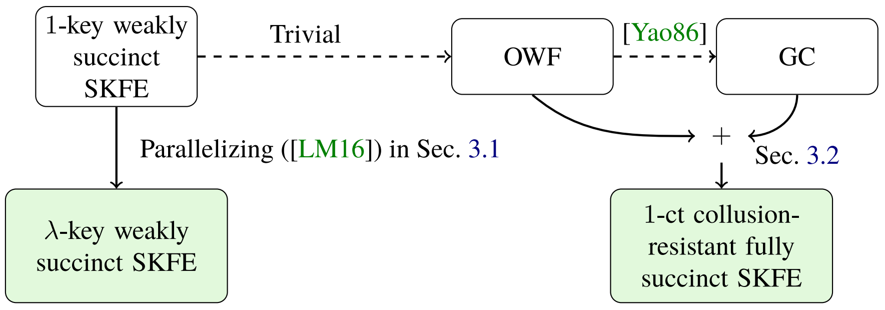
1-key weakly succinct SKFE
Trivial
OWF
[Yao86]
GC
Parallelizing ([LM16]) in Sec. 3.1
+
Sec. 3.2
λ-key weakly succinct SKFE
1-ct collusion-resistant fully succinct SKFE

**Figure 1:** Our building blocks. Green boxes denote our core schemes used in our iterated conctruction in Figure 2.

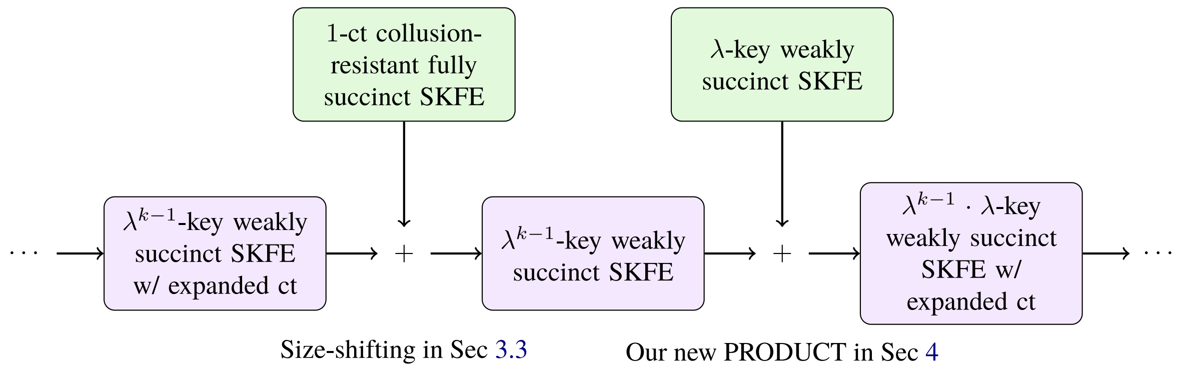
1-ct collusion-resistant fully succinct SKFE
λ-key weakly succinct SKFE
· · ·
λ^{k−1}-key weakly succinct SKFE w/ expanded ct
+
λ^{k−1}-key weakly succinct SKFE
+
λ^{k−1} · λ-key weakly succinct SKFE w/ expanded ct
· · ·
Size-shifting in Sec 3.3
Our new PRODUCT in Sec 4

**Figure 2:** An illustration of our iteration technique, in which our size-shifting procedure is sandwiched. For $k$-th iteration, first, we apply the size-shifting procedure to a $\lambda^{k-1}$-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.

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

### 1.4 Organization

In Section 2, we introduce some notions and review the definitions of cryptographic primitives that we use in this paper. Next, in Section 3, we introduce some basic constructions including the construction of 1CT and our hybrid encryption construction for size-shifting. In Section 4, we show our new PRODUCT construction. Then, in Section 5, we show how to transform single-key SKFE into collusion resistant one, that is, prove Theorem 1.1. In Section 6, we show the additional feature of our transformation.

## 2 Preliminaries

We define some notations and cryptographic primitives.

### 2.1 Notations

$x \xleftarrow{\mathsf{r}} X$ denotes choosing an element from a finite set $X$ uniformly at random, and $y \leftarrow \mathsf{A}(x; r)$ denotes assigning $y$ to the output of an algorithm $\mathsf{A}$ on an input $x$ and a randomness $r$. When there is no need to clearly write the randomness clearly, we omit it and simply write $y \leftarrow \mathsf{A}(x)$. For strings $x$ and $y$, $x \| y$ denotes the concatenation of $x$ and $y$. $\lambda$ denotes a security parameter. 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 $f(\lambda)$ being a negligible function. poly denotes an unspecified polynomial. PPT stands for probabilistic polynomial time. $[\ell]$ denotes the set of integers $\{1, \cdots, \ell\}$.

### 2.2 Pseudorandom Functions

**Definition 2.1 (Pseudorandom functions).** *For sets $\mathcal{D}$ and $\mathcal{R}$, let $\{\mathsf{F}_S(\cdot) : \mathcal{D} \rightarrow \mathcal{R} | S \in \{0,1\}^\lambda\}$ be a family of polynomially computable functions. We say that $\mathcal{F}$ is pseudorandom if for any PPT adversary $\mathcal{A}$, it holds that*

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

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

### 2.3 Secret Key Encryption

**Definition 2.2 (Secret key encryption).** *A SKE scheme* $\mathsf{SKE}$ *is a two tuple* $(\mathsf{E}, \mathsf{D})$ *of PPT algorithms.*

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

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

**Security** *Let* $\mathsf{SKE}$ *be an SKE scheme whose message space is $\mathcal{M}$. We define the security game between a challenger and an adversary $\mathcal{A}$ as follows. Below, let $n$ be a fixed polynomial of $\lambda$.*

**Initialization** *First the challenger selects a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. Next the challenger generates $n$ keys $K_j \xleftarrow{\mathsf{r}} \{0,1\}^\lambda$ for every $j \in [n]$ and sends $1^\lambda$ to $\mathcal{A}$.*

*$\mathcal{A}$ may make polynomially many encryption queries adaptively.*

**Encryption query** *$\mathcal{A}$ sends $(j, m_0, m_1) \in [n] \times \mathcal{M} \times \mathcal{M}$ to the challenger. Then, the challenger returns $c \leftarrow \mathsf{E}(K_j, m_b)$.*

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

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

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

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

## 2.4 Garbled Circuits

**Definition 2.3 (Garbled circuits).** *Let $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ be a family of circuits where each circuit in $\mathcal{C}_n$ takes $n$ bit inputs. A circuit garbling scheme $\mathsf{GC}$ is a two tuple $(\mathsf{Grbl}, \mathsf{Eval})$ of PPT algorithms.*

- *The garbling algorithm $\mathsf{Grbl}$, given a security parameter $1^\lambda$ and a circuit $C \in \mathcal{C}_n$, outputs a garbled circuit $\widetilde{C}$ together with $2n$ wire keys $\{w_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}$.*
- *The evaluation algorithm, given a garbled circuit $\widetilde{C}$ and $n$ wire keys $\{w_i\}_{i \in [n]}$, outputs $y$.*

**Correctness** *We require $\mathsf{Eval}(\widetilde{C}, \{w_{i,x_i}\}_{i \in [n]}) = C(m)$ for every $n \in \mathbb{N}$, $x \in \{0,1\}^n$, where $(\widetilde{C}, \{w_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}})$ $\leftarrow \mathsf{Grbl}(1^\lambda, C)$.*

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

**Initialization** *First, the challenger chooses a bit $b \xleftarrow{\mathsf{r}} \{0,1\}$ and sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ for the challenger. If $b = 0$, the challenger computes $(\widetilde{C}, \{w_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C)$ and returns $(\widetilde{C}, \{w_{i,x_i}\}_{i \in [n]})$ to $\mathcal{A}$. Otherwise, the challenger returns $(\widetilde{C}, \{w_i\}_{i \in [n]}) \leftarrow \mathsf{Sim}(1^\lambda, |C|, C(x))$.*

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

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

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

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

## 2.5 Decomposable Randomized Encoding

**Definition 2.4 (Decomposable randomized encoding).** *Let $c \geq 1$ be an integer constant. A $c$-local decomposable randomized encoding scheme $\mathsf{RE}$ for a function $f : \{0,1\}^n \rightarrow \{0,1\}^m$ consists of two polynomial-time algorithms $(\mathsf{RE.E}, \mathsf{RE.D})$.*

$\mathsf{RE.E}(1^\lambda, f, x)$ *takes as inputs the security parameter $1^\lambda$, a function $f$, and an input $x$ for $f$, chooses randomness $r$, and outputs an encoding $\widehat{f}(x; r)$ where $\widehat{f} : \{0,1\}^n \times \{0,1\}^\rho \rightarrow \{0,1\}^\mu$.*

$\mathsf{RE.D}(\widehat{f}(x; r))$ *takes as inputs an encoding $\widehat{f}(x; r)$ and outputs $f(x)$.*

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

**Correctness** *We require* $\Pr[f(x) = \mathsf{RE.D}(\mathsf{RE.E}(1^\lambda, f, x))] = 1$ *for any $f$ and $x$.*

**Decomposability** *Computation of $\widehat{f}$ can be decomposed into computation of $\mu$ functions. That is, $\widehat{f}(x; r) = (\widehat{f}_1(x; r), \cdots, \widehat{f}_\mu(x; r))$, where each $\widehat{f}_i$ depends on a single bit of $x$ at most and $c$ bits of $r$. We will write $\widehat{f}(x; r) = (\widehat{f}_1(x; r_{S_1}), \cdots, \widehat{f}_\mu(x; r_{S_\mu}))$, where $S_i$ denotes the subset of bits of $r$ that $\widehat{f}_i$ depends on. Parameters $\rho$ and $\mu$ are bounded by $s_f \cdot \mathrm{poly}(\lambda, n)$.*

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

**Initialization** *First, the challenger chooses a bit $b \xleftarrow{r} \{0, 1\}$ and sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends a function $f$ and an input $x \in \{0, 1\}^n$ for the challenger. If $b = 0$, the challenger computes $\left\{\widehat{f}_i(x; r)\right\}_{i=1}^{\mu} \leftarrow \mathsf{RE.E}(1^\lambda, f, x)$ and returns them to $\mathcal{A}$. Otherwise, the challenger returns $\left\{\widehat{f}_i(x; r)\right\}_{i=1}^{\mu} \leftarrow \mathsf{RE.Sim}(1^\lambda, |f|, f(x))$.*

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

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

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

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

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

**Theorem 2.5 ([Yao86, AIK06]).** *If there exist one-way functions, then there exists secure decomposable randomized encoding for all polynomial size functions.*

## 2.6 Secret-Key Functional Encryption

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

**Definition 2.6 (Secret-key functional encryption).** *An SKFE scheme* $\mathsf{SKFE}$ *is a four tuple* $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms. Below, let $\mathcal{M}$ and $\mathcal{F}$ be the message space and function space of* $\mathsf{SKFE}$*, respectively.*

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

**Definition 2.7 (Selective-message function privacy).** *Let* $\mathsf{SKFE}$ *be an SKFE scheme whose message space and function space are* $\mathcal{M}$ *and* $\mathcal{F}$*, respectively. Let* $q$ *be a fixed polynomial of* $\lambda$*. We define the selective-message function privacy game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

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

**key queries** *For a key query* $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ *from* $\mathcal{A}$*, the challenger generates* $sk_f \leftarrow \mathsf{KG}(\mathsf{MSK}, f_b)$*, 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* $\ell \in [p]$*.*

**Final phase** $\mathcal{A}$ *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* $\mathsf{SKFE}$ *is* $(q, \epsilon)$*-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 $(\mathrm{poly}, \epsilon)$-selective-message function private if it is $(q, \epsilon)$-selective-message function private for any polynomial $q$. Note that a $(\mathrm{poly}, \epsilon)$-selective-message function private SKFE scheme is also said to be $\epsilon$-secure collusion-resistant.[9]

**From message privacy to function privacy.** If $\mathcal{A}$ is allowed to send only a function $f$ (not a pair of functions) that satisfies $f(m_0^\ell) = f(m_1^\ell)$ for all $\ell$ at each key query in Definition 2.7, then we call the securtiy selective-message message privacy. A transformation from message private to function private SKFE is known.

**Theorem 2.8 ([BS15]).** *If there exists a* $(q, \delta)$*-selective-message message private SKFE scheme where* $q$ *is any fixed polynomial, there exists a* $(q, \delta)$*-selective-message function private SKFE scheme.*

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

**Definition 2.9 (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* $\mathsf{SKFE}$ *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,* $\mathsf{SKFE}$ *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* $\mathcal{A}$*.*

[9] Collusion-resistance generally does not require function privacy. Not only function private schemes but also message private schemes are referred to as collusion-resistant if they can securely issue a-priori unbounded polynomial number of functional keys.

## 2.7 Index Based Secret-Key Functional Encryption

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

In fact, for a single-key scheme, an iSKFE scheme is also an 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 2.12 for more details.

**Definition 2.10 (Index based secret-key functional encryption).** *An iSKFE scheme* $\mathsf{iSKFE}$ *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* $\mathsf{iSKFE}$, *respectively.*

- *The setup algorithm* $\mathsf{Setup}$, *given a security parameter* $1^\lambda$, *outputs a master secret key* $\mathsf{MSK}$.
- *The index based key generation algorithm* $\mathsf{iKG}$, *given a master secret key* $\mathsf{MSK}$, *a function* $f \in \mathcal{F}$, *and an index* $i \in \mathcal{I}$, *outputs a functional decryption key* $sk_f$.
- *The encryption algorithm* $\mathsf{Enc}$, *given a master secret key* $\mathsf{MSK}$ *and a message* $m \in \mathcal{M}$, *outputs a ciphertext* $\mathsf{CT}$.
- *The decryption algorithm* $\mathsf{Dec}$, *given a functional decryption key* $sk_f$ *and a ciphertext* $\mathsf{CT}$, *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. Selective-message security is known to be sufficient for constructing IO from functional encryption. Function privacy is a stronger security notion compared to the most basic security notion called message privacy. However, as shown by Brakerski and Segev [BS15], we can transform any message private SKFE to function private one without using any additional assumption. In addition, the security loss of the transformation is only constant. Therefore, in this paper, we start the transformation from selective-message function private single-key SKFE, and use selective-message function privacy as a standard security notion for SKFE and iSKFE.

**Definition 2.11 (Selective-message function privacy).** *Let* $\mathsf{iSKFE}$ *be an iSKFE scheme whose message space, function space, and index space are* $\mathcal{M}$, $\mathcal{F}$, *and* $\mathcal{I}$, *respectively. We let* $|\mathcal{I}| = q$. *We define the selective-message function privacy game between a challenger and an adversary* $\mathcal{A}$ *as follows.*

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

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

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

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

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

*For a negligible function $\epsilon(\cdot)$, We say that* iSKFE *is $(q, \epsilon)$-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 $(\mathrm{poly}, \epsilon)$-selective-message function private if the size of its index space $q$ is super-polynomial.

*Remark* 2.12 (Transforming iSKFE into SKFE). The goal of this paper is to construct $(\mathrm{poly}, \epsilon)$-selective-message function private SKFE based only on $(1, \epsilon')$-selective-message function private SKFE for some negligible functions $\epsilon$ and $\epsilon'$. In order to accomplish this task, we first construct $(\mathrm{poly}, \delta)$-selective-message function private iSKFE for some negligible function $\delta$.

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 $(\mathrm{poly}, \frac{1}{S} + \delta)$-selective-message function private. In Section 5.4, we formally show that this transformation works.

Next, we define the succinctness for SKFE and iSKFE.

**Definition 2.13 (Succinctness).** *Let $s$ and $n$ be the maximum size and input length of functions contained in $\mathcal{F}$, respectively.*

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

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

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

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

## 3 Basic Tools for Transformation

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

## 3.1 Parallel Construction

For any $q$ which is a polynomial of $\lambda$, 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 $\lambda$. 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 $sk_f \leftarrow (i, \mathsf{1Key}.sk_f)$.

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

- 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 $\mathsf{CT} \leftarrow \{\mathsf{CT}_k\}_{k\in[q]}$.

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

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

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

**Efficiency.** 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

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

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

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

where $\gamma < 1$ is a constant and $\mathrm{poly}_{\mathsf{Para}}$ is a fixed polynomial. Note that the encryption algorithm of $\mathsf{Parallel}_\lambda$ runs that of $\mathsf{1Key}$ $\lambda$ times. Therefore, $\mathsf{Para}.t_{\mathsf{Enc}}$ is $\lambda$ times bigger than the encryption time of $\mathsf{1Key}$, but the factor $\lambda$ is absorbed in $\mathrm{poly}_{\mathsf{Para}}(\lambda, n)$, and thus we can bound $\mathsf{Para}.t^{(k)}_{\mathsf{Enc}}$ by inequality (1). At the concrete instantiation in Section 5, we set $q := \lambda$ and use this bound.

**Security.** We have the following theorem.

**Theorem 3.1.** *Let* $\mathsf{1Key}$ *be* $(1,\delta)$*-selective-message function private SKFE. Then,* $\mathsf{Parallel}_q$ *is* $(q,\delta)$*-selective-message function private iSKFE.*

**Proof of Theorem 3.1.** We assume that the advantage of any adversary attacking $\mathsf{1Key}$ is bounded by $\epsilon_{\mathsf{1Key}}$. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{Parallel}_q$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Parallel}_q,\mathcal{A}}(\lambda) \leq q \cdot \epsilon_{\mathsf{1Key}}. \tag{2}$$

This means that if $\mathsf{1Key}$ is $\delta$-secure, then so is $\mathsf{Parallel}_q$. Below, we prove the above inequality (2).

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

**Initialization** On input security parameter $1^\lambda$, $\mathcal{B}$ sends it to $\mathcal{A}$. Then, $\mathcal{B}$ chooses $k^* \xleftarrow{\mathsf{r}} [q]$ and for every $k \in [q] \setminus \{k^*\}$, generates $\mathsf{MSK}_k \leftarrow \mathsf{1Key.Setup}(1^\lambda)$. When, $\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell\in[q]}$, $\mathcal{B}$ computes as follows.

- For every $k < k^*$ and $\ell \in [p]$, $\mathcal{B}$ computes $\mathsf{CT}_k^{(\ell)} \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_k, m_0^\ell)$.
- $\mathcal{B}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell\in[p]}$ to the challenger, and obtains the answer $\{\mathsf{CT}_{k^*}^{(\ell)}\}_{\ell\in[p]}$.
- For every $k > k^*$ and $\ell \in [p]$, $\mathcal{B}$ computes $\mathsf{CT}_k^{(\ell)} \leftarrow \mathsf{1Key.Enc}(\mathsf{MSK}_k, m_1^\ell)$.

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

**Key queries** 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 $\mathsf{1Key}.sk_f^i \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f_0)$, and returns $sk_f^i \leftarrow (i, \mathsf{1Key}.sk_f^i)$ 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 $sk_f^{k^*} \leftarrow (k^*, \mathsf{1Key}.sk_f)$ to $\mathcal{A}$.
- If $i > k^*$, $\mathcal{B}$ computes $\mathsf{1Key}.sk_f^i \leftarrow \mathsf{1Key.KG}(\mathsf{MSK}_i, f_1)$, and returns $sk_f^i \leftarrow (i, \mathsf{1Key}.sk_f^i)$ to $\mathcal{A}$.

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

Let $\beta$ 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 $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Parallel}_q,\mathcal{A}}(\lambda) = |\Pr[b' = 1 | k^* = 1 \land \beta = 1] - \Pr[b' = 1 | k^* = q \land \beta = 0]|$, 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 (2) holds. $\square$ (**Theorem 3.1**)

## 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 $\lambda$ bit, regardless of the length of a message to be encrypted. The construction uses a garbling scheme, an SKE scheme, and a PRF all of which can be constructed from one-way functions. The construction can be essentially seen as a flipped variant of the construction proposed by Sahai and Seyalioglu [SS10].

Let $n = |m|$. Let $\mathsf{GC} = (\mathsf{Grbl}, \mathsf{Eval})$ 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 $\mathsf{GC}$ ,$\mathsf{SKE}$, and $\mathsf{F}$, 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 $R \leftarrow \mathsf{F}_S(0\|0)$[10].
- Compute $(\widetilde{C}, \{L_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C_f)$, 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 $sk_f \leftarrow (\widetilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}})$.

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

- 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 $\mathsf{CT} \leftarrow (x, \{K_{j,x[j]}\}_{j \in [n]})$.

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

- Parse $(\widetilde{C}, \{c_{j,\alpha}\}_{j \in [n], \alpha \in \{0,1\}}) \leftarrow sk_f$ 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 $y \leftarrow \mathsf{Eval}(\widetilde{C}, \{L_j\}_{j \in [n]})$.

**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.

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

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

$$\begin{aligned}
|\mathsf{MSK}| &= \lambda,\ \mathsf{1CT}.\ell^{\mathsf{Enc}}(\lambda, n) = (\lambda+1)n, \\
\mathsf{1CT}.t_{\mathsf{Enc}}(\lambda, n) &= n \cdot \mathrm{poly}_{\mathsf{1CT}}(\lambda, \log n), \\
\mathsf{1CT}.t_{\mathsf{KG}}(\lambda, n, s) &= t_{\mathsf{F}} + t_{\mathsf{Grbl}}(\lambda, |f|) + 2 \cdot n \cdot t_{\mathsf{E}} \\
&\le \mathrm{poly}(\lambda, \log n) + |f| \cdot \mathrm{poly}(\lambda) + n \cdot \mathrm{poly}(\lambda) \\
&\le (|f| + n) \cdot \mathrm{poly}(\lambda),
\end{aligned} \tag{3}$$

where $\mathrm{poly}_{\mathsf{1CT}}$ 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 have the following theorem.

**Theorem 3.2.** *Let* GC *be $\delta$-secure garbling scheme,* SKE *$\delta$-secure SKE, and* F *$\delta$-secure PRF. Then,* 1CT *is single-ciphertext $(\mathrm{poly}, \delta)$-selective-message function private SKFE.*

**Proof of Theorem 3.2.** We assume that the advantage of any adversary attacking GC, SKE, and F is bounded by $\epsilon_{\mathsf{GC}}$, $\epsilon_{\mathsf{SKE}}$, and $\epsilon_{\mathsf{PRF}}$, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of 1CT. Moreover, we assume that $\mathcal{A}$ makes $q$ key queries at most, where $q$ is a polynomial of $\lambda$. Then, it holds that

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{A}}(\lambda) \le 2(q \cdot \epsilon_{\mathsf{GC}} + \epsilon_{\mathsf{SKE}} + \epsilon_{\mathsf{PRF}}). \tag{4}$$

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

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

- **Initialization** First, the challenger sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends $(m_0, m_1)$ to the challenger. Next, the challenger generates $S \leftarrow \{0,1\}^\lambda$ and chooses a challenge bit $b \xleftarrow{\mathsf{r}} \{0,1\}$. 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 $\mathcal{A}$.
- **key queries** For a key query $(f_0, f_1) \in \mathcal{F} \times \mathcal{F}$ from $\mathcal{A}$, the challenger first computes $K_{j,\alpha} \leftarrow \mathsf{F}_S(j\|\alpha)$ for every $j \in [n]$ and $\alpha \in \{0,1\}$, and $R \leftarrow \mathsf{F}_S(0\|0)$. Then, the challenger computes $(\widetilde{C}, \{L_{j,\alpha}\}_{j\in[n],\alpha\in\{0,1\}}) \leftarrow \mathsf{Grbl}(1^\lambda, C_{f_b})$, 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 $\mathcal{A}$.
- **Final phase** $\mathcal{A}$ outputs $b' \in \{0,1\}$.

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

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

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

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

For $h = 0, \cdots, 4$, let $\mathtt{SUC}_i$ be the event that $\mathcal{A}$ succeeds in guessing the challenge bit, that is, $b = b'$ occurs in Game $i$. 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

$$\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{A}}(\lambda) = |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \le \sum_{h=0}^{3} |\Pr[\mathtt{SUC}_h] - \Pr[\mathtt{SUC}_{h+1}]|. \tag{5}$$

Below, we estimate that each term on the right side of inequality (5) is negligible.

**Lemma 3.3.** $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \le \epsilon_{\mathsf{PRF}}$.

The proof is straightforward thus omitted.

**Lemma 3.4.** $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \epsilon_{\mathsf{SKE}}$.

**Proof of Lemma 3.4.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{SKE}$.

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

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

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

Let $\beta$ be the challenge bit between the challenger and $\mathcal{B}$. Then, the advantage of $\mathcal{B}$ is estimated as $\mathsf{Adv}_{\mathsf{SKE},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 1] - \Pr[\beta' = 1|\beta = 0]|$. When $\beta = 0$, $\mathcal{B}$ perfectly simulates Game 0 for $\mathcal{A}$. On the other hand, when $\beta = 1$, $\mathcal{B}$ perfectly simulates Game 1 for $\mathcal{A}$. In addition, $\mathcal{B}$ outputs 1 if and only if $b = b'$ occurs. Therefore, we have $\mathsf{Adv}^{\mathsf{cpa}}_{\mathsf{SKE},n,\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]|$, and thus $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \epsilon_{\mathsf{SKE}}$ holds. $\square$ **(lemma 3.4)**

**Lemma 3.5.** $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| \le q \cdot \epsilon_{\mathsf{GC}}$.

**Proof of Lemma 3.5.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{GC}$.

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

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

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

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

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

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

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

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

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

When $k^* = 1$ and $\beta = 1$, $\mathcal{B}$ perfectly simulates Game 2 for $\mathcal{A}$. On the other hand, when $k^* = q$ and $\beta = 0$, $\mathcal{B}$ perfectly simulates Game 3 for $\mathcal{A}$. Therefore, we have $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = |\Pr[b' = 1|k^* = 1 \wedge \beta = 1] - \Pr[b' = 1|k^* = q \wedge \beta = 0]|$, and thus we obtain $|\Pr[\mathtt{SUC}_2] - \Pr[\mathtt{SUC}_3]| = q \cdot \mathsf{Adv}^{\mathsf{gc}}_{\mathsf{GC},\mathsf{Sim},\mathcal{B}}(\lambda) \leq q \cdot \epsilon_{\mathsf{GC}}$. □ (**lemma 3.5**)

**Lemma 3.6.** $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = 0$.

**Proof of Lemma 3.6.** The difference between Game 3 and 4 is how the challenger generates $x$. However, $x$ is uniformly distributed in both of games thus $|\Pr[\mathtt{SUC}_3] - \Pr[\mathtt{SUC}_4]| = 0$. □ (**lemma 3.6**)

From inequality (5) and Lemmas 3.4 to 3.6, we see that inequality (4) holds. □ (**Theorem 3.2**)

## 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 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 $\{\mathsf{F}_S : \mathcal{I} \rightarrow \mathcal{R} | S \in \{0,1\}^\lambda\}$ 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 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 3.
- Return $sk_f \leftarrow sk_G$.

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

- 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 $\mathsf{Hyb.CT} \leftarrow (\mathsf{CT}, \mathsf{1CT.CT})$.

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

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

**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 3:** Construction of a key generation circuit $G$.

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 5.

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

$$\begin{aligned} |G| &= 2|f| + |\mathsf{1CT.sk}_f| + |i| + |\mathsf{1CT.KG}| + |\mathsf{F}| \\ &\le 2|f| + (|f| + |m|) \cdot \mathrm{poly}(\lambda) + \log q + (|f| + |m|) \cdot \mathrm{poly}(\lambda) + \mathrm{poly}(\lambda, \log q) \\ &\le (|f| + |m|) \cdot \mathrm{poly}_G(\lambda, \log q) \ , \end{aligned} \tag{6}$$

where $\mathrm{poly}_G$ is some fixed polynomial. The second inequality holds due to the efficiency of 1CT in Section 3.2. Note that $\mathsf{1CT}.sk_f$ is output by 1CT.KG thus $|\mathsf{1CT}.sk_f|$ is bounded by $|\mathsf{1CT.KG}|$.

**Security.** We have the following theorem.

**Theorem 3.7.** *Let* $\mathsf{iSKFE}$ *be* $(q,\delta)$*-selective-message function private iSKFE, where* $q$ *is a fixed function of* $\lambda$*. Let* $\mathsf{1CT}$ *be single-ciphertext* $(\mathrm{poly},\delta)$*-selective-message function private SKFE. Let* $\mathsf{F}$ *be a* $\delta$*-secure PRF. Then,* $\mathsf{HYBRD}$ *is* $(q,\delta)$*-selective-message function private iSKFE.*

Note that the above theorem holds even if $q$ is not polynomial of $\lambda$. In fact, in the concrete instantiation in Section 5, we set $q$ as a super-polynomial of $\lambda$.

**Proof of Theorem 3.7.** We assume that the advantage of any adversary attacking $\mathsf{iSKFE}$, $\mathsf{1CT}$, and $\mathsf{PRF}$ is bounded by $\epsilon$, $\epsilon_{\mathsf{1CT}}$, and $\epsilon_{\mathsf{PRF}}$, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{HYBRD}$. We assume that $\mathcal{A}$ sends $p$ message pairs at most to the challenger at the initialization step. Then, it holds that

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) \le 2\left((2p+1)\cdot\epsilon + p\cdot\epsilon_{\mathsf{1CT}} + p\cdot\epsilon_{\mathsf{PRF}}\right). \tag{7}$$

This means that if all of $\mathsf{iSKFE}$, $\mathsf{1CT}$, and $\mathsf{F}$ are $\delta$-secure, then so is $\mathsf{HYBRD}$. Below, we prove this via a sequence of games.

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

- **Initialization** First, the challenger sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell\in[p]}$ 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 $\{(\mathsf{CT}^{(\ell)}, \mathsf{1CT.CT}^{(\ell)})\}_{\ell\in[p]}$ to $\mathcal{A}$.
- **Key queries** When $\mathcal{A}$ makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I}\times\mathcal{F}\times\mathcal{F}$, the challenger returns $sk_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, \perp, \perp, i], i)$ to $\mathcal{A}$.
- **Final phase** $\mathcal{A}$ output $b'$.

For every $\ell^* \in [p]$, we define the following games. 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 $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\perp, \perp, 0))$.

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

**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 $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\mathsf{1CT{\cdot}MSK}^{(\ell^*)}, S^{(\ell^*)}, 1))$. In addition, when $\mathcal{A}$ makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, the challenger responds with $sk_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, \bot, i], i)$.

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 $sk_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[\bot, f_1^i, \bot, i], i)$. In this game, the challenger generates $\mathsf{CT}^{(\ell)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\mathsf{1CT.MSK}^{(\ell)}, S^{(\ell)}, 1))$ 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 $(h,\ell^*)$. 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

$$\begin{aligned}\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\le \sum_{\ell^* \in [p]} |\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \\ &\quad + \sum_{\ell^* \in [p]} \sum_{h=1}^{4} |\Pr[\mathtt{SUC}_{(h,\ell^*)}] - \Pr[\mathtt{SUC}_{(h+1,\ell^*)}]| \\ &\quad + |\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]| \end{aligned} \tag{8}$$

Below, we estimate each term on the right side of inequality (8).

**Lemma 3.8.** *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(5,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \le \epsilon$.*

**Proof of Lemma 3.8.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{iSKFE}$.

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

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

Then, $\mathcal{B}$ sends $\{(M_0^\ell, M_1^\ell)\}_{\ell \in [p]}$ to the challenger and gets the answer $\{\mathsf{CT}^{(\ell)}\}_{\ell \in [p]}$. 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 $\ell \ge \ell^*$. Finally, $\mathcal{B}$ sets $\mathsf{Hyb.CT}^{(\ell)} \leftarrow (\mathsf{CT}^{(\ell)}, \mathsf{1CT.CT}^{(\ell)})$ for every $\ell \in [p]$, and sends $\{\mathsf{Hyb.CT}^{(\ell)}\}_{\ell \in [p]}$ to $\mathcal{A}$.

**Key queries** When $\mathcal{A}$ makes a key query $(i, f_0^i, f_1^i) \in \mathcal{I} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ first computes $u_i \leftarrow \mathsf{1CT.KG}(\mathsf{MSK}^{(\ell^*)}, f_b^i; r_i^{(\ell^*)})$, 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 $\mathcal{A}$.

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

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

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

If $\mathcal{A}$ makes only one key query under every index $i \in [q]$, then so does $\mathcal{B}$. Therefore, since $\mathcal{A}$ is a valid adversary for HYBRD, $\mathcal{B}$ is a valid adversary for iSKFE, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{iSKFE},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. $\mathcal{B}$ 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. $\square$ **(lemma 3.8)**

**Lemma 3.9.** *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(1,\ell^*)}] - \Pr[\mathtt{SUC}_{(2,\ell^*)}]| \leq \epsilon_{\mathsf{PRF}}$.

The proof is straightforward thus omitted.

**Lemma 3.10.** *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{1CT}}$.

**Proof of Lemma 3.10.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks 1CT.

**Initialization** On input security parameter $1^\lambda$, $\mathcal{B}$ sends it to $\mathcal{A}$. Then, $\mathcal{B}$ chooses $b \xleftarrow{r} \{0,1\}$. When, $\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [q]}$, $\mathcal{B}$ first computes $\{\mathsf{CT}^{(\ell)}\}_{\ell \in [p]}$ and $\{\mathsf{1CT.CT}^{(\ell)}\}_{\ell \in [p]}$ 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 $\mathsf{CT}^{(\ell^*)} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (\perp, \perp, 0))$. 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 $\mathsf{1CT.CT}^{(\ell)} \leftarrow \mathsf{1CT.Enc}(\mathsf{1CT.MSK}^{(\ell)}, m_b^\ell)$.

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

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

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

Let $\beta$ be the challenge bit between the challenger and $\mathcal{B}$. For every key query $(i, f_0^i, f_1^i)$ made by $\mathcal{A}$, $f_0^i(m_0^{\ell^*}) = f_1^i(m_0^{\ell^*})$ holds since $\mathcal{A}$ is a valid adversary for HYBRD. In addition, $\mathcal{B}$ sends only one message tuple in the initialization step. Therefore, $\mathcal{B}$ is a valid adversary for 1CT, and thus we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{1CT},\mathcal{B}}(\lambda) = |\Pr[\beta' = 1|\beta = 0] - \Pr[\beta' = 1|\beta = 1]|$. 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. $\square$ **(lemma 3.10)**

**Lemma 3.11.** *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(3,\ell^*)}] - \Pr[\mathtt{SUC}_{(4,\ell^*)}]| \leq \epsilon_{\mathsf{PRF}}$.

The proof is straightforward thus omitted.

**Lemma 3.12.** *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 3.8 thus is omitted.

**Lemma 3.13.** $|\Pr[\mathtt{SUC}_{(5,p)}] - \Pr[\mathtt{SUC}_6]| \leq \epsilon$.

**Proof of Lemma 3.13.** The only difference between Game $(5, p)$ and 6 is how $sk_G^{(i)}$ is generated for every $i \in [q]$. In Game $(5, p)$, it is generated as $sk_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[f_b^i, f_1^i, \perp, i], i)$. On the other hand, in Game 6, it is generated as $sk_G^{(i)} \leftarrow \mathsf{iKG}(\mathsf{MSK}, G[\perp, f_1^i, \perp, i], i)$. 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

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

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. $\square$ (**lemma 3.13**)

From inequality (8) and Lemmas 3.8 to 3.13, we see that inequality (7) holds. $\square$ (**Theorem 3.7**)

## 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 1.3, 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. In addition, we assume that all of randomness spaces of Lf.Setup, Lf.iKG, and 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 $r_{\mathsf{Setup}}^i \leftarrow \mathsf{F}_S(i\|0)$, $S_i \leftarrow \mathsf{F}_S(i\|1)$, and $r_{\mathsf{iKG}}^i \leftarrow \mathsf{F}_S(i\|2)$.
- Generate $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda; r_{\mathsf{Setup}}^i)$.
- Compute $\mathsf{Rt.sk}_{e_i} \leftarrow \mathsf{Rt.iKG}(\mathsf{Rt.MSK}, e[\mathsf{Lf.MSK}_i, S_i, 0], i; r_{\mathsf{iKG}}^i)$ 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 $t \xleftarrow{r} \{0,1\}^\lambda$.
- Compute $\mathsf{Rt.CT} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m, \perp, t, \perp))$.
- 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 $y \leftarrow \mathsf{Lf.Dec}(\mathsf{Lf}.sk_f^{i,j}, \mathsf{Lf.CT})$.

**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 $\mathsf{Lf.MSK}_i = \perp$, 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:** Construction of an encryption circuit $e$.

**Correctness.** Let $|\mathcal{I}_{\mathsf{Rt}}| = q_{\mathsf{Rt}}$. In the construction, we use $q_{\mathsf{Rt}}$ instances of Leaf and thus $q_{\mathsf{Rt}}$ 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 $q_{\mathsf{Rt}}$ functional keys. In order to ensure that only $q_{\mathsf{Rt}}$ 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 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.** Let $|\mathsf{Lf.Enc}|$ and $|\mathsf{F}'|$ denote the size of the circuits computing Lf.Enc and $\mathsf{F}'$, respectively. Then, the size of $e$ is

$$\begin{aligned} |e| = |\mathsf{Lf.MSK}| + \lambda + 1 + |\mathsf{Lf.Enc}| + |\mathsf{F}'| &\leq 2|\mathsf{Lf.Enc}| + \mathrm{poly}(\lambda) \\ &\leq |\mathsf{Lf.Enc}| \cdot \mathrm{poly}_e(\lambda), \end{aligned} \tag{9}$$

where $\mathrm{poly}_e$ is a fixed polynomial.

Let $\mathsf{Rt}.t_{\mathsf{Enc}}$ and $\mathsf{Rt}.t_{\mathsf{iKG}}$ be bounds of the running time of Rt.Enc and Rt.iKG. Let $\mathsf{Lf}.t_{\mathsf{Enc}}$ and $\mathsf{Lf}.t_{\mathsf{iKG}}$ be bounds of the running time of Rt.Enc and Rt.iKG. Let $\mathsf{Prd}.t_{\mathsf{Enc}}$ and $\mathsf{Prd}.t_{\mathsf{iKG}}$ be bounds of the running time of Prd.Enc and Prd.iKG. Note that the length of a ciphertext output by 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.** Let $|\mathcal{I}_{\mathsf{Rt}}| = q_{\mathsf{Rt}}$ and $|\mathcal{I}_{\mathsf{Lf}}| = q_{\mathsf{Lf}}$. Then, we have the following theorem.

**Theorem 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* $\mathsf{F}$ *and* $\mathsf{F}'$ *be* $\delta$*-secure PRF. Then,* PRDCT *is* $(q_{\mathsf{Rt}} \cdot q_{\mathsf{Lf}}, \delta)$*-selective-message function private iSKFE.*

**Proof of Theorem 4.1.** We assume that the advantage of any adversary attacking Root, Leaf, F, and F$'$ is bounded by $\epsilon_{\mathsf{Rt}}$, $\epsilon_{\mathsf{Lf}}$, $\epsilon_{\mathsf{PRF}}$, and $\epsilon_{\mathsf{PRF}}$, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective message function privacy of PRDCT. We assume that $\mathcal{A}$ makes key queries for at most $q$ different indices $\{i_k\}_{k\in[q]} \in \mathcal{I}_{\mathsf{Rt}}^q$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) \le 2\left((2q+2)\cdot\epsilon_{\mathsf{Rt}} + q\cdot\epsilon_{\mathsf{Lf}} + (2q+1)\epsilon_{\mathsf{PRF}}\right). \tag{10}$$

This means that if all of Root, Leaf, F, and F$'$ are $\delta$-secure, then so does PRDCT. Below, we prove this via a sequence of games.

Below, we prove equality (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 $\lambda$.

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

**Initialization** First, the challenger sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell\in[p]}$ 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 $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, \bot, t^{(\ell)}, \bot))$ for every $\ell \in [p]$, and returns $\{(\mathsf{Rt.CT}^{(\ell)})\}_{\ell\in[p]}$ to $\mathcal{A}$.

**Key queries** When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, 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 $\mathcal{A}$.

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

**Game 1** Same as Game 0 except how the challenger responds to a key query made by $\mathcal{A}$. At the initialization step, the challenger prepares a list $\mathcal{L}$ which stores an index $i \in \mathcal{I}$ and corresponding master secret key $\mathsf{Lf.MSK}_i$, 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 $r^i_{\mathsf{Setup}}$, $S_i$, and $r^i_{\mathsf{iKG}}$ 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 $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \bot))$ 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 that 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

$$\begin{aligned}\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\le |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| + |\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \\ &\quad + \sum_{k^* \in [q]} |\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]| \\ &\quad + \sum_{k^* \in [q]} \sum_{h=3}^{6} |\Pr[\mathtt{SUC}_{(h,k^*)}] - \Pr[\mathtt{SUC}_{(h+1,k^*)}]| \\ &\quad + |\Pr[\mathtt{SUC}_{(7,q)}] - \Pr[\mathtt{SUC}_8]| \end{aligned} \tag{11}$$

Below, we estimate each term on the right side of inequality (11).

**Lemma 4.2.** $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \le \epsilon_{\mathsf{PRF}}$.

The proof of it is straightforward thus omitted.

**Lemma 4.3.** $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \le \epsilon_{\mathsf{Rt}}$.

**Proof of Lemma 4.3.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Root.

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

**Key queries** When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ 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 $r^i_{\mathsf{Setup}}, S_i, r^i_{\mathsf{iKG}} \xleftarrow{r} \{0,1\}^\lambda$, 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 $\mathcal{L}$.

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

Let $\beta$ 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 $e[\mathsf{Lf.MSK}_i, S_i, 0](m_b^\ell, \perp, t^{(\ell)}, \perp) = e[\mathsf{Lf.MSK}_i, S_i, 0](m_b^\ell, m_1^\ell, t^{(\ell)}, \perp)$. In addition, $\mathcal{B}$ makes 1 key query under every index $i \in \mathcal{I}$ at most. Therefore, $\mathcal{B}$ is a valid adversary for 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 $b$. Therefore, we have $\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{Root},\mathcal{B}}(\lambda) = |\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]|$ thus $|\Pr[\mathtt{SUC}_1] - \Pr[\mathtt{SUC}_2]| \leq \epsilon_{\mathsf{Rt}}$ holds. □ (**lemma 4.3**)

**Lemma 4.4.** *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.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Root.

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

**Key queries** When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ 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 $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0,1\}^\lambda$. 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 $(i, e[\mathsf{Lf.MSK}_i, S_i, 0], e[\perp, \perp, 0])$ 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 $\mathsf{Lf.MSK}_i \leftarrow \mathsf{Lf.Setup}(1^\lambda)$, $S_i \xleftarrow{r} \{0,1\}^\lambda$. 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 $\mathcal{L}$.

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

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

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

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

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

for every $\ell \in [p]$. Therefore, $\mathcal{B}$ is a valid adversary for 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 $|\Pr[\mathtt{SUC}_{(7,k^*-1)}] - \Pr[\mathtt{SUC}_{(3,k^*)}]| \le \epsilon_{\mathsf{Rt}}$ holds. □ **(lemma 4.4)**

**Lemma 4.5.** *For every $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(3,k^*)}] - \Pr[\mathtt{SUC}_{(4,k^*)}]| \le \epsilon_{\mathsf{PRF}}$.*

The proof is straightforward thus omitted.

**Lemma 4.6.** *For every $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(4,k^*)}] - \Pr[\mathtt{SUC}_{(5,k^*)}]| \le \epsilon_{\mathsf{Lf}}$.*

**Proof of Lemma 4.6.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks Leaf.

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

**Key queries** When $\mathcal{A}$ makes a key query $(i, j, f_0^{i,j}, f_1^{i,j}) \in \mathcal{I}_{\mathsf{Rt}} \times \mathcal{I}_{\mathsf{Lf}} \times \mathcal{F} \times \mathcal{F}$, $\mathcal{B}$ 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 $i \ne i^*$, 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 $\mathcal{L}$.

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

Let $\beta$ be the challenge bit between the challenger and $\mathcal{B}$. Since $\mathcal{A}$ is a valid adversary for PRDCT, for every $\ell$ 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 $|\Pr[\mathtt{SUC}_{(4,k^*)}] - \Pr[\mathtt{SUC}_{(5,k^*)}]| \le \epsilon_{\mathsf{Lf}}$ holds. □ **(lemma 4.6)**

**Lemma 4.7.** *For every $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(5,k^*)}] - \Pr[\mathtt{SUC}_{(6,k^*)}]| \le \epsilon_{\mathsf{PRF}}$.*

The proof is straightforward thus omitted.

**Lemma 4.8.** *For every $k^* \in [q]$, $|\Pr[\mathtt{SUC}_{(6,k^*)}] - \Pr[\mathtt{SUC}_{(7,k^*)}]| \le \epsilon$.*

The proof is almost the same as that of Lemma 4.4 thus is omitted.

**Lemma 4.9.** $|\Pr[\mathtt{SUC}_{(7,p)}] - \Pr[\mathtt{SUC}_8]| \le \epsilon_{\mathsf{Rt}}$.

**Proof of Lemma 4.9.** 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 $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (m_b^\ell, m_1^\ell, t^{(\ell)}, \perp))$. On the other hand, in Game 8, it is generated as $\mathsf{Rt.CT}^{(\ell)} \leftarrow \mathsf{Rt.Enc}(\mathsf{Rt.MSK}, (\perp, m_1^\ell, t^{(\ell)}, \perp))$. 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

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

This is because $m_b^\ell$ 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 $|\Pr[\mathtt{SUC}_{(7,q)}] - \Pr[\mathtt{SUC}_8]| \le \epsilon$ holds. □ (**lemma 4.9**)

From inequality (11) and Lemmas 4.2 to 4.9, inequality (10) holds. □ (**Theorem 4.1**)

## 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 a collusion-resistant 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 increase the index space by using our new PRODUCT construction introduced in Section 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 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.

### 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 $\lambda$-key iSKFE scheme $\mathsf{Parallel}_\lambda$ from the underlying single-key SKFE scheme. This is done by simply running $\lambda$ instances of the single-key scheme as in Section 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 $\lambda^2$-key scheme $\mathsf{PRDCT}_2$ by combining 2 instances of $\mathsf{Parallel}_\lambda$ using our PRODUCT construction in Section 4. The encryption time of $\mathsf{PRDCT}_2$ 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 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 $\mathsf{HYBRD}_2$ by combining $\mathsf{PRDCT}_2$ 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 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 $\mathsf{PRDCT}_2$ in the hybrid construction is also $O(\lambda)$. Therefore, the encryption time of $\mathsf{HYBRD}_2$ 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 $\mathsf{HYBRD}_2$. 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 $\mathsf{PRDCT}_3$ by using $\mathsf{HYBRD}_2$ as $\mathsf{Root}$ and *a fresh instance* of $\mathsf{Parallel}_\lambda$ as $\mathsf{Leaf}$. In this case, a ciphertext of $\mathsf{Parallel}_\lambda$ is embedded into a ciphertext of $\mathsf{HYBRD}_2$ in the security proof. Therefore, the encryption time of $\mathsf{PRDCT}_3$ is roughly

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

where $m$ is a message to be encrypted by $\mathsf{PRDCT}_3$. 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* 5.1 (Iterated linear and iterated square composition). In our iterated construction, on the $k$-th application of PRODUCT construction, we use $\lambda^k$-key scheme constructed so far as $\mathsf{Root}$ and *a fresh instance* of $\mathsf{Parallel}_\lambda$ as $\mathsf{Leaf}$. 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 $\mathsf{Root}$ and $\mathsf{Leaf}$ are the resulting scheme of the previous $k-1$ compositions. In this composition, we can construct $\lambda^{2^k}$-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 $\mathsf{PRDCT}_3$ by using $\mathsf{HYBRD}_2$ as both $\mathsf{Root}$ and $\mathsf{Leaf}$ in our PRODUCT construction. In this case, a ciphertext of $\mathsf{HYBRD}_2$ is embedded into another ciphertext of $\mathsf{HYBRD}_2$ in the security proof. Thus, the encryption time of $\mathsf{PRDCT}_3$ is roughly

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

where $m$ is a message to be encrypted by $\mathsf{PRDCT}_3$. 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.

## 5.2 Construction of Collusion-Resistant iSKFE

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

$$\langle \mathsf{iSKFE}_{\mathsf{Rt}}, \mathsf{iSKFE}_{\mathsf{Lf}} \rangle_{\mathsf{product}} = \mathsf{iSKFE}$$

denote that an iSKFE scheme $\mathsf{iSKFE}$ is constructed from our new PRODUCT construction in Section 4 by using $\mathsf{iSKFE}_{\mathsf{Rt}}$ as Root and $\mathsf{iSKFE}_{\mathsf{Lf}}$ as $\mathsf{Leaf}$. Moreover, we let

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

denote that an SKFE scheme $\mathsf{SKFE}'$ is constructed from our proposed hybrid encryption construction introduced in Section 3.3 by using $\mathsf{iSKFE}$ 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 3.1 that the number of parallelization is $\lambda$. That is, we set $q = \lambda$ and use $\mathsf{1Key}$ in the construction introduced in Section 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 $\mathsf{HYBRD}_\eta$ follows from those of building block constructions.

Note that for every $k \in [\eta]$, both $\mathsf{HYBRD}_k$ and $\mathsf{PRDCT}_k$ support $\lambda^k$ functional keys. In particular, the number of functional keys supported by $\mathsf{HYBRD}_\eta$ is $\lambda^\eta$ thus is super-polynomial since $\eta = \omega(1)$.

Our collusion-resistant iSKFE is $\mathsf{HYBRD}_\eta$ where $\eta = \omega(1)$. $\mathsf{HYBRD}_\eta$ is based solely on $\mathsf{1Key}$, and the following theorem holds.

**Theorem 5.2.** *Assuming there exists $(1, \delta)$-selective-message function private SKFE that is weakly succinct, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, there exists $(\mathrm{poly}, \delta)$-selective-message function private iSKFE the size of whose index space is $\lambda^{\zeta^{1/2}}$ thus is super-polynomial in $\lambda$.*

## 5.3 Analysis of Our Collusion-Resistant iSKFE

In this section, we formally analyze the security and efficiency of $\mathsf{HYBRD}_\eta$ and prove Theorem 5.2.

**Proof of Theorem 5.2.** We start with the security bound analysis then move to the security analysis.

**Security bound analysis.** We assume that the advantages of any adversary attacking $\mathsf{Parallel}_\lambda$, $\mathsf{1CT}$, and $\mathsf{PRF}$ is bounded by $\epsilon_{\mathsf{Para}}$, $\epsilon_{\mathsf{1CT}}$, and $\epsilon_{\mathsf{PRF}}$, 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 $\mathsf{HYBRD}_k$ and $\mathsf{PRDCT}_k$, respectively. Then, from inequalities (7) and (10), for every $k \in [\eta - 1]$, we have

$$\begin{aligned}
\epsilon_{\mathsf{Hyb}_{k+1}} &\leq 2\left((2p+1)\epsilon_{\mathsf{Prd}_{k+1}} + p \cdot \epsilon_{\mathsf{1CT}} + p \cdot \epsilon_{\mathsf{PRF}}\right) \\
&\leq 4(2p+1)\left((2q+2)\epsilon_{\mathsf{Hyb}_k} + q \cdot \epsilon_{\mathsf{Para}} + (2q+1)\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}\right) \\
&\leq 8(2p+1)(q+1) \cdot \epsilon_{\mathsf{Hyb}_k} + 8(2p+1)(q+1)\left(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}\right),
\end{aligned}$$

where $p$ is a polynomial of $\lambda$ denoting the number of message pairs an adversary attacking $\mathsf{HYBRD}_\eta$ queries, and $q$ is the number of key queries made by the adversary. Then, by setting $Q := 8(2p+1)(q+1)$, we get

$$\epsilon_{\mathsf{Hyb}_{k+1}} \leq Q \cdot \epsilon_{\mathsf{Hyb}_k} + Q \cdot \left(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}\right).$$

Therefore, it holds that

$$\begin{aligned}
\epsilon_{\mathsf{Hyb}_\eta} &\leq Q^{\eta-1} \cdot \epsilon_{\mathsf{Hyb}_1} + \left(\sum_{k=0}^{\eta-2} Q^k\right) \cdot Q(\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}) \\
&\leq Q^{\eta-1} \cdot 2\left((p+1) \cdot \epsilon_{\mathsf{Para}} + p \cdot \epsilon_{\mathsf{1CT}} + p \cdot \epsilon_{\mathsf{PRF}}\right) \\
&\qquad + (\eta - 2) \cdot Q^{\eta-2} \cdot (\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}) \\
&\leq (\eta - 2) \cdot Q^{\eta-1} \cdot 3(p+1) \cdot (\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}).
\end{aligned}$$

Since $p$ and $q$ are polynomials of $\lambda$, and $Q = 8(2p+1)(q+1)$, we get

$$\epsilon_{\mathsf{Hyb}_\eta} \leq \lambda^{O(\eta)} \cdot (\epsilon_{\mathsf{Para}} + \epsilon_{\mathsf{1CT}} + \epsilon_{\mathsf{PRF}}). \tag{12}$$

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 3.1, $\mathsf{Parallel}_\lambda$ is also $\delta$-secure. If there exists a $(1, \delta)$-selective-message function private SKFE scheme, then there also exists a single-ciphertext $(\mathrm{poly}, \delta)$-selective-message function private SKFE scheme $\mathsf{1CT}$ since it can be constructed based only on $\delta$-secure one-way functions as we show in Section 3.2. In addition, we can construct a $\delta$-secure PRF from $\delta$-secure one-way functions. Then, by using such $\delta$-secure primitives as building blocks of $\mathsf{HYBRD}_\eta$ 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 $\eta$ as slightly super-constant (e.g., $\log\log\lambda$) regardless of the security bound of the building block scheme since we can ensure that $\mathsf{HYBRD}_\eta$ supports super-polynomial number of keys and is $\delta$-secure if we do so. This is not the case. $\mathsf{HYBRD}_\eta$ is an iSKFE scheme and thus we finally need to transform it into an SKFE scheme. As stated in Remark 2.12, 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 $\eta$ as slightly super-constant such as $\mathrm{loglog}\lambda$ when transforming quasi-polynomially (resp. sub-exponentially) secure single-key SKFE scheme. See Section 5.4 for more details.

This completes the security bound analysis.

**Efficiency analysis.** We show that each algorithm of $\mathsf{HYBRD}_\eta$ runs in polynomial time of $\lambda, n$ and $s$, where $s$ and $n$ are the maximum size and input length of functions supported by $\mathsf{HYBRD}_\eta$.

For every $k \in [\eta]$, we use new instances of $\mathsf{Parallel}_\lambda$ and $\mathsf{1CT}$ when constructing $\mathsf{PRDCT}_k$ and $\mathsf{HYBRD}_k$, 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 $\mathsf{PRDCT}_k$ and $\mathsf{HYBRD}_k$. 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 $\mathsf{HYBRD}_\eta$, it is sufficient to prove that each algorithm of $\mathsf{Parallel}_\lambda^{(k)}$ and $\mathsf{1CT}^{(k)}$ for every $k \in [\eta]$ used in $\mathsf{HYBRD}_\eta$ runs in polynomial time of $\lambda, n$ 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 3 and 4, respectively. Below, let $G[f]$ denote the key generation circuit $G[f, \perp, \perp, i]$, where $f$ is a function and $i$ is an index. Note that which index is hardwired does not affect the size of $G$ thus we omit writing the index. In addition, let $e[\mathsf{Enc}]$ denote the encryption circuit computing $\mathsf{Enc}$ (and one PRF evaluation). From the bounds (6) and (9) that we show in Section 3.3 and 4, we can bound the size of $G$ and $e$ as,

$$
\begin{aligned}
|G[f]| &\leq (|f| + n_f) \cdot \mathrm{poly}_G(\lambda, \log q), && (13)\\
|e[\mathsf{Enc}]| &\leq |\mathsf{Enc}| \cdot \mathrm{poly}_e(\lambda), && (14)
\end{aligned}
$$

where $\mathrm{poly}_G$ and $\mathrm{poly}_e$ 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 (1) in Section 3.1, for every $k \in [\eta]$, $\mathsf{Para}.t_{\mathsf{Enc}}^{(k)}$ is bounded as

$$
\mathsf{Para}.t_{\mathsf{Enc}}^{(k)} \leq |f|^\gamma \cdot \mathrm{poly}_{\mathsf{Para}}(\lambda, n_f), \tag{15}
$$

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 $\mathrm{poly}_{\mathsf{Para}}$ 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 $\mathrm{poly}_{\mathsf{Para}}$ 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 $G[e[\mathsf{Para.Enc}^{(k)}]]$. Therefore, from (15), we have

$$
\mathsf{Para}.t_{\mathsf{Enc}}^{(k-1)} \leq |G[e[\mathsf{Para.Enc}^{(k)}]]|^\gamma \cdot \mathrm{poly}_{\mathsf{Para}}(\lambda, 2\lambda + 1). \tag{16}
$$

Here, the input length of the circuit $e[\mathsf{Para.Enc}^{(k)}]$ is bounded by $2(2\lambda+1) + \lambda + |\mathsf{Para.Enc}^{(k)}|$ since the length of a ciphertext output by $\mathsf{Para.Enc}^{(k)}$ is bounded by the size of $\mathsf{Para.Enc}^{(k)}$. Then, from (13) and (14), for every $k \in [\eta]$, we can estimate $|G[e[\mathsf{Para.Enc}^{(k)}]]|$ as

$$
\begin{aligned}
|G[e[\mathsf{Para.Enc}^{(k)}]]| &\leq \Big(|e[\mathsf{Para.Enc}^{(k)}]| + (2(2\lambda+1) + \lambda + |\mathsf{Para.Enc}^{(k)}|)\Big) \cdot \mathrm{poly}_G(\lambda, \log \lambda^k)\\
&= \Big(|\mathsf{Para.Enc}^{(k)}| \cdot \mathrm{poly}_e(\lambda) + ((2(2\lambda+1) + \lambda + |\mathsf{Para.Enc}^{(k)}|)\Big) \cdot\\
&\qquad \mathrm{poly}_G(\lambda, \log \lambda^k)\\
&\leq |\mathsf{Para.Enc}^{(k)}| \cdot \mathrm{poly}_1(\lambda, k) \leq |\mathsf{Para.Enc}^{(k)}| \cdot \mathrm{poly}_1(\lambda, \eta), && (17)
\end{aligned}
$$

where $\mathrm{poly}_1$ is a fixed polynomial. Thus, from (16) and (17), for every $k \in \{2, \cdots, \eta\}$, we have

$$\begin{aligned} \mathsf{Para}.t_{\mathsf{Enc}}^{(k-1)} &\leq \left(|\mathsf{Para}.\mathsf{Enc}^{(k)}| \cdot \mathrm{poly}_1(\lambda, \eta)\right)^{\gamma} \cdot \mathrm{poly}_{\mathsf{Para}}(\lambda, 2\lambda+1) \\ &\leq |\mathsf{Para}.\mathsf{Enc}^{(k)}|^{\gamma} \cdot \mathrm{poly}_2(\lambda, \eta) = (\mathsf{Para}.t_{\mathsf{Enc}}^{(k)})^{\gamma} \cdot \mathrm{poly}_2(\lambda, \eta), \end{aligned} \tag{18}$$

where $\mathrm{poly}_2$ is also a fixed polynomial. In the last equality, we consider $|\mathsf{Para}.\mathsf{Enc}^{(k)}|$ 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 $\mathsf{HYBRD}_{\eta}$ is hardwired. Therefore, by using (13) and (15) again, we can bound $\mathsf{Para}.t_{\mathsf{Enc}}^{(\eta)}$ as

$$\begin{aligned} \mathsf{Para}.t_{\mathsf{Enc}}^{(\eta)} &\leq ((s+n) \cdot \mathrm{poly}_G(\lambda, \log \lambda^{\eta}))^{\gamma} \cdot \mathrm{poly}_{\mathsf{Para}}(\lambda, 2\lambda+1) \\ &\leq s^{\gamma} \cdot \mathrm{poly}_3(\lambda, n, \eta), \end{aligned} \tag{19}$$

where $\mathrm{poly}_3$ is also a fixed polynomial.

From (18) and (19), for every $k \in [\eta]$, it holds that

$$\begin{aligned} \mathsf{Para}.t_{\mathsf{Enc}}^{(k)} &\leq (s^{\gamma} \cdot \mathrm{poly}_3(\lambda, n, \eta))^{\gamma^{\eta-k}} \cdot \prod_{j=0}^{\eta-k-1} \mathrm{poly}_2(\lambda, \eta)^{\gamma^j} \\ &\leq s^{\gamma} \cdot \mathrm{poly}_3(\lambda, n, \eta) \cdot \mathrm{poly}_2(\lambda, \eta)^{\frac{1}{1-\gamma}} \leq s^{\gamma} \cdot \mathrm{poly}_4(\lambda, n, \eta), \end{aligned} \tag{20}$$

where $\mathrm{poly}_4$ 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 $\lambda, n, \eta$ and $s$.

Next, we analyze the encryption time of $1\mathsf{CT}^{(k)}$ for every $k \in [\eta]$. For every $k \in [\eta]$, let $1\mathsf{CT}.t_{\mathsf{Enc}}^{(k)}$ be the bound of the running time of the encryption algorithm of $1\mathsf{CT}^{(k)}$. For every $k \in [\eta]$, $1\mathsf{CT}^{(k)}$ generates a functional key tied to the circuit $e[\mathsf{Para}.\mathsf{Enc}^{(k)}]$. From (20), the input length of $e[\mathsf{Para}.\mathsf{Enc}^{(k)}]$ is bounded by

$$\begin{aligned} 2(2\lambda+1) + \lambda + |\mathsf{Para}.\mathsf{Enc}^{(k)}| &\leq 2(2\lambda+1) + \lambda + s^{\gamma} \cdot \mathrm{poly}_4(\lambda, n, \eta) \\ &\leq s^{\gamma} \cdot \mathrm{poly}_5(\lambda, n, \eta), \end{aligned}$$

where $\mathrm{poly}_5$ is a fixed polynomial. Then, from the quasi-linear efficiency of $1\mathsf{CT}$ that we show as (3) in Section 3.2, for every $k \in [\eta]$, we have

$$\begin{aligned} 1\mathsf{CT}.t_{\mathsf{Enc}}^{(k)} &\leq (s^{\gamma} \cdot \mathrm{poly}_5(\lambda, n, \eta)) \cdot \mathrm{poly}_{1\mathsf{CT}}(\lambda, \log(s^{\gamma} \cdot \mathrm{poly}_5(\lambda, n))) \\ &\leq s^{\gamma'} \cdot \mathrm{poly}_6(\lambda, n, \eta), \end{aligned} \tag{21}$$

where $\mathrm{poly}_{1\mathsf{CT}}$ and $\mathrm{poly}_6$ are fixed polynomials and $\gamma'$ is a constant such that $\gamma < \gamma' < 1$. Therefore, for $k \in [\eta]$, we can see that the encryption algorithm of $1\mathsf{CT}^{(k)}$ runs in polynomial of $\lambda$, $n, \eta$, and $s$.

From (20) and (21), the encryption time of $\mathsf{HYBRD}_{\eta}$ is bounded by

$$\eta \cdot \left(s^{\gamma} \cdot \mathrm{poly}_5(\lambda, n, \eta) + s^{\gamma'} \cdot \mathrm{poly}_6(\lambda, n, \eta)\right) \leq s^{\gamma'} \cdot \mathrm{poly}_7(\lambda, n, \eta), \tag{22}$$

where $\mathrm{poly}_7$ is a fixed polynomial.

As stated earlier, $\eta$ is at most sub-linear in $\lambda$ in the construction. Thus, (22) means that the encryption algorithm of $\mathsf{HYBRD}_{\eta}$ runs in polynomial time of $\lambda$, $n$ and $s$. In this case, we can see that all of algorithms of $\mathsf{HYBRD}_{\eta}$ runs in polynomial of $\lambda$, $n$ and $s$. Moreover, (22) means that $\mathsf{HYBRD}_{\eta}$ is weakly succinct.[11]

From these analysis, $\mathsf{HYBRD}_{\eta}$ is $\delta = \lambda^{-\zeta}$-secure and supports $\lambda^{\eta} = \lambda^{\zeta^{1/2}}$ functional keys. More formally, $\mathsf{HYBRD}_{\eta}$ is $(\lambda^{\zeta^{1/2}}, \delta)$-selective-message function private iSKFE. Since $\zeta = \omega(1)$, $\lambda^{\zeta^{1/2}}$ is super-polynomial. Therefore, $\mathsf{HYBRD}_{\eta}$ is $(\mathrm{poly}, \delta)$-selective-message function private. $\square$ (**Theorem 5.2**)

[11] Analogously, we see that if the underlying single-key SKFE is succinct, then so does $\mathsf{HYBRD}_{\eta}$.

## 5.4 Converting iSKFE into SKFE

Finally, we show how to transform iSKFE into SKFE. We provide the overview of this transformation in Remark 2.12 in Section 2.7. 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. $\mathsf{Setup}'$, $\mathsf{Enc}'$, and $\mathsf{Dec}'$ are exactly the same as $\mathsf{Setup}$, $\mathsf{Enc}$, and $\mathsf{Dec}$, respectively. We define $\mathsf{KG}$ as follows.

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

- Generate $i \xleftarrow{\mathsf{r}} \mathcal{I}$ and return $sk_f \leftarrow \mathsf{iKG}(\mathsf{MSK}, f, i)$.

The message and function spaces of $\mathsf{SKFE}$ is the same as those of $\mathsf{iSKFE}$. The correctness of $\mathsf{SKFE}$ directly follows from that of $\mathsf{iSKFE}$. Then, we have the following theorem.

**Theorem 5.3.** *Let* $\mathsf{iSKFE}$ *be* $(\mathrm{poly}, \delta)$*-selective-message function private iSKFE the size of whose index space is* $S$*. Then,* $\mathsf{SKFE}$ *is* $(\mathrm{poly}, \delta')$*-selective-message function private SKFE, where* $\delta' = \frac{1}{S} + \delta$.

**Proof of Theorem 5.3.** We assume that the advantage of any adversary attacking $\mathsf{iSKFE}$ is bounded by $\epsilon$. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{SKFE}$. We assume that $\mathcal{A}$ makes $q$ key queries at most, where $q$ is a polynomial of $\lambda$ and $q \leq S$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{SKFE},\mathcal{A}}(\lambda) \leq 2\left(\frac{q(q-1)}{S} + \frac{\epsilon}{2}\right). \tag{23}$$

This means that if $\mathsf{iSKFE}$ is $\delta$-secure, then $\mathsf{SKFE}$ is $\delta'$ secure, where $\delta' = \frac{1}{S} + \delta$. Below, we prove the above inequality (23). First, consider the following sequence of games.

**Game 0** This is the original selective-message function privacy game regarding $\mathsf{SKFE}$.

**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 $i \xleftarrow{\mathsf{r}} \mathcal{I}$, and checks whether the list $\mathcal{L}$ contains the index $i$ or not. If so, the challenger generates $i' \xleftarrow{\mathsf{r}} \mathcal{I} \setminus \mathcal{L}$ 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). Then, we can estimate the advantage of $\mathcal{A}$ as

$$\begin{aligned}\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{PRDCT},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\leq |\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| + |\Pr[\mathtt{SUC}_1] - \frac{1}{2}|. \end{aligned} \tag{24}$$

Below, we estimate each term on the right side of the inequality (24).

**Lemma 5.4.** $|\Pr[\mathtt{SUC}_0] - \Pr[\mathtt{SUC}_1]| \leq \frac{q(q-1)}{S}$.

**Proof of Lemma 5.4.** 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.
□ (**Lemma 5.4**)

**Lemma 5.5.** $|\Pr[\mathtt{SUC}_1] - \frac{1}{2}| \leq \frac{\epsilon}{2}$.

**Proof of Lemma 5.4.** 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 $\mathsf{iSKFE}$ 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, $|\Pr[\mathtt{SUC}_1] - \frac{1}{2}| \leq \frac{\epsilon}{2}$ □ (**Lemma 5.5**)

From the inequality (24) and Lemmas 5.4 and 5.5, inequality (23) holds. □ (**Theorem 5.3**)

## 5.5 From Single-Key SKFE to Collusion-Resistant SKFE

We combine the results that we proved though Section 5 to give our main result. First, we combine Theorem 5.2 and 5.3, and obtain the following main theorem.

**Theorem 5.6.** *Assuming there exists $(1, \delta)$-selective-message function private SKFE for $\mathsf{P/poly}$ that is weakly succinct, where $\delta(\lambda) = \lambda^{-\zeta}$ and $\zeta = \omega(1)$. Then, there exists $(\mathrm{poly}, \delta')$-selective-message function private SKFE for $\mathsf{P/poly}$, where $\delta'(\lambda) = \lambda^{-\zeta^{1/2}}$.*[12]

Theorem 5.6 states that if the underlying single-key scheme is sub-exponentially secure, then so is the resulting scheme. Formally, we have the following theorem.

**Theorem 5.7.** *Assuming there exists $(1, \delta)$-selective-message function private SKFE for $\mathsf{P/poly}$ that is weakly succinct, where $\delta(\lambda) = 2^{-\lambda^{\gamma}}$ and $\gamma < 1$ is a constant. Then, there exists $(\mathrm{poly}, \delta')$-selective-message function private SKFE for $\mathsf{P/poly}$, where $\delta'(\lambda) = 2^{-\lambda^{\gamma/2}}$.*

Therefore, by combining Theorem 5.7 with the result by Kitagawa *et al.* [KNT17a], we have the following corollary.

**Corollary 5.8.** *Assuming there exist sub-exponentially secure single-key weakly succinct SKFE for $\mathsf{P/poly}$. Then, there exists IO for $\mathsf{P/poly}$.*

# 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 our main results, that is Theorem 5.6 and 5.7, we obtain the following corollary.

**Corollary 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.

## 6.1 Transforming Weakly Succinct SKFE into Succinct One

Ananth *et al.* [AJS15] proved that we can transform a collusion-resistant SKFE scheme into a succinct one. By using this result and Theorem 5.6, we can construct succinct SKFE based on weakly succinct one. However, the construction incurs at least quasi-polynomial security loss due to the security loss of our result.

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

**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 5.2 for $\eta = O(1)$.

By setting $\eta = O(1)$ in the construction of $\mathsf{HYBRD}_\eta$, from the inequality (12), we can construct an iSKFE scheme supporting $\lambda^\eta$ functional keys with polynomial security loss. Let $q = \lambda^\eta$. Then, from the inequality (22), the encryption time of $\mathsf{HYBRD}_\eta$ depends on $q$ only in poly-logarithmic. Namely, $\mathsf{HYBRD}_\eta$ is collusion succinct. Formally, we obtain the following theorem from the analysis in Section 5.3.

**Theorem 6.2.** *Assuming there exists a $(1,\delta)$-selective-message function private SKFE scheme that is weakly succinct. Then, there exists a $(q,\delta)$-selective-message function private iSKFE scheme that is collusion succinct, where $q$ is a fixed polynomial of $\lambda$.*

Such a collusion-succinct scheme can be transformed into a single-key succinct scheme by the technique utilizing decomposable randomized encoding used by Bitansky and Vaikuntanathan [BV15, Proposition IV.1]. They proposed the transformation for a PKFE scheme, but we can easily observe that it works even if the building block scheme is an iSKFE scheme. Note that the resulting scheme is a single-key iSKFE scheme and thus is also a single-key SKFE scheme from the discussion in Remark 2.12. Formally, the following theorem holds.

**Theorem 6.3.** *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.*

To prove Theorem 6.3, we need to prove that we can construct a succinct SKFE scheme from a collusion succinct iSKFE scheme. We again stress that the transformation in this section is almost the same as that proposed by Bitansky and Vaikuntanathan [BV15, Proposition IV.1]. The differences between theirs and ours is that the underlying scheme is an iSKFE scheme and ours focus on a function private scheme while theirs does on a message private scheme.

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 $\mathsf{SCNCT}$, respectively. Let $\mathsf{RE}$ 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 $\mathsf{RE}$ are $\mu$ and $\rho$, respectively. Then, both $\mu$ and $\rho$ are bounded by $s \cdot \mathrm{poly}_{\mathsf{RE}}(\lambda, n)$, where $\mathrm{poly}_{\mathsf{RE}}$ 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 $\mathsf{SCNCT}$ 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 $t \leftarrow \{0,1\}^\lambda$.
- Compute decomposed $f$, that is, $(\widehat{f}_1, \cdots, \widehat{f}_\mu)$ together with $(S_1, \cdots, S_\mu)$ where $S_i \subseteq [\rho]$ and $|S_i| = c$.
- Compute $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\widehat{f}_i, \bot, S_i, \bot, t, \bot], i)$. The circuit $D_{\mathsf{re}}$ is defined in Figure 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 $y$.

**Decomposable randomized encoding circuit** $D_{\mathsf{re}}[\widehat{f}_{0,i}, \widehat{f}_{1,i}, S_{0,i}, S_{1,i}, t, u_i](m, K, \alpha)$

**Hardwired:** decomposed functions $\widehat{f}_{0,i}$ and $\widehat{f}_{1,i}$, sets $S_{0,i}$ and $S_{1,i}$, tag $t$, and functional key $u_i$.
**Input:** message $m$, PRF key $K$, and bit $\alpha$.

```
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 $e_i \leftarrow \widehat{f}_{i,\alpha}(x; r_{S_{i,\alpha}})$.
```

**Figure 5:** Construction of decomposable randomized encoding circuit $D_{\mathsf{re}}$.

The correctness of $\mathsf{SCNCT}$ directly follows from that of $\mathsf{iSKFE}$. Then, we have the following theorem.

**Theorem 6.4.** *Let* $\mathsf{iSKFE}$ *be a* $(\mu, \delta)$*-selective-message function private iSKFE scheme that is collusion succinct. Let* $\mathsf{RE}$ *a be* $\delta$*-secure decomposable randomized encoding. Let* $\mathsf{F}$ *be a* $\delta$*-secure PRF. Then,* $\mathsf{SCNCT}$ *is a* $(1, \delta)$*-selective-message function private SKFE that is succinct.*

**Proof of Theorem 6.4.** We start with analyzing the succinctness of $\mathsf{SCNCT}$, and then move on to the security proof.

**Succinctness.** Let $D_i$ be the circuit $D_{\mathsf{re}}[\widehat{f}_{i,0}, \widehat{f}_{i,1}, S_{i,0}, S_{i,1}, t, u_i]$. $D_i$ includes at most $c$ times PRF evaluation on the domain $\{0,1\}^\lambda \times [\rho]$ and single evaluation of $\widehat{f}_{i,0}$ or $\widehat{f}_{i,1}$. $|\widehat{f}_{i,0}|$ and $|\widehat{f}_{i,1}|$ 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 $\mathsf{RE}$. Therefore, the size of $D_i$ is bounded by $\mathrm{poly}_D(\lambda, n, \log s)$, where $\mathrm{poly}_D$ is a polynomial. Since $\mathsf{iSKFE}$ is collusion succinct, the encryption time of $\mathsf{SCNCT}$ is bounded by

$$\begin{aligned}\mathrm{poly}(\lambda, n, |D_i|, \log \mu) &\leq \mathrm{poly}(\lambda, n, \mathrm{poly}_D(\lambda, n, \log s), \log(s \cdot \mathrm{poly}_{\mathsf{RE}}(\lambda, n)))\\ &\leq \mathrm{poly}'(\lambda, n, \log s),\end{aligned}$$

where $\mathrm{poly}$ and $\mathrm{poly}'$ are polynomials. This implies that $\mathsf{SCNCT}$ is succinct.

**Security Proof.** We assume that the advantages of any adversary attacking $\mathsf{iSKFE}$, $\mathsf{RE}$, and $\mathsf{F}$ are bounded by $\epsilon$, $\epsilon_{\mathsf{RE}}$, and $\epsilon_{\mathsf{PRF}}$, respectively. Let $\mathcal{A}$ be an adversary that attacks the selective-message function privacy of $\mathsf{SCNCT}$. We assume that $\mathcal{A}$ makes $p$ encryption queries at most, where $p$ is a polynomial of $\lambda$. Then, we have

$$\mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{SCNCT},\mathcal{A}}(\lambda) \leq 2\left((2p+1) \cdot \epsilon + 2p \cdot \epsilon_{\mathsf{RE}} + 2p \cdot \epsilon_{\mathsf{PRF}}\right). \quad (25)$$

This means that if all of $\mathsf{iSKFE}$, $\mathsf{RE}$, and $\mathsf{F}$ are $\delta$-secure, then so is $\mathsf{SCNCT}$. Below, we prove the above inequality (25). First, consider the following sequence of games.

**Game 0** This is the original selective-message function privacy game regarding $\mathsf{SCNCT}$.

> **Initialization** First, the challenger sends security parameter $1^\lambda$ to $\mathcal{A}$. Then, $\mathcal{A}$ sends $\{(m_0^\ell, m_1^\ell)\}_{\ell \in [p]}$ 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 $\{(\mathsf{CT}^{(\ell)})\}_{\ell \in [p]}$ 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, $(\widehat{f}_{b,1}, \cdots, \widehat{f}_{b,\mu})$ together with $(S_{b,1}, \cdots, S_{b,\mu})$. Then, the challenger computes $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\widehat{f}_{b,i}, \bot, S_{b,i}, \bot, t, \bot], i)$ 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 $\{\mathsf{CT}^{(\ell)}\}_{\ell \in [p]}$ 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 $f_\alpha$, that is, $(\widehat{f}_{\alpha,1}, \cdots, \widehat{f}_{\alpha,\mu})$ 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 $u_i^{(\ell^*)} \leftarrow \widehat{f}_{i,b}(m_b^{\ell^*}; r_{S_{i,b}}^{(\ell^*)})$ and $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}], i)$. 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, $\mathsf{Sim}$ is a simulator for $\mathsf{RE}$.

**Game** $(4, \ell^*)$ Same as Game $(3, \ell^*)$ except that for every $i \in [\mu]$, the challenger generates $u_i^{(\ell^*)} \leftarrow \widehat{f}_{i,1}(m_1^{\ell^*}; r_{S_{i,1}})$.

**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 $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \bot], i)$.

We define one additional game.

**Game** 7 Same as Game $(6, p)$ except that for every $i \in [\mu]$, the challenger computes $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\bot, \widehat{f}_{1,i}, \bot, S_{1,i}, t, \bot], i)$. 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 $(h, \ell^*)$. 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

$$\begin{aligned}\frac{1}{2} \cdot \mathsf{Adv}^{\mathsf{sm\text{-}fp}}_{\mathsf{HYBRD},\mathcal{A}}(\lambda) &= |\Pr[\mathtt{SUC}_0] - \frac{1}{2}| \\ &\le \sum_{\ell^* \in [p]} |\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \\ &\quad + \sum_{\ell^* \in [p]} \sum_{h=1}^{5} |\Pr[\mathtt{SUC}_{(h,\ell^*)}] - \Pr[\mathtt{SUC}_{(h+1,\ell^*)}]| \\ &\quad + |\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7]| \qquad (26)\end{aligned}$$

Below, we estimate each term on the right side of inequality (26).

**Lemma 6.5.** *For every* $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(6,\ell^*-1)}] - \Pr[\mathtt{SUC}_{(1,\ell^*)}]| \le \epsilon$.

**Proof of Lemma 6.5.** Using the adversary $\mathcal{A}$, we construct the following adversary $\mathcal{B}$ that attacks $\mathsf{iSKFE}$.

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

- For every $\ell < \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (m_1^\ell, K^{(\ell)}, 1)$.
- $\mathcal{B}$ sets $M_0^{\ell^*} = (m_b^{\ell^*}, K^{(\ell^*)}, 0)$ and $M_1^{\ell^*} = (\perp, \perp, 0)$.
- For every $\ell > \ell^*$, $\mathcal{B}$ sets $M_0^\ell = M_1^\ell = (m_b^\ell, K^{(\ell)}, 0)$.

Then, $\mathcal{B}$ sends $\{(M_0^\ell, M_1^\ell)\}_{\ell \in [p]}$ to the challenger and returns the answer $\{\mathsf{CT}^{(\ell)}\}_{\ell \in [p]}$ 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 $f_\alpha$, that is, $(\widehat{f}_{\alpha,1}, \cdots, \widehat{f}_{\alpha,\mu})$ 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 $u_i^{(\ell^*)} \leftarrow \widehat{f}_{b,i}(m_b^{\ell^*}; r_{S_{b,i}}^{(\ell^*)})$. Then, for every $i \in [\mu]$, $\mathcal{B}$ queries $(i, D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \perp], D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}])$ 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 $\beta$ be the challenge bit between the challenger and $\mathcal{B}$. For every $i \in [\mu]$ and $\ell \in [p]$, we have

$$D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \perp](m_b^\ell, K^{(\ell)}, \alpha) = D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}](m_b^\ell, K^{(\ell)}, \alpha),$$

for every $b, \alpha \in \{0,1\}$ if $K^{(\ell)} \neq \perp$ since $u_i^{(\ell^*)}$ is ignored in the right hand side. In addition, it holds that

$$\begin{aligned}D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \perp](m_b^{\ell^*}, K^{(\ell^*)}, 0) &= \widehat{f}_{b,i}(m_b^{\ell^*}; r_{S_{b,i}}^{(\ell^*)}) \\ &= u_i^{(\ell^*)} = D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, u_i^{(\ell^*)}](\perp, \perp, 0).\end{aligned}$$

Moreover, $\mathcal{B}$ makes $\mu$ key queries at most, and each of them is under different index $i \in [\mu]$. Therefore, $\mathcal{B}$ is a valid adversary for $\mathsf{iSKFE}$, 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 6.5**)

**Lemma 6.6.** *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(1,\ell^*)}] - \Pr[\mathtt{SUC}_{(2,\ell^*)}]| \leq \epsilon_{\mathsf{PRF}}$.*

The proof is straightforward thus is omitted.

**Lemma 6.7.** *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{RE}}$.*

**Proof of Lemma 6.7.** In both of Game $(2, \ell^*)$ and Game $(3, \ell^*)$,$\{u_i^{(\ell^*)}\}_{i\in[\mu]}$ is generated using truly random strings $\{r_j^{(\ell^*)}\}_{j\in[\rho]}$. In addition, if $\{u_i^{(\ell^*)}\}_{i\in[\mu]}$ is given, the actual value of $\{r_j^{(\ell^*)}\}_{j\in[\rho]}$ is not needed for simulating both games. The only difference between two games is that $\{u_i^{(\ell^*)}\}_{i\in[\mu]}$ 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 $\mathsf{RE}$, we have $|\Pr[\mathtt{SUC}_{(2,\ell^*)}] - \Pr[\mathtt{SUC}_{(3,\ell^*)}]| \leq \epsilon_{\mathsf{RE}}$. □ (**Lemma 6.7**)

**Lemma 6.8.** *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 6.7 thus is omitted.

**Lemma 6.9.** *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(4,\ell^*)}] - \mathtt{SUC}_{(5,\ell^*)}| \leq \epsilon_{\mathsf{PRF}}$.*

The proof is straightforward thus is omitted.

**Lemma 6.10.** *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 6.5 thus is omitted.

**Lemma 6.11.** *For every $\ell^* \in [p]$, $|\Pr[\mathtt{SUC}_{(6,p)}] - \Pr[\mathtt{SUC}_7|] \leq \epsilon$.*

**Proof of Lemma 6.11.** 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 $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \perp], i)$. On the other hand, in Game 7, it is generated as $\mathsf{sk}_{f_i} \leftarrow \mathsf{iKG}(\mathsf{MSK}, D_{\mathsf{re}}[\perp, \widehat{f}_{1,i}, \perp, S_{1,i}, t, \perp], i)$. 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

$$D_{\mathsf{re}}[\widehat{f}_{b,i}, \widehat{f}_{1,i}, S_{b,i}, S_{1,i}, t, \perp](m_1^\ell, K^{(\ell)}, 1) = D_{\mathsf{re}}[\perp, \widehat{f}_{1,i}, \perp, S_{1,i}, t, \perp](m_1^\ell, K^{(\ell)}, 1).$$

This is because $\widehat{f}_{b,i}$ and $S_{b,i}$ are ignored in the left hand side. Therefore, we can construct an adversary attacking $\mathsf{iSKFE}$ 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 6.11**)

From inequality (26) and Lemmas 6.5 to 6.11, inequality (25) holds. □ (**Theorem 6.4**)

Note that the existence of $\delta$-secure iSKFE scheme implies that of $\delta$-secure decomposable randomized encoding and PRF since they are constructed from $\delta$-secure one-way functions. Thus, from Theorem 6.2 and 6.4, we obtain Theorem 6.3. We again stress that this result incurs only polynomial security loss. □ (**Theorem 6.3**)

## 6.2 Transforming Weakly Selective-Secure SKFE into Selective-Secure One

We can transform an weakly selective-message message private SKFE scheme into a selective-message function private one.

**Theorem 6.12.** *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 6.13 ([KNT17b]).** *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 6.13 by the result of Kitagawa et al. Their construction does not directly use underlying SKFE and first transform it into strong exponentially-efficient IO (SXIO). SXIO that is sufficient for their construction can be based on single-key weakly-succinct SKFE [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 6.13.

In addition, by Theorem 2.8 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

[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.

[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.

[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, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 852–880. 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.

[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.

[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.

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 480–511. Springer, Heidelberg, January 2016.

[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.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

[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.

[KNT17a] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Indistinguishability obfuscation for all circuits from secret-key functional encryption. Cryptology ePrint Archive, Report 2017/361, 2017. http://eprint.iacr.org/2017/361.

[KNT17b] Fuyuki Kitagawa, Ryo Nishimaki, and Keisuke Tanaka. Simple generic constructions of succinct functional encryption. Cryptology ePrint Archive, Report 2017/275, 2017. http://eprint.iacr.org/2017/275.

[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, May 2017.

[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.

[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.

[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.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556.

[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.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015.

[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.