On Extension of Evaluation Algorithms in Keyed-Homomorphic Encryption

Hirotomo Shinoki[1] Koji Nuida[23]

[1] Graduate School of Information Science and Technology, The University of Tokyo
[2] Institute of Mathematics for Industry (IMI), Kyushu University
(`nuida@imi.kyushu-u.ac.jp`)
[3] National Institute of Advanced Industrial Science and Technology (AIST)

Abstract

Homomorphic encryption (HE) is public key encryption that enables computation over ciphertexts without decrypting them, while it is known that HE cannot achieve IND-CCA2 security. To overcome this issue, the notion of keyed-homomorphic encryption (KH-PKE) was introduced, which has a separate homomorphic evaluation key and can achieve stronger security (Emura et al., PKC 2013).

The contributions of this paper are twofold. First, the syntax of KH-PKE supposes that homomorphic evaluation is performed for single operations, and its security notion called KH-CCA security was formulated based on this syntax. Consequently, if the homomorphic evaluation algorithm is enhanced in a way of gathering up sequential operations as a single evaluation, then it is not obvious whether or not KH-CCA security is preserved. In this paper, we show that KH-CCA security is in general not preserved under such modification, while KH-CCA security is preserved when the original scheme additionally satisfies circuit privacy.

Secondly, Catalano and Fiore (ACM CCS 2015) proposed a conversion method from linearly HE schemes into two-level HE schemes, the latter admitting addition and a single multiplication for ciphertexts. In this paper, we extend the conversion to the case of linearly KH-PKE schemes to obtain two-level KH-PKE schemes. Moreover, based on the generalized version of Catalano–Fiore conversion, we also construct a similar conversion from d-level KH-PKE schemes into $2d$-level KH-PKE schemes.

1 Introduction

Homomorphic encryption (HE) [33] is a kind of public key encryption that allows computation over encrypted data without knowing the secret key, and has several applications such as delegated computation on the clouds. Major classes of HE include additive HE [25, 32] and multiplicative HE [17, 34] that allow only a single kind of operations, and fully HE (FHE) [9, 10, 11, 12, 16, 23, 24] that allows arbitrary computation over encrypted data. Among them, there is a trade-off between the efficiency (for additive/multiplicative HE) and the enhanced functionality (for FHE). As an intermediate class, there also exists leveled HE (or somewhat HE) where a limitation on the number of possible operations exists (typically for multiplication) while the efficiency is much better than FHE. In particular, there exist some constructions of two-level HE (2LHE) schemes [1, 8, 22, 26] in which an arbitrary number of additions and a single multiplication are possible. Besides such direct constructions of 2LHE schemes, Catalano and Fiore [14] proposed a general conversion method from an additive HE scheme into a 2LHE scheme (with non-compact level-two ciphertexts). We refer to this method as "Catalano–Fiore conversion" in this paper. They also proposed in [13] a generalization of the conversion to obtain a (slightly restricted) $2d$-level HE scheme (i.e., that allows additions and $2d-1$ multiplications) from a d-level HE scheme.

For ordinary public key encryption (PKE) schemes, IND-CCA2 security is regarded as a standard security requirement due to e.g., Bleichenbacher's attack [6] and the implication of non-malleability from IND-CCA2 security [4]. However, in principle HE schemes cannot achieve IND-CCA2 security due to the ability of unrestricted computation over ciphertexts. To resolve the issue, Emura et al. [19, 20] proposed the notion of keyed-homomorphic PKE (KH-PKE) in which the homomorphic evaluation on ciphertexts requires an evaluation key. They introduced a security definition for KH-PKE called KH-CCA security, which, roughly speaking, ensures IND-CCA2 security for adversaries not having the evaluation key and IND-CCA1 security for those having the evaluation key in advance. It is also known that KH-CCA security implies security against ciphertext validity attacks [18].

As concrete instantiations of KH-PKE, Emura et al. [19, 20] proposed a multiplicative KH-PKE scheme based on the Decisional Diffie–Hellman (DDH) assumption and an additive KH-PKE scheme based on the Decisional Composite Residuosity (DCR) assumption. Multiplicative KH-PKE schemes are also proposed by Libert et al. [29] based on the Decisional Linear (DLIN) assumption and by Jutla and Roy [27] based on the Symmetric External Diffie–Hellman (SXDH) assumption. On the other hand, for fully homomorphic versions of KH-PKE called keyed-FHE, Lai et al. [28] proposed a construction using indistinguishability obfuscation (iO) [2] and recently Sato et al. [35, 36] proposed a construction without iO. Moreover, recently Maeda and Nuida [30] proposed a two-level KH-PKE scheme based on the SXDH assumption. To the best of our knowledge, these are all of the known constructions of KH-PKE schemes in the literature, which are still few in comparison to ordinary (non-keyed) HE schemes. In particular, there exists only one known construction of leveled KH-PKE schemes.

On the other hand, we note that except for keyed-FHE, the homomorphic evaluation algorithm in KH-PKE was formulated in a way of corresponding to a single operation, say $C_1 + C_2$. When we perform two operations sequentially, say $(C_1 + C_2) + C_3$, some instantiation of KH-PKE (such as in [19, 20]) performs a rerandomization at the end of the computation of $C' := C_1 + C_2$ and then another rerandomization at the end of the computation of $C' + C_3$. From the viewpoint of efficiency, we want to gather the two operations as a single operation and perform only one rerandomization at the end of the computation. Now, in order to formalize such a technique, the formulation of the homomorphic evaluation algorithm should be enlarged to also handle such sequential operations at once. However, as an adversary in the KH-CCA game is supposed to have oracle access to the evaluation algorithm, and the modification of the evaluation algorithm as above also enhances the ability of the oracle, the adversary after the modification becomes, in theory, stronger than the original case. As a result, it is not obvious whether or not the KH-CCA security is preserved by this modification of the evaluation algorithm. (We note that, as a related work, Emura et al. [21] studied similar security issues when constructing "mis-operation resistant" searchable homomorphic encryption from keyed-homomorphic identity-based encryption. However, their work only concerned such an issue in some concrete schemes, and no argument was given in the same generality as the present paper.)

1.1 Our Contributions

Our contributions in this paper are twofold. First, we consider the modification of the evaluation algorithm to handle multiple operations at once as in the last paragraph; let $\mathcal{E}$ and $\mathsf{Comp}(\mathcal{E})$ denote the original and the modified KH-PKE schemes, respectively. We show that, in general, the KH-CCA security of $\mathcal{E}$ does not imply the KH-CCA security of $\mathsf{Comp}(\mathcal{E})$; under some reasonable assumptions, we construct a KH-CCA secure $\mathcal{E}$ for which $\mathsf{Comp}(\mathcal{E})$ is not KH-CCA secure (Theorem 1). We also show that, if $\mathcal{E}$ is moreover circuit private, then the KH-CCA security of $\mathcal{E}$ implies the KH-CCA security of $\mathsf{Comp}(\mathcal{E})$ (Theorem 2).

We explain a technical overview of our results above. As the counterexample, from any KH-CCA secure KH-PKE scheme $\mathcal{E}_0$ we construct a KH-CCA secure KH-PKE scheme $\mathcal{E}$ with the following property: $\mathcal{E}$ has a special ciphertext C_0 for which given the result $C + C_0$ of a homomorphic operation for C_0 and any ciphertext C, the original ciphertext C can be easily recovered. Now given a challenge ciphertext C^*, a KH-CCA adversary against $\mathsf{Comp}(\mathcal{E})$ asks the evaluation oracle to obtain at once the ciphertext $(C^* + C') + C_0$ where C' is another ciphertext. Due to the property above, now the adversary recovers the ciphertext $C^* + C'$ and knows its plaintext (and also knows the plaintext of C^* by using the plaintext of C') by querying $C^* + C'$ to the decryption oracle (which is not prohibited, as $C^* + C'$ itself was not returned by the evaluation oracle). Hence $\mathsf{Comp}(\mathcal{E})$ is not KH-CCA secure. On the other hand, the circuit privacy assumed in Theorem 2 guarantees that there exists no such special ciphertext C_0.

Secondly, we extend the Catalano–Fiore conversion for (non-keyed) HE schemes to the case of KH-PKE schemes, to obtain a two-level KH-PKE scheme from a linearly KH-PKE (Theorem 3). We also generalize the result to the case of converting a d-level KH-PKE scheme into a $2d$-level KH-PKE scheme (Theorem 4). As a technical overview, we note that in the original Catalano–Fiore conversion, a level-2 ciphertext consists of a number of level-1 ciphertexts. Therefore, if we just apply it to a KH-PKE scheme, then a level-2 ciphertext of the resulting scheme is malleable even without the evaluation key, which violates the KH-CCA security. To resolve this issue, we modify level-2 ciphertexts by encrypting the whole of each level-2 ciphertext again (where the key for the latter encryption is included in the evaluation key). Assuming appropriate security properties for the latter encryption, an adversary cannot modify nor generate a two-level ciphertext without using the evaluation key or the evaluation oracle. This property enables us to control the behaviors of ciphertexts well in our security proof.

1.2 Organization of the Paper

Section 2 summarizes basic definitions and properties used in this paper. Section 3 summarizes basic definitions for KH-PKE. In Section 4, we describe the first part of our results on the extended evaluation algorithm for multiple sequential operations. Section 5 summarizes the definitions for the original Catalano–Fiore conversion for non-keyed HE schemes. In Section 6, we describe the second part of our results on the extension of the Catalano–Fiore conversion to KH-PKE schemes.

2 Preliminaries

2.1 Basic Definitions and Properties

In this paper, "PPT" is an abbreviation of "probabilistic polynomial-time". We write $x \overset{\$}{\leftarrow} S$ to mean a uniformly random choice of an element x from a finite set S. We say that a function $f \colon \mathbb{N} \to \mathbb{R}$ is *negligible* (in security parameter λ) if for any integer $k > 0$, there exists an integer $\lambda_k > 0$ satisfying that for any $\lambda > \lambda_k$ we have $|f(\lambda)| < \lambda^{-k}$. For random variables X, Y on a finite set U, their statistical distance is defined by

$$\mathsf{SD}[X,Y] = \sum_{u \in U} |\Pr[u \leftarrow X] - \Pr[u \leftarrow Y]| \enspace .$$

The following lemma is used in our security proofs.

Lemma 1 (Difference Lemma [37]). *Suppose that for events* A, B, F *we have* $\Pr[A \wedge \neg F] = \Pr[B \wedge \neg F]$. *Then we have*

$$|\Pr[A] - \Pr[B]| \leq \Pr[F] \enspace .$$

2.2 Homomorphic Encryption

We explain the syntax for (public key) homomorphic encryption (HE). Let $\mathcal{F}$ be a set of possible operations for plaintexts. Then an HE scheme consists of the following four PPT algorithms.

- $\mathsf{Gen}(1^\lambda)$: Given the security parameter λ as input, it outputs a public key pk and a secret key sk.
- $\mathsf{Enc}(\mathsf{pk}, M)$: Given a public key pk and a plaintext M as input, it outputs a ciphertext C.
- $\mathsf{Dec}(\mathsf{sk}, C)$: Given a secret key sk and a ciphertext C as input, it outputs either a plaintext or a failure symbol $\perp$.
- $\mathsf{Eval}(\mathsf{pk}, f, C_1, \dots, C_n)$: Given a public key pk, an n-ary operation $f \in \mathcal{F}$, and ciphertexts $C_1, \dots, C_n$ as input, it outputs either a ciphertext or a failure symbol $\perp$.

We require an HE scheme to satisfy the correctness as follows: for any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$,

- for any plaintext M and any $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M)$, we always have $M \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$;
- for any n-ary operation $f \in \mathcal{F}$ and any ciphertexts $C_1, \ldots, C_n$, if $M_i \leftarrow \mathsf{Dec}(\mathsf{sk}, C_i)$ for $i = 1, \ldots, n$ and $C \leftarrow \mathsf{Eval}(\mathsf{pk}, f, C_1, \ldots, C_n)$, then we always have $f(M_1, \ldots, M_n) \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.

We also consider a d-level HE scheme for integer $d \geq 1$ (where 1-level HE is the same as additive HE). It satisfies the following conditions:

- The level $\ell(C) \in \{1, 2, \ldots, d\}$ is determined for each ciphertext C.
- The encryption algorithm outputs a ciphertext of level 1.
- The addition can be performed for a pair of ciphertexts of the same level, say L, and its output has level L.
- The multiplication can be performed for a pair of ciphertexts of levels L_1 and L_2, respectively, provided $L_1 + L_2 \leq d$; and its output has level $L_1 + L_2$.

2.3 Symmetric Key Encryption

We explain the syntax for symmetric key encryption (SKE). An SKE scheme consists of the following three PPT algorithms.

- $\mathsf{Gen}(1^\lambda)$: Given the security parameter λ as input, it outputs an encryption key K.
- $\mathsf{Enc}(\mathsf{K}, M)$: Given an encryption key K and a plaintext M as input, it outputs a ciphertext C.
- $\mathsf{Dec}(\mathsf{K}, C)$: Given an encryption key K and a ciphertext C as input, it outputs either a plaintext or a failure symbol $\perp$.

We require an SKE scheme to satisfy the correctness: for any $\mathsf{K} \leftarrow \mathsf{Gen}(1^\lambda)$, any plaintext M, and any $C \leftarrow \mathsf{Enc}(\mathsf{K}, M)$, we always have $M \leftarrow \mathsf{Dec}(\mathsf{K}, C)$.

We explain two security definitions for SKE used in this paper.

Definition 1 (IND-CPA Security)**.** We say that an SKE scheme $\mathcal{SE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is (Left-or-Right) *IND-CPA secure* if for any PPT adversary $\mathcal{A}$, the advantage

$$\left| \Pr\left[\mathsf{K} \leftarrow \mathsf{Gen}(1^\lambda); b \xleftarrow{\$} \{0,1\}; b' \leftarrow \mathcal{A}^{\mathcal{O}(b)} : b = b'\right] - \frac{1}{2} \right|$$

is negligible in λ, where $\mathcal{O}(b)$ denotes an oracle that, given two plaintexts m_0, m_1, returns an output of $\mathsf{Enc}(\mathsf{K}, m_b)$.

We note that the "Left-or-Right" type definition above is known to be equivalent to the "Find-then-Guess" type definition of IND-CPA security [3].

The following security notion intuitively means that an adversary (without the encryption key) cannot generate a valid ciphertext.

Definition 2 (INT-CTXT Security)**.** We say that an SKE scheme $\mathcal{SE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is *INT-CTXT secure* if for any PPT adversary $\mathcal{A}$, the winning probability of $\mathcal{A}$ in the following game is negligible in λ:

- First, the challenger generates $\mathsf{K} \leftarrow \mathsf{Gen}(1^\lambda)$ and sets $\mathsf{List} = \emptyset$. Then $\mathcal{A}$ performs the following two kinds of procedures, possibly adaptively and many times:
  - $\mathcal{A}$ sends a plaintext m to the challenger. The challenger sends an output C of $\mathsf{Enc}(\mathsf{K}, m)$ back to $\mathcal{A}$ and appends C to List.
  - $\mathcal{A}$ sends a ciphertext C^* to the challenger. When $C^* \notin \mathsf{List}$ and $\mathsf{Dec}(\mathsf{K}, C^*) \neq \perp$, $\mathcal{A}$ wins the game. Otherwise, the challenger returns to $\mathcal{A}$ "valid" if $\mathsf{Dec}(\mathsf{K}, C^*) \neq \perp$ and "invalid" if $\mathsf{Dec}(\mathsf{K}, C^*) = \perp$.

We note that an SKE scheme satisfying both IND-CPA and INT-CTXT security can be constructed from an IND-CPA secure SKE and an SUF-CMA secure message authentication code explained in the next subsection [5].

2.4 Message Authentication Codes

We explain the syntax for message authentication codes (MACs). A MAC consists of the following three PPT algorithms.

- $\mathsf{Gen}(1^\lambda)$: Given the security parameter λ as input, it outputs a MAC key K.
- $\mathsf{Tag}(\mathsf{K}, M)$: Given a MAC key K and a plaintext M as input, it outputs a MAC tag τ.
- $\mathsf{Verify}(\mathsf{K}, M, \tau)$: Given a MAC key K, a plaintext M, and a MAC tag τ as input, it outputs 0 ("invalid") or 1 ("valid").

We require a MAC to satisfy the correctness: for any $\mathsf{K} \leftarrow \mathsf{Gen}(1^\lambda)$, any plaintext M, and any $\tau \leftarrow \mathsf{Tag}(\mathsf{K}, M)$, we always have $1 \leftarrow \mathsf{Verify}(\mathsf{K}, M, \tau)$.

We explain the security definition for MAC used in this paper.

Definition 3 (SUF-CMA Security)**.** We say that a MAC $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Tag}, \mathsf{Verify})$ is *SUF-CMA secure* if for any PPT adversary $\mathcal{A}$, the winning probability of $\mathcal{A}$ in the following game is negligible in λ:

1. The challenger generates $\mathsf{K} \leftarrow \mathsf{Gen}(1^\lambda)$ and sets $\mathsf{List} = \emptyset$.
2. $\mathcal{A}$ sends a plaintext m to the challenger. The challenger generates $\tau \leftarrow \mathsf{Tag}(\mathsf{K}, m)$, sends (m, τ) back to $\mathcal{A}$, and appends (m, τ) to List. This procedure may be performed multiple times.
3. $\mathcal{A}$ sends a pair (m^*, τ^*) to the challenger. $\mathcal{A}$ wins the game if and only if $(m^*, \tau^*) \notin \mathsf{List}$ and $1 \leftarrow \mathsf{Verify}(\mathsf{K}, m^*, \tau^*)$.

3 Keyed-Homomorphic Public-Key Encryption

We explain the syntax for keyed-homomorphic public key encryption (KH-PKE). Let $\mathcal{M}$ be a plaintext space and let $\mathcal{F} \subset \{f \mid f \colon \mathcal{M}^2 \to \mathcal{M}\}$. A KH-PKE scheme consists of the following four PPT algorithms.

- $\mathsf{Gen}(1^\lambda)$: Given the security parameter λ as input, it outputs a public key pk, a secret key sk, and an evaluation key ek.
- $\mathsf{Enc}(\mathsf{pk}, M)$: Given a public key pk and a plaintext M as input, it outputs a ciphertext C.
- $\mathsf{Dec}(\mathsf{sk}, C)$: Given a secret key sk and a ciphertext C as input, it outputs either a plaintext or a failure symbol $\perp$.
- $\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2)$: Given an evaluation key ek, an operation $f \in \mathcal{F}$, and ciphertexts C_1, C_2 as input, it outputs either a ciphertext or a failure symbol $\perp$.

We require a KH-PKE scheme to satisfy the correctness as follows.

Definition 4 (Correctness for KH-PKE)**.** We say that a KH-PKE scheme is *correct* if the following two conditions hold for any $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$.

- For any plaintext M and any $C \leftarrow \mathsf{Enc}(\mathsf{pk}, M)$, we always have $M \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.
- For any ciphertexts C_1, C_2 and any operation $f \in \mathcal{F}$, if $M_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_1)$, $M_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_2)$, and $C \leftarrow \mathsf{Eval}(\mathsf{ek}, f, C_1, C_2)$, then we always have $f(M_1, M_2) \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.

For the security of KH-PKE, in contrast to ordinary (non-keyed) HE schemes that cannot in principle achieve IND-CCA2 security, a KH-PKE scheme can achieve IND-CCA2 security against adversaries not having an evaluation key. By also considering leakage of an evaluation key, the following KH-CCA security was introduced.

Definition 5 (KH-CCA Security)**.** We say that a KH-PKE scheme is *KH-CCA secure* if for any PPT adversary $\mathcal{A}$, the advantage

$$\Big| \Pr\big[(\mathsf{pk},\mathsf{sk},\mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda); (M_0^*, M_1^*, \mathsf{st}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{find},\mathsf{pk}); \\ b \overset{\$}{\leftarrow} \{0,1\}; C^* \leftarrow \mathsf{Enc}(\mathsf{pk}, M_b^*); b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{guess},\mathsf{st},C^*) : b = b'\big] - \frac{1}{2} \Big|$$

is negligible in λ. Here $\mathcal{O}$ denotes three oracles RevEK, $\mathsf{Dec}(\cdot)$, and $\mathsf{Eval}(\cdot,\cdot,\cdot)$ defined as follows, and we set $\mathsf{List} = \emptyset$ in the find phase and set $\mathsf{List} = \{C^*\}$ at the beginning of the guess phase.

- RevEK : It returns the evaluation key ek. This oracle can be used only once.
- $\mathsf{Dec}(\cdot)$: For a ciphertext C as input, it returns $\perp$ if $C \in \mathsf{List}$, and otherwise it returns an output of $\mathsf{Dec}(\mathsf{sk}, C)$. In the guess phase, this oracle cannot be used when RevEK has been used.
- $\mathsf{Eval}(\cdot,\cdot,\cdot)$: For an operation f and ciphertexts C_1, C_2 as input, it returns an output C of $\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2)$. Moreover, if $C_1 \in \mathsf{List}$ or $C_2 \in \mathsf{List}$, then C is appended to List. This oracle cannot be used when RevEK has been used.

We also extend the notion of circuit privacy for ordinary HE schemes (following the definition by Catalano and Fiore [14]) to the case of KH-PKE.

Definition 6 (Circuit Privacy for KH-PKE)**.** We say that a KH-PKE scheme is *circuit private* if there exist a PPT algorithm Sim and a negligible function ϵ satisfying the following condition: for any $(\mathsf{pk},\mathsf{sk},\mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$, any $f \in \mathcal{F}$, and any ciphertexts C_1, C_2, if $m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_1)$ and $m_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_2)$, then we have

$$\mathsf{SD}[\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2), \mathsf{Sim}(1^\lambda, \mathsf{ek}, f(m_1, m_2))] \le \epsilon(\lambda) \ .$$

4 On Extension of the Evaluation Algorithm

In this section, we introduce an extension of the evaluation algorithm in KH-PKE schemes to multiple sequential operations, and investigate the effect to the security. We describe our two results in Section 4.1, and give proofs for them in Sections 4.2 and 4.3.

4.1 Extension of the Evaluation Algorithm

Our extension of the evaluation algorithm for KH-PKE is defined as follows.

Definition 7. Let $\mathcal{E}$ be a KH-PKE scheme with the set $\mathcal{F}$ of possible operations. Let $C(\mathcal{F})$ denote the set of circuits for which an element of $\mathcal{F}$ is associated to each gate in the circuit. Now for each n-input circuit $f \in C(\mathcal{F})$, we define the extended evaluation algorithm $\mathsf{Eval}(\mathsf{ek}, f, C_1, \dots, C_n)$ by naturally composing the evaluation algorithms in $\mathcal{E}$. We write the resulting scheme with the extended evaluation algorithm as $\mathsf{Comp}(\mathcal{E})$.

For example, if f is a circuit representing the computation $(C_1 + C_2) + C_3$, then

$$\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2, C_3) = \mathsf{Eval}(\mathsf{ek}, +, \mathsf{Eval}(\mathsf{ek}, +, C_1, C_2), C_3) \ .$$

We also naturally extend the KH-CCA security to such a scheme $\mathsf{Comp}(\mathcal{E})$ by modifying the evaluation oracle accordingly. A motivation of considering such an extension $\mathsf{Comp}(\mathcal{E})$ is that the extended evaluation algorithm can sometimes be implemented more efficiently without changing the output distribution. For example, when the evaluation algorithm in the original scheme $\mathcal{E}$ performs a rerandomization for each output, the computation of $\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2, C_3)$ in the example above can be simplified by omitting the first rerandomization at the end of $\mathsf{Eval}(\mathsf{ek}, +, C_1, C_2)$.

Now the extension from $\mathcal{E}$ to $\mathsf{Comp}(\mathcal{E})$ also changes the security definition in a direction of enhancing an oracle, hence strengthening the ability of adversaries. Therefore, it is not obvious whether or not KH-CCA security of $\mathcal{E}$ implies KH-CCA security of $\mathsf{Comp}(\mathcal{E})$. In fact, we have the following non-implication result, which is proved in Section 4.2.

Theorem 1. *Assume that there exist a KH-CCA secure KH-PKE scheme with non-trivial group as the plaintext space, and an SUF-CMA secure MAC. Then there exists a KH-CCA secure KH-PKE scheme $\mathcal{E}$ for which $\mathsf{Comp}(\mathcal{E})$ is not KH-CCA secure.*

On the other hand, as an affirmative result, we show that KH-CCA security of $\mathcal{E}$ implies KH-CCA security of $\mathsf{Comp}(\mathcal{E})$ when $\mathcal{E}$ is moreover circuit private. To state our result, we extend the definitions of KH-PKE and its circuit privacy to the case of leveled KH-PKE.

Definition 8 (Leveled KH-PKE)**.** By modifying KH-PKE, we define a syntax for *leveled KH-PKE* as follows.

- Associated to the ciphertext space $\mathcal{C}$, there exist a finite set S and a polynomial-time computable function $\ell\colon \mathcal{C} \to S$ called the *level function*.
- There exists an element $a \in S$ for which any output ciphertext C of Enc satisfies $\ell(C) = a$. (Intuitively, all the fresh ciphertexts have the same level.)
- For each operation $f \in \mathcal{F}$, there exist a subset $L_f \subset S^2$ and a function $p_f\colon L_f \to S$ satisfying that $\mathsf{Eval}(\mathsf{ek}, f, \cdot, \cdot)$ only accepts pairs of ciphertexts (C_1, C_2) with $(\ell(C_1), \ell(C_2)) \in L_f$ and we have $\ell(\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2)) = p_f(\ell(C_1), \ell(C_2))$ for any such (C_1, C_2). (Intuitively, the latter condition means that the level of the resulting ciphertext by the homomorphic evaluation is determined only by the type of operation and the levels of input ciphertexts.)

Definition 9 (Leveled Circuit Privacy for KH-PKE)**.** Let $\mathcal{E}$ be a leveled KH-PKE scheme with level function ℓ. We say that $\mathcal{E}$ is *leveled circuit private* if there exist a PPT algorithm Sim and a negligible function ϵ satisfying the following: for any $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$, any $f \in \mathcal{F}$, and any ciphertexts C_1, C_2 with $(\ell(C_1), \ell(C_2)) \in L_f$, if $m_1 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_1)$ and $m_2 \leftarrow \mathsf{Dec}(\mathsf{sk}, C_2)$, then we have

$$\mathsf{SD}[\mathsf{Eval}(\mathsf{ek}, f, C_1, C_2), \mathsf{Sim}(1^\lambda, \mathsf{ek}, p_f(\ell(C_1), \ell(C_2)), f(m_1, m_2))] \leq \epsilon(\lambda) \ .$$

We note that any KH-PKE scheme $\mathcal{E}$ is also a leveled KH-PKE scheme with constant level function, and if $\mathcal{E}$ is circuit private then $\mathcal{E}$ is also leveled circuit private. Now our result is as follows, which is proved in Section 4.3.

Theorem 2. *If a leveled KH-PKE scheme $\mathcal{E}$ is KH-CCA secure and leveled circuit private, then $\mathsf{Comp}(\mathcal{E})$ is KH-CCA secure.*

4.2 Proof of Theorem 1

In this section, we prove Theorem 1. Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a KH-CCA secure KH-PKE scheme and $\mathsf{MAC} = (\mathsf{Gen}_{\mathsf{MAC}}, \mathsf{Tag}, \mathsf{Verify})$ be an SUF-CMA secure MAC in the assumptions of the statement. By modifying $\mathcal{E}$, we construct a KH-PKE scheme $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ as in the statement.

The key generation algorithm Gen' is defined as follows:

- $\mathsf{Gen}'(1^\lambda)$: It generates $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ and $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$, and outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}')$ where $\mathsf{ek}' := (\mathsf{ek}, \mathsf{K})$.

Given the ciphertext space $\mathcal{C}$ of $\mathcal{E}$ and the tag space $\mathcal{T}$ of MAC, the ciphertext space of $\mathcal{E}'$ is defined as $\mathcal{C}' := \mathcal{C} \cup (\mathcal{C} \times \mathcal{T}) \cup \{S\}$ where S is an additional symbol. The encryption algorithm Enc' is the same as Enc (hence it outputs an element of $\mathcal{C} \subset \mathcal{C}'$). For the decryption algorithm Dec', roughly speaking, a "tagged" ciphertext in $\mathcal{C} \times \mathcal{T} \subset \mathcal{C}'$ is correctly decrypted if and only if the tag is valid, while $S \in \mathcal{C}'$ is decrypted to the unit element. Precisely, for $\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$ where $\mathsf{ct} \in \mathcal{C}'$,

- when $\mathsf{ct} = c \in \mathcal{C}$, it outputs $\mathsf{Dec}(\mathsf{sk}, c)$;
- when $\mathsf{ct} = (c, \tau) \in \mathcal{C} \times \mathcal{T}$, it outputs $\mathsf{Dec}(\mathsf{sk}, c)$ if $1 \leftarrow \mathsf{Verify}(\mathsf{K}, C, \tau)$, and otherwise it outputs $\perp$;
- when $\mathsf{ct} = S$, it outputs the unit element e of the plaintext space.

For the evaluation algorithm Eval', roughly speaking, the evaluation with ciphertext S switches non-tagged ciphertexts and tagged ciphertexts, and the tagged ciphertexts are correctly evaluated (and the tag is removed) if and only if the tag is valid. Precisely, for $\mathsf{Eval}'(\mathsf{ek}', \mathsf{ct}_1, \mathsf{ct}_2)$ where $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{C}'$,

- when $\mathsf{ct}_i = C_i \in \mathcal{C}$ for each $i \in \{1, 2\}$, it outputs $\mathsf{Eval}(\mathsf{ek}, C_1, C_2)$;
- when $\mathsf{ct}_i = S$ and $\mathsf{ct}_{3-i} = C \in \mathcal{C}$ for some $i \in \{1, 2\}$, it outputs $(C, \mathsf{Tag}(\mathsf{K}, C))$;
- when $\mathsf{ct}_i = S$ and $\mathsf{ct}_{3-i} = (C, \tau) \in \mathcal{C} \times \mathcal{T}$ for some $i \in \{1, 2\}$, it outputs C if $1 \leftarrow \mathsf{Verify}(\mathsf{K}, C, \tau)$, and otherwise it outputs $\perp$;
- when $\mathsf{ct}_1 = \mathsf{ct}_2 = S$, it outputs S.
- when $\mathsf{ct}_i = (C_i, \tau) \in \mathcal{C} \times \mathcal{T}$ and $\mathsf{ct}_{3-i} = C_{3-i} \in \mathcal{C}$ for some $i \in \{1, 2\}$, it outputs $\mathsf{Eval}(\mathsf{ek}, C_1, C_2)$ if $1 \leftarrow \mathsf{Verify}(\mathsf{K}, C_i, \tau)$, and otherwise it outputs $\perp$;
- when $\mathsf{ct}_i = (C_i, \tau_i) \in \mathcal{C} \times \mathcal{T}$ for each $i \in \{1, 2\}$, it outputs $\mathsf{Eval}(\mathsf{ek}, C_1, C_2)$ if $1 \leftarrow \mathsf{Verify}(\mathsf{K}, C_1, \tau_1) = 1$ and $1 \leftarrow \mathsf{Verify}(\mathsf{K}, C_2, \tau_2)$, and otherwise it outputs $\perp$.

We note that the correctness of $\mathcal{E}$ and MAC imply that $\mathcal{E}'$ is also correct.

Now the scheme $\mathsf{Comp}(\mathcal{E}')$ is not KH-CCA secure: given a challenge ciphertext C^* with unknown plaintext m^*, an adversary chooses a circuit representing the function $f(x_1, x_2, x_3) := (x_1 * x_2) * x_3$ where $*$ is the group operation, and generates $C \leftarrow \mathsf{Enc}'(\mathsf{pk}, m)$ with some plaintext $m \neq e$. Then the adversary makes a query to the extended Eval oracle and obtains $(C', \tau) \leftarrow \mathsf{Eval}'(\mathsf{ek}', f, C^*, C, S)$. As C' is a ciphertext of $m^* * m \neq m^*$, the adversary can query $C' \neq C^*$ to the Dec oracle (note that C' itself was not returned by the Eval oracle) and obtain $m^* * m$, from which the adversary can know the m^*.

The remaining task is to show that $\mathcal{E}'$ is KH-CCA secure. In the proof, we define $V(\mathsf{K}, C) = V(\mathsf{K}, S) = 1$ for $C \in \mathcal{C}$ and define $V(\mathsf{K}, (C, \tau))$ for $(C, \tau) \in \mathcal{C} \times \mathcal{T}$ to be the output of $\mathsf{Verify}(\mathsf{K}, C, \tau)$. On the other hand, we define $F(C) = C$ for $C \in \mathcal{C}$, $F(S) = S$, and $F(C, \tau) = C$ for $(C, \tau) \in \mathcal{C} \times \mathcal{T}$.

Let $\mathcal{A}_{\mathcal{E}'}$ be any PPT adversary for the KH-CCA game for $\mathcal{E}'$. To show that the advantage of $\mathcal{A}_{\mathcal{E}'}$ is negligible, we construct a PPT adversary $\mathcal{B}_{\mathcal{E}}$ for the KH-CCA game for $\mathcal{E}$ that executes $\mathcal{A}_{\mathcal{E}'}$ internally. Here we use the following lists:

- DList : the list used in the KH-CCA game for $\mathcal{E}$ (note that $\mathcal{B}_{\mathcal{E}}$ can know the current content of DList by simulating the updates of DList by the challenger)
- DList' : the list used in the KH-CCA game for $\mathcal{E}'$, maintained by $\mathcal{B}_{\mathcal{E}}$; $\mathsf{DList}' := \emptyset$ at the beginning
- IList : a list of pairs (C, m) of a ciphertext $C \in \mathcal{C}$ and a plaintext m satisfying that C is known to have plaintext m from the viewpoint of $\mathcal{B}_{\mathcal{E}}$; $\mathsf{IList} := \emptyset$ at the beginning
- ICList : the list of the first components of the pairs in IList ($\mathsf{ICList} = \emptyset$ at the beginning)

The adversary $\mathcal{B}_{\mathcal{E}}$ performs as follows.

- Given a public key pk for $\mathcal{E}$, $\mathcal{B}_{\mathcal{E}}$ generates $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$ and sends pk to $\mathcal{A}_{\mathcal{E}'}$.
- When $\mathcal{A}_{\mathcal{E}'}$ makes a query to RevEK, $\mathcal{B}_{\mathcal{E}}$ makes a query to the own oracle RevEK to obtain ek, and sends $(\mathsf{ek}, \mathsf{K})$ back to $\mathcal{A}_{\mathcal{E}'}$.
- When $\mathcal{A}_{\mathcal{E}'}$ makes a decryption query with input $\mathsf{ct} \in \mathcal{C}'$:

1. If $\mathsf{ct} \in \mathsf{DList}'$ or $V(\mathsf{K}, \mathsf{ct}) = 0$, then $\mathcal{B}_{\mathcal{E}}$ returns $\perp$ to $\mathcal{A}_{\mathcal{E}'}$.
2. If $\mathsf{ct} = S$, then $\mathcal{B}_{\mathcal{E}}$ returns e to $\mathcal{A}_{\mathcal{E}'}$.
3. If $F(\mathsf{ct}) \in \mathsf{ICList}$ and $(F(\mathsf{ct}), m) \in \mathsf{IList}$ with a plaintext m, then $\mathcal{B}_{\mathcal{E}}$ returns m to $\mathcal{A}_{\mathcal{E}'}$.
4. If $F(\mathsf{ct}) \in \mathsf{DList}$, then $\mathcal{B}_{\mathcal{E}}$ outputs a uniformly random bit b' to the challenger and finishes the game; **we write this event as** T.
5. Otherwise, $\mathcal{B}_{\mathcal{E}}$ makes a decryption query with input $F(\mathsf{ct})$, and forwards the response to $\mathcal{A}_{\mathcal{E}'}$.

- When $\mathcal{A}_{\mathcal{E}'}$ makes an evaluation query with inputs $\mathsf{ct}_1, \mathsf{ct}_2 \in \mathcal{C}'$:
  1. For each $i \in \{1, 2\}$,
     - if $F(\mathsf{ct}_i) \in \mathsf{ICList}$, then let m_i be the plaintext with $(F(\mathsf{ct}_i), m_i) \in \mathsf{IList}$;
     - if $F(\mathsf{ct}_i) \neq S$ and $F(\mathsf{ct}_i) \notin \mathsf{DList}$, then $\mathcal{B}_{\mathcal{E}}$ makes a decryption query with input $F(\mathsf{ct}_i)$, obtains the response m_i, and appends $(F(\mathsf{ct}_i), m_i)$ to IList if $m_i \neq \perp$;
     - otherwise, let $m_i := \perp$.
  2. If $V(\mathsf{K}, \mathsf{ct}_1) = 0$ or $V(\mathsf{K}, \mathsf{ct}_2) = 0$, then $\mathcal{B}_{\mathcal{E}}$ returns $\perp$ to $\mathcal{A}_{\mathcal{E}'}$.
  3. If $\mathsf{ct}_1 = S$ or $\mathsf{ct}_2 = S$, then
     (a) $\mathcal{B}_{\mathcal{E}}$ computes the output ct of $\mathsf{Eval}'(\mathsf{ek}', \mathsf{ct}_1, \mathsf{ct}_2)$ by itself (note that this is possible by the definition of Eval') and returns ct to $\mathcal{A}_{\mathcal{E}'}$;
     (b) if $\mathsf{ct}_1 \in \mathsf{DList}'$ or $\mathsf{ct}_2 \in \mathsf{DList}'$, then $\mathcal{B}_{\mathcal{E}}$ appends ct to DList'.
  4. Otherwise,
     (a) $\mathcal{B}_{\mathcal{E}}$ makes an evaluation query with input $(F(\mathsf{ct}_1), F(\mathsf{ct}_2))$, obtains the response C, and returns C to $\mathcal{A}_{\mathcal{E}'}$;
     (b) if $\mathsf{ct}_1 \in \mathsf{DList}'$ or $\mathsf{ct}_2 \in \mathsf{DList}'$, then $\mathcal{B}_{\mathcal{E}}$ appends C to DList';
     (c) if $m_1 \neq \perp$ and $m_2 \neq \perp$, then $\mathcal{B}_{\mathcal{E}}$ appends $(C, m_1 * m_2)$ to IList.
- When $\mathcal{A}_{\mathcal{E}'}$ gives challenge plaintexts, $\mathcal{B}_{\mathcal{E}}$ forwards them to the challenger, appends the challenger's response C to DList', and returns C to $\mathcal{A}_{\mathcal{E}'}$.
- When $\mathcal{A}_{\mathcal{E}'}$ finally outputs a bit b', $\mathcal{B}_{\mathcal{E}}$ outputs the b' to the challenger and finishes the game.

If the event T does not occur during an execution of $\mathcal{B}_{\mathcal{E}}$, then $\mathcal{B}_{\mathcal{E}}$ perfectly simulates the KH-CCA game for $\mathcal{E}'$ from the viewpoint of $\mathcal{A}_{\mathcal{E}'}$. This and Difference Lemma imply

$$\left| \Pr_{\mathcal{B}_{\mathcal{E}}}[b = b'] - \Pr_{\mathcal{A}_{\mathcal{E}'}}[b = b'] \right| \leq \Pr[T] \ ,$$

therefore by the triangle inequality, the difference of the advantages of $\mathcal{A}_{\mathcal{E}'}$ and $\mathcal{B}_{\mathcal{E}}$ is at most $\Pr[T]$. Hence by the KH-CCA security of $\mathcal{E}$, our task is reduced to showing that $\Pr[T]$ is negligible.

During an execution of $\mathcal{B}_{\mathcal{E}}$, we say that a ciphertext $\mathsf{ct} \in \mathcal{C}'$ at some step is *unallowable* if $\mathsf{ct} \notin \mathsf{DList}'$, $V(\mathsf{K}, \mathsf{ct}) = 1$, $\mathsf{ct} \neq S$, $F(\mathsf{ct}) \notin \mathsf{ICList}$, and $F(\mathsf{ct}) \in \mathsf{DList}$. Let T' denote the event that an unallowable ciphertext is given from $\mathcal{A}_{\mathcal{E}'}$ to $\mathcal{B}_{\mathcal{E}}$ as an input for a decryption or an evaluation query. Note that if the event T occurs with $\mathsf{ct} \in \mathcal{C}'$ then ct is unallowable; hence $\Pr[T] \leq \Pr[T']$, and our task is reduced to showing that $\Pr[T']$ is negligible. Now we have the following lemma.

Lemma 2. *In the current setting, if the event T' occurs firstly with unallowable ciphertext* ct*, then we have* $\mathsf{ct} \in \mathcal{C} \times \mathcal{T}$ *and* ct *was not generated by algorithm* Eval' *executed by* $\mathcal{B}_{\mathcal{E}}$ *at some earlier step.*

Proof. Assume for the contrary that $\mathsf{ct} = C \in \mathcal{C}$, hence $F(\mathsf{ct}) = C$. As $\mathsf{ct} \notin \mathsf{DList}'$, the timing where C was appended to DList is an evaluation query with input $(F(\mathsf{ct}_1), F(\mathsf{ct}_2))$ by $\mathcal{B}_{\mathcal{E}}$ (rather than the challenge query) made in order to respond to an evaluation query by $\mathcal{A}_{\mathcal{E}'}$, which we call the query Q, with input, say $\mathsf{ct}_1, \mathsf{ct}_2$. By the definition of $\mathcal{B}_{\mathcal{E}}$, now $V(\mathsf{K}, \mathsf{ct}_1) = V(\mathsf{K}, \mathsf{ct}_2) = 1$, $\mathsf{ct}_1 \neq S$, $\mathsf{ct}_2 \neq S$, and $\mathsf{ct}_1, \mathsf{ct}_2 \notin \mathsf{DList}'$ at the beginning of the query Q (as otherwise $\mathcal{B}_{\mathcal{E}}$ would append $\mathcal{B}_{\mathcal{E}}$'s response $C = \mathsf{ct}$ to DList', a contradiction). Moreover, as $\mathcal{B}_{\mathcal{E}}$ did not append a pair of the form (C, m) to IList during the query Q (note that $C \notin \mathsf{ICList}$ as ct is unallowable), we have $F(\mathsf{ct}_i) \in \mathsf{DList} \setminus \mathsf{ICList}$ for some $i \in \{1, 2\}$ at the beginning of the query Q. Summarizing the argument, it follows that this ct_i was also unallowable at the beginning of the query Q, meaning that an event T' already occurred at the earlier query Q. This is a contradiction. Hence we have $\mathsf{ct} \in \mathcal{C} \times \mathcal{T}$.

Assume for the contrary that ct was generated by algorithm Eval' executed by $\mathcal{B}_{\mathcal{E}}$ at some earlier step. This step was an evaluation query by $\mathcal{A}_{\mathcal{E}'}$, which we call the query Q', with inputs, say $\mathsf{ct}_1, \mathsf{ct}_2$, where $\mathsf{ct}_i = S$ for some $i \in \{1, 2\}$. By the definition of Eval', ct_{3-i} is of the form $\mathsf{ct}_{3-i} = C_{3-i} \in \mathcal{C}$, hence $V(\mathsf{K}, \mathsf{ct}_{3-i}) = 1$ and $\mathsf{ct}_{3-i} \neq S$. The fact $\mathsf{ct} \notin \mathsf{DList}'$ implies that $\mathsf{ct}_{3-i} \notin \mathsf{DList}'$ at the beginning of the query Q'. Moreover, as $F(\mathsf{ct}) = C_{3-i} = F(\mathsf{ct}_{3-i})$ by the definition of Eval', the fact $F(\mathsf{ct}) \notin \mathsf{ICList}$ implies that $F(\mathsf{ct}_{3-i}) \notin \mathsf{ICList}$, therefore we have $F(\mathsf{ct}_{3-i}) \in \mathsf{DList}$ at the beginning of the query Q' as otherwise a pair of the form $(F(\mathsf{ct}_{3-i}), m_{3-i})$ would be appended to IList during the query Q', a contradiction. Summarizing the argument, it follows that this ct_{3-i} was also unallowable at the beginning of the query Q', meaning that an event T' already occurred at the earlier query Q'. This is a contradiction. Hence the lemma holds. □

We construct a PPT adversary $\mathcal{B}_{\mathsf{MAC}}$ against the SUF-CMA game for MAC as follows. $\mathcal{B}_{\mathsf{MAC}}$ generates $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}) \leftarrow \mathsf{Gen}(1^\lambda)$ and then simulates the KH-CCA game between $\mathcal{B}_{\mathcal{E}}$ and the challenger (by using sk and ek), except for the following differences:

- The simulated $\mathcal{B}_{\mathcal{E}}$ does not generate $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathsf{MAC}}(1^\lambda)$ at the beginning. At every time $\mathcal{B}_{\mathcal{E}}$ is required to generate a tag in order to execute the algorithm Eval', $\mathcal{B}_{\mathsf{MAC}}$ queries the oracle in the SUF-CMA game to obtain the tag.
- If the event T' occurs firstly with unallowable ciphertext ct, then $\mathsf{ct} \in \mathcal{C} \times \mathcal{T}$ by Lemma 2; now $\mathcal{B}_{\mathsf{MAC}}$ outputs the ct to the challenger of the SUF-CMA game. If the event T' has not occurred during the simulated KH-CCA game, then $\mathcal{B}_{\mathsf{MAC}}$ aborts.
- If $\mathcal{A}_{\mathcal{E}'}$ which is executed internally in the simulated $\mathcal{B}_{\mathcal{E}}$ makes a query to RevEK, then $\mathcal{B}_{\mathsf{MAC}}$ aborts. (Note that the event T' never occur after $\mathcal{A}_{\mathcal{E}'}$ makes a query to RevEK, as then $\mathcal{A}_{\mathcal{E}'}$ in the guess phase cannot use the oracles Dec and Eval by the definition of the KH-CCA game. Hence this change does not affect the probability that T' occurs.)

By Lemma 2, if $\mathcal{B}_{\mathsf{MAC}}$ outputs $\mathsf{ct} \in \mathcal{C} \times \mathcal{T}$ (i.e., T' occurs), then the tag in ct was not generated by the challenger of the SUF-CMA game, therefore $\mathcal{B}_{\mathsf{MAC}}$ wins the SUF-CMA game. Moreover, the winning probability of $\mathcal{B}_{\mathsf{MAC}}$ is negligible by the SUF-CMA security of MAC. Hence $\Pr[T']$ is also negligible, as desired. This completes the proof of Theorem 1.

4.3 Proof of Theorem 2

In this section, we prove Theorem 2. Let ℓ denote the level function for $\mathcal{E}$, and let Sim be the algorithm in the leveled circuit privacy for $\mathcal{E}$. Then by the definition of the leveled circuit privacy, the output of $\mathsf{Sim}(1^\lambda, \mathsf{ek}, L, m)$ is a ciphertext of level L for plaintext m except for negligible probability. To simplify the argument, we assume without loss of generality that this property holds with probability 1.

Let $\mathcal{A}_{\mathsf{Comp}}$ be any PPT adversary for the KH-CCA game for $\mathsf{Comp}(\mathcal{E})$. To show that the advantage of $\mathcal{A}_{\mathsf{Comp}}$ is negligible, we construct a PPT adversary $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ ($\nu \in \{0, 1\}$) for the KH-CCA game for $\mathcal{E}$ that executes $\mathcal{A}_{\mathsf{Comp}}$ internally. Here we use the following lists and functions:

- DList : the list used in the KH-CCA game for $\mathcal{E}$ (note that $\mathcal{B}_{\mathcal{E}}$ can know the current content of DList by simulating the updates of DList by the challenger)
- $\mathsf{DList}_{\mathsf{Comp}}$: the list used in the KH-CCA game for $\mathsf{Comp}(\mathcal{E})$, maintained by $\mathcal{B}_{\mathcal{E}}$; $\mathsf{DList}_{\mathsf{Comp}} := \emptyset$ at the beginning ($\mathcal{B}_{\mathcal{E}}$ will be constructed in a way that $\mathsf{DList}_{\mathsf{Comp}} \subset \mathsf{DList}$)

- IList : a list of pairs (C, m) of a ciphertext C and a plaintext m satisfying that C is known to have plaintext m from the viewpoint of $\mathcal{B}_{\mathcal{E}}$; $\mathsf{IList} := \emptyset$ at the beginning
- ICList : the list of the first components of the pairs in IList ($\mathsf{ICList} = \emptyset$ at the beginning)
- F_i ($i \in \{0,1\}$) : the function with domain $\mathsf{DList} \cup \mathsf{ICList}$, satisfying that any $C \in \mathsf{DList} \cup \mathsf{ICList}$ would have plaintext $F_i(C)$ if among the two challenge plaintexts M_0^* and M_1^* the challenger chose M_i^* to generate the challenge ciphertext; we suppose that the list IList will be automatically updated in a way that a pair (C, m) belongs to IList whenever $F_0(C) = F_1(C) = m$

The adversary $\mathcal{B}_{\mathcal{E}}$ performs as follows.

- Given a public key pk for $\mathcal{E}$, $\mathcal{B}_{\mathcal{E}}$ sends pk to $\mathcal{A}_{\mathsf{Comp}}$.
- When $\mathcal{A}_{\mathsf{Comp}}$ makes a query to RevEK, $\mathcal{B}_{\mathcal{E}}$ makes a query to the own oracle RevEK to obtain ek, and sends ek back to $\mathcal{A}_{\mathsf{Comp}}$.
- When $\mathcal{A}_{\mathsf{Comp}}$ makes a decryption query or an evaluation query:
  - If at least one of the input ciphertexts of the query belongs to $\mathsf{DList} \setminus (\mathsf{DList}_{\mathsf{Comp}} \cup \mathsf{ICList})$, **we write this event as T and let C_T be the first such ciphertext; we call C_T *the critical ciphertext*.** Now $\mathcal{B}_{\mathcal{E}}$ makes a query to RevEK to obtain ek, and by using ek, computes $C_{T,i} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, \ell(C_T), F_i(C_T))$ for $i \in \{0,1\}$. Then
    - ∗ if $C_T = C_{T,i}$ with $i \in \{0,1\}$, then $\mathcal{B}_{\mathcal{E}}$ outputs $b' := i$ to the challenger and finishes the game;
    - ∗ otherwise, $\mathcal{B}_{\mathcal{E}}$ outputs a uniformly random bit b' to the challenger and finishes the game.
  - Otherwise, for each input ciphertext C not belonging to DList (hence not belonging to $\mathsf{DList}_{\mathsf{Comp}}$ either, by the property $\mathsf{DList}_{\mathsf{Comp}} \subset \mathsf{DList}$), $\mathcal{B}_{\mathcal{E}}$ makes a decryption query with input C, obtains the response m, and
    - ∗ appends (C, m) to IList and sets $F_0(C) = F_1(C) := m$ if $m \neq \bot$;
    - ∗ returns $\bot$ to $\mathcal{A}_{\mathsf{Comp}}$ if $m = \bot$.

    Note that after this process (when $\bot$ has not been returned to $\mathcal{A}_{\mathsf{Comp}}$), any input ciphertext belongs to $\mathsf{DList}_{\mathsf{Comp}} \cup \mathsf{ICList}$.

    When the query is a decryption query with input ciphertext C, if $C \in \mathsf{DList}_{\mathsf{Comp}}$ then $\mathcal{B}_{\mathcal{E}}$ returns $\bot$ to $\mathcal{A}_{\mathsf{Comp}}$; otherwise, $\mathcal{B}_{\mathcal{E}}$ returns the plaintext m with $(C, m) \in \mathsf{IList}$ to $\mathcal{A}_{\mathsf{Comp}}$.

    On the other hand, when the query is an evaluation query:

    1. For $i = 1, \ldots, n$ recursively, where n is the number of gates in the circuit to be computed, $\mathcal{B}_{\mathcal{E}}$ obtains the output ciphertext $C_{i,\mathrm{out}}$ of the i-th gate with input ciphertexts, say $C_{i,\mathrm{in},1}$ and $C_{i,\mathrm{in},2}$, by making a query to the own evaluation oracle for $\mathcal{E}$ with inputs $C_{i,\mathrm{in},1}, C_{i,\mathrm{in},2}$. At the same time, for each $j \in \{0,1\}$, $\mathcal{B}_{\mathcal{E}}$ computes the value of $F_j(C_{i,\mathrm{out}})$ by using the values of $F_j(C_{i,\mathrm{in},1})$ and $F_j(C_{i,\mathrm{in},2})$.
       - ∗ Now it follows recursively (by using the property $\mathsf{DList}_{\mathsf{Comp}} \subset \mathsf{DList}$) that all of $C_{i,\mathrm{in},1}$, $C_{i,\mathrm{in},2}$, and $C_{i,\mathrm{out}}$ belong to $\mathsf{DList} \cup \mathsf{ICList}$; note that if $F_0(C_{i,\mathrm{in},k}) = F_1(C_{i,\mathrm{in},k})$ for both $k = 0, 1$ then $F_0(C_{i,\mathrm{out}}) = F_1(C_{i,\mathrm{out}})$.
       - ∗ **We call the ciphertexts $C_{1,\mathrm{out}}, \ldots, C_{n-1,\mathrm{out}}$ *intermediate ciphertexts.***
    2. If at least one of the input ciphertexts of the query belongs to $\mathsf{DList}_{\mathsf{Comp}}$, then $\mathcal{B}_{\mathcal{E}}$ appends $C_{n,\mathrm{out}}$ to $\mathsf{DList}_{\mathsf{Comp}}$.
    3. $\mathcal{B}_{\mathcal{E}}$ returns $C_{n,\mathrm{out}}$ to $\mathcal{A}_{\mathsf{Comp}}$.
- When $\mathcal{A}_{\mathsf{Comp}}$ gives challenge plaintexts M_0^* and M_1^*, $\mathcal{B}_{\mathcal{E}}$ sends them to the challenger and obtains the challenge ciphertext C^*, appends C^* to $\mathsf{DList}_{\mathsf{Comp}}$, sets $F_i(C^*) := M_i^*$ for $i \in \{0,1\}$, and returns C^* to $\mathcal{A}_{\mathsf{Comp}}$.

- When $\mathcal{A}_{\mathsf{Comp}}$ finally outputs a bit $\hat{b}$,
  - if $\nu = 0$, then $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ outputs $b' := \hat{b}$ to the challenger;
  - if $\nu = 1$, then $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ outputs a uniformly random bit b' to the challenger.

If the event T does not occur during an execution of $\mathcal{B}_{\mathcal{E}}$, then $\mathcal{B}_{\mathcal{E}}$ perfectly simulates the KH-CCA game for $\mathsf{Comp}(\mathcal{E})$ from the viewpoint of $\mathcal{A}_{\mathsf{Comp}}$. This and Difference Lemma imply

$$\left| \Pr_{\mathcal{B}_{\mathcal{E}}^{(0)}}[b' = b] - \Pr_{\mathcal{A}_{\mathsf{Comp}}}[b' = b] \right| \le \Pr[T]$$

(note that $\Pr[T]$ in $\mathcal{B}_{\mathcal{E}}^{(0)}$ and in $\mathcal{B}_{\mathcal{E}}^{(1)}$ are equal), therefore by the triangle inequality, the difference of the advantages of $\mathcal{A}_{\mathsf{Comp}}$ and $\mathcal{B}_{\mathcal{E}}^{(0)}$ is at most $\Pr[T]$. Hence by the KH-CCA security of $\mathcal{E}$, our task is reduced to showing that $\Pr[T]$ is negligible. Moreover, in an execution of $\mathcal{B}_{\mathcal{E}}^{(1)}$, $\mathcal{B}_{\mathcal{E}}^{(1)}$ wins the KH-CCA game with conditional probability 1 if the event T occurs and $C_T = C_{T,b}$, and otherwise $\mathcal{B}_{\mathcal{E}}^{(1)}$ wins with conditional probability $1/2$. Hence we have

$$\Pr_{\mathcal{B}_{\mathcal{E}}^{(1)}}[b' = b] - \frac{1}{2} = \frac{1}{2} \Pr[T \wedge C_T = C_{T,b}] \ .$$

The left-hand side is negligible by the KH-CCA security of $\mathcal{E}$, therefore $\Pr[T \wedge C_T = C_{T,b}]$ is negligible as well.

Let G_0 denote the original KH-CCA game for $\mathcal{E}$ between $\mathcal{B}_{\mathcal{E}}$ and the challenger. As $\mathcal{B}_{\mathcal{E}}$ is PPT, the number of computations of the evaluation algorithm by the challenger, say $C \leftarrow \mathsf{Eval}(\mathsf{ek}, f, C_1, C_2)$, is bounded by a positive polynomial, say $P(\lambda)$. For $1 \le k \le P(\lambda)$, let G_k be the game modifying G_0 in a way that for the first k computations of Eval as above, the challenger instead computes $m_i \leftarrow \mathsf{Dec}(\mathsf{sk}, C_i)$ for $i = 1, 2$ and $C \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, p_f(\ell(C_1), \ell(C_2)), f(m_1, m_2))$. By the condition of Sim in the leveled circuit privacy, the behaviors of G_{k-1} and G_k have only negligible statistical distance, so do the behaviors of G_0 and $G := G_{P(\lambda)}$. Hence our task is reduced to showing that $\Pr_G[T]$ is negligible, while it follows that $\Pr_G[T \wedge C_T = C_{T,b}]$ is negligible.

We note that in the game G, all the computations of Eval by the challenger have been replaced by Sim. As a result, the distribution of the response by $\mathcal{B}_{\mathcal{E}}$ to an evaluation query from the internal $\mathcal{A}_{\mathsf{Comp}}$ is independent of the intermediate ciphertexts generated during this query. Therefore, the behavior of the internal $\mathcal{A}_{\mathsf{Comp}}$ at some step is independent of the previously generated intermediate ciphertexts. As $\mathcal{B}_{\mathcal{E}}$ is PPT, the number of ciphertexts sent from the internal $\mathcal{A}_{\mathsf{Comp}}$ to $\mathcal{B}_{\mathcal{E}}$ at some query is bounded by a positive polynomial, say $Q(\lambda)$. For each $1 \le h \le Q(\lambda)$, let $C_{\mathcal{A}}^{(h)}$ denote the h-th ciphertext sent from $\mathcal{A}_{\mathsf{Comp}}$ to $\mathcal{B}_{\mathcal{E}}$. Now the event T occurs if and only if for some $1 \le h \le Q(\lambda)$ and some ciphertext C, we have $C_{\mathcal{A}}^{(h)} = C$, $C_{\mathcal{A}}^{(h)}$ is a critical ciphertext, and $C_{\mathcal{A}}^{(h')}$ is not a critical ciphertext for any $h' < h$. As these cases are disjoint, we have

$$\Pr_G[T] = \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]$$

where $E(\vec{C})$ denotes the event that $C_{\mathcal{A}}^{(h')} = C_{h'}$ for any $1 \le h' \le h$, and $E'(\vec{C})$ denotes the event that $C_1, \dots, C_{h-1}$ are not critical and C_h is critical. On the other hand, under the two events $E(\vec{C})$ and $E'(\vec{C})$, we have $C_T = C_h$, therefore the conditional probability of $T \wedge C_T = C_{T,b}$ is equal to $\Pr[C_h \leftarrow \mathsf{Sim}] := \Pr[C_h \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, \ell(C_h), m_h)]$ where m_h is the plaintext for C_h. Hence we have

$$\Pr_G[T \wedge C_T = C_{T,b}] = \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})] \cdot \Pr[C_h \leftarrow \mathsf{Sim}] \ .$$

Moreover, under the event $E(\vec{C})$, the event $E'(\vec{C})$ occurs only if at least one of the intermediate ciphertexts is equal to C_h, which occurs with probability at most $P(\lambda) \cdot \Pr[C_h \leftarrow \mathsf{Sim}]$. Hence we have $\Pr_G[E'(\vec{C}) \mid E(\vec{C})] \leq P(\lambda) \cdot \Pr[C_h \leftarrow \mathsf{Sim}]$, therefore

$$\begin{aligned}
Q(\lambda)P(\lambda) \cdot \Pr_G[T \wedge C_T = C_{T,b}] &\geq Q(\lambda) \cdot \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]^2 \\
&\geq \left(\sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \right) \cdot \left(\sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]^2 \right) .
\end{aligned}$$

By applying Cauchy–Schwarz inequality to two vectors $(\Pr_G[E(\vec{C})]^{1/2})_{h,\vec{C}}$ and $(\Pr_G[E(\vec{C})]^{1/2} \Pr_G[E'(\vec{C}) \mid E(\vec{C})])_{h,\vec{C}}$ in the right-hand side, we have

$$\begin{aligned}
\sqrt{Q(\lambda)P(\lambda) \cdot \Pr_G[T \wedge C_T = C_{T,b}]} &\geq \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})]^{1/2} \cdot \Pr_G[E(\vec{C})]^{1/2} \Pr_G[E'(\vec{C}) \mid E(\vec{C})] \\
&= \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})] \\
&= \Pr_G[T] \ .
\end{aligned}$$

Now the left-hand side is negligible, as $\Pr_G[T \wedge C_T = C_{T,b}]$ is negligible as shown above and $P(\lambda)$ and $Q(\lambda)$ are polynomials. Hence it follows that $\Pr_G[T]$ is also negligible, as desired. This completes the proof of Theorem 2.

5 Catalano–Fiore Conversion

In this section, we summarize the original Catalano–Fiore conversion [14] and its generalization [13] for the case of non-keyed HE schemes.

5.1 The Original Catalano–Fiore Conversion

We explain the Catalano–Fiore conversion [14] for linearly HE schemes; i.e., having a finite commutative unital ring as the plaintext space and allowing additions $C_1 \boxplus C_2$ and scalar multiplication $m \boxdot C$ (the latter generates a ciphertext for plaintext $m \cdot \mathsf{Dec}(\mathsf{sk}, C)$). We say that an HE scheme is *public-space* if a uniformly random plaintext can be efficiently sampled. Note that many HE schemes are public-space, and most of the known non-public-space HE schemes such as [7, 8, 15, 31] can be easily converted to public-space schemes.

The conversion yields a two-level HE (2LHE) scheme; that is, the ciphertexts are classified into level-1 ciphertexts and level-2 ciphertexts; homomorphic addition is possible for ciphertexts of the same level; and homomorphic multiplication for two level-1 ciphertexts is possible and yields a level-2 ciphertext. The conversion is described as follows.

Definition 10 (Catalano–Fiore Conversion)**.** Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a linearly HE scheme that is public-space in the sense described above. For the algorithm Eval, we write its special case of addition as Add and of scalar multiplication as cMult, and abbreviate them as the operators $\boxplus$ and $\boxdot$, respectively. Then we define a new 2LHE scheme $\mathsf{CF}(\mathcal{E}) = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}'_1, \mathsf{Dec}'_2, \mathsf{Eval}')$ as follows. Here Dec'_1 and Dec'_2 are decryption algorithms for level-1 and level-2 ciphertexts, respectively.

- Gen' and Enc' : The same as Gen and Enc.

- $\mathsf{Eval}'(\mathsf{pk}, f, \boldsymbol{C})$: We suppose that $\boldsymbol{C}$ is a pair of two ciphertexts when f is addition or multiplication, and $\boldsymbol{C}$ is a single ciphertext when f is scalar multiplication. For $i = 1, 2$, we write Add_i to mean addition for level-i ciphertexts, cMult_i to mean scalar multiplication for level-i ciphertexts, and Mult to mean multiplication for level-1 ciphertexts. These algorithms are defined as follows.
  - Add_1 and cMult_1 : The same as Add and cMult.
  - $\mathsf{Mult}(\mathsf{pk}, C_1, C_2)$: It chooses plaintexts $m_1, m_2 \leftarrow \mathcal{M}$ uniformly at random, sets $\alpha \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1 \cdot m_2)$, and for each $i = 1, 2$, sets $C'_i \leftarrow \mathsf{Enc}(\mathsf{pk}, -m_i)$ and $\beta_i \leftarrow C_i \boxplus C'_i$. Then it computes
  $$\gamma \leftarrow \alpha \boxplus (m_2 \boxdot \beta_1) \boxplus (m_1 \boxdot \beta_2)$$
  (where the $\boxplus$'s are calculated from left to right), and outputs $(\gamma, \beta_1, \beta_2)$.
  - $\mathsf{Add}_2(\mathsf{pk}, C_1, C_2)$: First, it parses the inputs as
  $$C_1 = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1i}, \beta_{2i}, \dots, \beta_{1n}, \beta_{2n}) \ ,$$
  $$C_2 = (\gamma, \delta_{11}, \delta_{21}, \dots, \delta_{1j}, \delta_{2j}, \dots, \delta_{1m}, \delta_{2m}) \ .$$
  Then it sets $\epsilon \leftarrow \alpha \boxplus \gamma$ and puts
  $$C = (\epsilon, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n}, \delta_{11}, \delta_{21}, \dots, \delta_{1m}, \delta_{2m}) \ .$$
  Finally, it outputs $C' \leftarrow \mathsf{Rerand}(\mathsf{pk}, C)$ where Rerand is as defined later.
  - $\mathsf{cMult}_2(\mathsf{pk}, m, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. Then it sets $\alpha' \leftarrow m \boxdot \alpha$ and for each $k = 1, \dots, n$, sets $\beta'_{1k} \leftarrow m \boxdot \beta_{1k}$, $\beta'_{2k} \leftarrow \beta_{2k}$, and puts $C' = (\alpha', \beta'_{11}, \beta'_{21}, \dots, \beta'_{1n}, \beta'_{2n})$. Finally, it outputs $C'' \leftarrow \mathsf{Rerand}(\mathsf{pk}, C')$ where Rerand is as defined later.
- $\mathsf{Dec}'_1(\mathsf{sk}, C)$: It outputs $m \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.
- $\mathsf{Dec}'_2(\mathsf{sk}, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. It computes
$$m = \mathsf{Dec}(\mathsf{sk}, \alpha) + \sum_{i=1}^{n} \mathsf{Dec}(\mathsf{sk}, \beta_{1i}) \cdot \mathsf{Dec}(\mathsf{sk}, \beta_{2i})$$
and outputs m.

Now the algorithm Rerand used in the construction of Eval' is given as follows.

- $\mathsf{Rerand}(\mathsf{pk}, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. For each $i = 1, 2$ and $j = 1, \dots, n$, it chooses $m_{ij} \leftarrow \mathcal{M}$ uniformly at random and sets $\gamma_{ij} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{ij})$. Moreover, it sets $\beta'_{ij} \leftarrow \beta_{ij} \boxplus \gamma_{ij}$ and $\delta_j \leftarrow \mathsf{Enc}(\mathsf{pk}, -(m_{1j} \cdot m_{2j}))$. Then it sets
$$\epsilon_j \leftarrow \delta_j \boxplus ((-m_{2j}) \boxdot \beta_{1j}) \boxplus ((-m_{1j}) \boxdot \beta_{2j})$$
and $\alpha' \leftarrow \alpha \boxplus \epsilon_1 \boxplus \cdots \boxplus \epsilon_n$, and outputs $C' = (\alpha', \beta'_{11}, \beta'_{21}, \dots, \beta'_{1n}, \beta'_{2n})$.

5.2 The Generalized Catalano–Fiore Conversion

In [13], Catalano and Fiore proposed a generalized conversion method from a d-level HE scheme into a $2d$-level HE scheme with a restriction that in the resulting $2d$-level HE scheme, homomorphic multiplication for a pair of ciphertexts including one of level $d + 1$ or higher is not possible. The conversion is described as follows.

Definition 11 (Generalized Catalano–Fiore Conversion)**.** Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a d-level HE scheme that is public-space in the sense described above. For the algorithm Eval, we write its special case of addition as Add, scalar multiplication as cMult, and multiplication as Mult, and abbreviate Add and cMult as the operators $\boxplus$ and $\boxdot$, respectively. Here, when the two input ciphertexts for Add have different levels, it is interpreted in a way that the ciphertext with lower level is implicitly converted in advance into the level of the other ciphertext. Then we define a new $2d$-level HE scheme $\mathsf{CF}(\mathcal{E}) = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}'_{\leq d}, \mathsf{Dec}'_{>d}, \mathsf{Eval}')$ as follows. Here $\mathsf{Dec}'_{\leq d}$ and $\mathsf{Dec}'_{>d}$ are decryption algorithms for ciphertexts of level $\leq d$ and of level $> d$, respectively.

- Gen' and Enc' : The same as Gen and Enc.
- $\mathsf{Eval}'(\mathsf{pk}, f, \boldsymbol{C})$: We suppose that $\boldsymbol{C}$ is a pair of two ciphertexts when f is addition or multiplication, and $\boldsymbol{C}$ is a single ciphertext when f is scalar multiplication. We write $\mathsf{Add}_{\leq d}$ to mean addition for ciphertexts of level $\leq d$, $\mathsf{Add}_{>d}$ to mean addition for ciphertexts of level $> d$, $\mathsf{cMult}_{\leq d}$ to mean scalar multiplication for ciphertexts of level $\leq d$, $\mathsf{cMult}_{>d}$ to mean scalar multiplication for ciphertexts of level $> d$, $\mathsf{Mult}_{\leq d}$ to mean multiplication for two ciphertexts for which the sum of levels is $\leq d$, and $\mathsf{Mult}_{>d}$ to mean multiplication for two ciphertexts for which the sum of levels is $> d$. These algorithms are defined as follows.
  - $\mathsf{Add}_{\leq d}$, $\mathsf{cMult}_{\leq d}$, and $\mathsf{Mult}_{\leq d}$: The same as Add, cMult, and Mult.
  - $\mathsf{Mult}_{>d}(\mathsf{pk}, C_1, C_2)$: Suppose that C_1 and C_2 are of levels $d_1 \leq d$ and $d_2 \leq d$, respectively. It chooses plaintexts $m_1, m_2 \leftarrow \mathcal{M}$ uniformly at random, sets $\alpha \leftarrow \mathsf{Enc}(\mathsf{pk}, m_1 \cdot m_2)$, and for each $i = 1, 2$, sets $C'_i \leftarrow \mathsf{Enc}(\mathsf{pk}, -m_i)$ and $\beta_i \leftarrow C_i \boxplus C'_i$. Then it computes
  $$\gamma \leftarrow \alpha \boxplus (m_2 \boxdot \beta_1) \boxplus (m_1 \boxdot \beta_2)$$
  (where the $\boxplus$'s are calculated from left to right), and outputs $(\gamma, \beta_1, \beta_2)$.
  - $\mathsf{Add}_{>d}(\mathsf{pk}, C_1, C_2)$: First, it parses the inputs as
  $$C_1 = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1i}, \beta_{2i}, \dots, \beta_{1n}, \beta_{2n}) \ ,$$
  $$C_2 = (\gamma, \delta_{11}, \delta_{21}, \dots, \delta_{1j}, \delta_{2j}, \dots, \delta_{1m}, \delta_{2m}) \ .$$
  Then it sets $\epsilon \leftarrow \alpha \boxplus \gamma$ and puts
  $$C = (\epsilon, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n}, \delta_{11}, \delta_{21}, \dots, \delta_{1m}, \delta_{2m}) \ .$$
  Finally, it outputs $C' \leftarrow \mathsf{Rerand}(\mathsf{pk}, C)$ where Rerand is as defined later.
  - $\mathsf{cMult}_{>d}(\mathsf{pk}, m, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. Then it sets $\alpha' \leftarrow m \boxdot \alpha$ and for each $k = 1, \dots, n$, sets $\beta'_{1k} \leftarrow m \boxdot \beta_{1k}$, $\beta'_{2k} = \beta_{2k}$, and puts $C' = (\alpha', \beta'_{11}, \beta'_{21}, \dots, \beta'_{1n}, \beta'_{2n})$. Finally, it outputs $C'' \leftarrow \mathsf{Rerand}(\mathsf{pk}, C')$ where Rerand is as defined later.
- $\mathsf{Dec}'_{\leq d}(\mathsf{sk}, C)$: It outputs $m \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.
- $\mathsf{Dec}'_{>d}(\mathsf{sk}, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. It computes
$$m = \mathsf{Dec}(\mathsf{sk}, \alpha) + \sum_{i=1}^{n} \mathsf{Dec}(\mathsf{sk}, \beta_{1i}) \cdot \mathsf{Dec}(\mathsf{sk}, \beta_{2i})$$
and outputs m.

Now the algorithm Rerand used in the construction of Eval' is given as follows.

- $\mathsf{Rerand}(\mathsf{pk}, C)$: First, it parses the input C as $C = (\alpha, \beta_{11}, \beta_{21}, \dots, \beta_{1n}, \beta_{2n})$. For each $i = 1, 2$ and $j = 1, \dots, n$, it chooses $m_{ij} \leftarrow \mathcal{M}$ uniformly at random and sets $\gamma_{ij} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{ij})$. Moreover, it sets $\beta'_{ij} \leftarrow \beta_{ij} \boxplus \gamma_{ij}$ and $\delta_j \leftarrow \mathsf{Enc}(\mathsf{pk}, -(m_{1j} \cdot m_{2j}))$. Then it sets
$$\epsilon_j \leftarrow \delta_j \boxplus ((-m_{2j}) \boxdot \beta_{1j}) \boxplus ((-m_{1j}) \boxdot \beta_{2j})$$
and $\alpha' \leftarrow \alpha \boxplus \epsilon_1 \boxplus \cdots \boxplus \epsilon_n$, and outputs $C' = (\alpha', \beta'_{11}, \beta'_{21}, \dots, \beta'_{1n}, \beta'_{2n})$.

6 Catalano–Fiore Conversion for KH-PKE

In this section, we extend the Catalano–Fiore conversion to the case of KH-PKE. In Section 6.1, we show that the original Catalano–Fiore conversion applied to a linearly KH-PKE scheme does not preserve KH-CCA security. In Section 6.2, we describe our proposed extension of the Catalano–Fiore conversion to the case of KH-PKE. In Section 6.3, we give a security proof for our proposed conversion. In Section 6.4, we also give a similar argument for the case of the generalized Catalano–Fiore conversion for leveled HE schemes.

6.1 Motivation: The Original Catalano–Fiore Conversion Fails

First, we consider a two-level KH-PKE scheme $\mathsf{CF}(\mathcal{E})$ obtained by simply applying the original Catalano–Fiore conversion to a KH-PKE scheme $\mathcal{E}$. In this case, $\mathsf{CF}(\mathcal{E})$ is in general not KH-CCA secure even if $\mathcal{E}$ is KH-CCA secure. Indeed, the following properties of $\mathsf{CF}(\mathcal{E})$ are contradictory to KH-CCA security:

- An adversary without the evaluation key, given a level-1 ciphertext C, can still generate a level-2 ciphertext $(C, \mathsf{Enc}(\mathsf{pk}, 0), \mathsf{Enc}(\mathsf{pk}, 0))$ with the same plaintext as C.
- An adversary without the evaluation key, given a level-2 ciphertext $C = (\alpha, \beta_1, \beta_2)$, can still generate another level-2 ciphertext $(\alpha, \beta_2, \beta_1)$ with the same plaintext as C.

6.2 Catalano–Fiore Conversion for KH-PKE

The essence of the attacks mentioned in Section 6.1 is that an adversary can handle each component of a level-2 ciphertext separately. Our idea to prevent such attacks is that we will encrypt the whole of a level-2 ciphertext again by an appropriate SKE scheme. The resulting conversion method is described as follows.

Definition 12 (Our Proposed Conversion for KH-PKE)**.** Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a linearly KH-PKE scheme that is public-space, and let $\mathcal{SE} = (\mathsf{Gen}_{\mathcal{SE}}, \mathsf{Enc}_{\mathcal{SE}}, \mathsf{Dec}_{\mathcal{SE}})$ be an SKE scheme. For the algorithm Eval, we write its special case of addition as Add and of scalar multiplication as cMult, and abbreviate them as the operators $\boxplus$ and $\boxdot$, respectively. Then we define a new two-level KH-PKE scheme $\mathsf{CF}'(\mathcal{E}, \mathcal{SE}) = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}'_1, \mathsf{Dec}'_2, \mathsf{Eval}')$ as follows. Here Dec'_1 and Dec'_2 are decryption algorithms for level-1 and level-2 ciphertexts, respectively.

- $\mathsf{Gen}'(1^\lambda)$: It generates $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek})$ by $\mathsf{Gen}(1^\lambda)$ and K by $\mathsf{Gen}_{\mathcal{SE}}(1^\lambda)$, and outputs $(\mathsf{pk}, \mathsf{sk}, \mathsf{ek}')$ where $\mathsf{ek}' = (\mathsf{ek}, \mathsf{K})$.
- $\mathsf{Enc}'(\mathsf{pk}, m)$: The same as $\mathsf{Enc}(\mathsf{pk}, m)$.
- $\mathsf{Eval}'(\mathsf{ek}', f, \boldsymbol{C})$: We suppose that $\boldsymbol{C}$ is a pair of two ciphertexts when f is addition or multiplication, and $\boldsymbol{C}$ is a single ciphertext when f is scalar multiplication. For $i = 1, 2$, we write Add_i to mean addition for level-i ciphertexts, cMult_i to mean scalar multiplication for level-i ciphertexts, and Mult to mean multiplication for level-1 ciphertexts. These algorithms are defined as follows.
  - $\mathsf{Add}_1(\mathsf{ek}', C_1, C_2)$ and $\mathsf{cMult}_1(\mathsf{ek}', m, C)$: The same as $\mathsf{Add}(\mathsf{ek}, C_1, C_2)$ and $\mathsf{cMult}(\mathsf{ek}, m, C)$.
  - $\mathsf{Mult}(\mathsf{ek}', C_1, C_2)$: It chooses plaintexts $m_1, m_2 \leftarrow \mathcal{M}$ uniformly at random, sets $\alpha \leftarrow \mathsf{Enc}(m_1 \cdot m_2)$, and for each $i = 1, 2$, sets $C'_i \leftarrow \mathsf{Enc}(-m_i)$ and $\beta_i \leftarrow C_i \boxplus C'_i$. Then it computes
  $$\gamma \leftarrow \alpha \boxplus (m_2 \boxdot \beta_1) \boxplus (m_1 \boxdot \beta_2)$$
  (where the $\boxplus$'s are calculated from left to right), and outputs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, \gamma || \beta_1 || \beta_2)$ where "$||$" denotes the concatenation of strings.
  - $\mathsf{Add}_2(\mathsf{ek}', C_1, C_2)$: First, it computes $C'_1 \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C_1)$ and $C'_2 \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C_2)$ (it rejects the input if $C'_1 = \perp$ or $C'_2 = \perp$), and parses them as
  $$C'_1 = \alpha || \beta_{11} || \beta_{21} || \cdots || \beta_{1n} || \beta_{2n} \ ,$$
  $$C'_2 = \gamma || \delta_{11} || \delta_{21} || \cdots || \delta_{1m} || \delta_{2m} \ .$$

Then it sets $\epsilon \leftarrow \alpha \boxplus \gamma$ and puts

$$C' = \epsilon||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}||\delta_{11}||\delta_{21}||\cdots||\delta_{1m}||\delta_{2m} \ .$$

Finally, it computes $C'' \leftarrow \mathsf{Rerand}(\mathsf{ek}, C')$ and outputs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C'')$ where Rerand is as defined later.

– $\mathsf{cMult}_2(\mathsf{ek}', m, C)$: First, it computes $C' \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$ (it rejects the input if $C' = \perp$), and parses C' as $C' = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. Then it sets $\alpha' \leftarrow m \boxdot \alpha$ and for each $k = 1, \ldots, n$, sets $\beta'_{1k} \leftarrow m \boxdot \beta_{1k}$, $\beta'_{2k} = \beta_{2k}$ and puts $C'_0 = \alpha||\beta'_{11}||\beta'_{21}||\cdots||\beta'_{1n}||\beta'_{2n}$. Finally, it computes $C''_0 \leftarrow \mathsf{Rerand}(\mathsf{ek}, C'_0)$ and outputs $C_0 \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C''_0)$ where Rerand is as defined later.

- $\mathsf{Dec}'_1(\mathsf{sk}, C)$: It outputs $m \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.
- $\mathsf{Dec}'_2(\mathsf{sk}, C)$: First, it computes $C' \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$ (it rejects the input if $C' = \perp$), and parses C' as $C' = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. It computes

$$m = \mathsf{Dec}(\mathsf{sk}, \alpha) + \sum_{i=1}^{n} \mathsf{Dec}(\mathsf{sk}, \beta_{1i}) \cdot \mathsf{Dec}(\mathsf{sk}, \beta_{2i})$$

and outputs m.

Now the algorithm Rerand used in the construction of Eval' is given as follows.

- $\mathsf{Rerand}(\mathsf{ek}, S)$: First, it parses the input as $S = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. For each $i = 1, 2$ and $j = 1, \ldots, n$, it chooses $m_{ij} \leftarrow \mathcal{M}$ uniformly at random and sets $\gamma_{ij} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{ij})$. Moreover, it sets $\beta'_{ij} \leftarrow \beta_{ij} \boxplus \gamma_{ij}$ and $\delta_j \leftarrow \mathsf{Enc}(\mathsf{pk}, -(m_{1j} \cdot m_{2j}))$. Then it sets

$$\epsilon_j \leftarrow \delta_j \boxplus ((-m_{2j}) \boxdot \beta_{1j}) \boxplus ((-m_{1j}) \boxdot \beta_{2j})$$

and $\alpha' \leftarrow \alpha \boxplus \epsilon_1 \boxplus \cdots \boxplus \epsilon_n$, and outputs $\alpha'||\beta'_{11}||\beta'_{21}||\cdots||\beta'_{1n}||\beta'_{2n}$.

The correctness of $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ follows from the correctness of $\mathcal{E}$ and $\mathcal{SE}$ and can be verified straightforwardly. In the scheme above, for a level-1 ciphertext C, we define $D(C) := \{C\}$. For a level-2 ciphertext C, we define $D(C)$ to be the set of the level-1 ciphertexts obtained by parsing $\mathsf{Dec}_{\mathcal{SE}}(C)$. We call the elements of $D(C)$ the *components* of C.

We show that the conversion method CF' preserves "KH-CCA secure + circuit private" provided the SKE scheme $\mathcal{SE}$ satisfies certain conditions. Namely, we have the following result, which is proved in Section 6.3.

Theorem 3. *Let $\mathcal{E}$ be a linearly KH-PKE scheme that is KH-CCA secure and circuit private, and let $\mathcal{SE}$ be an SKE scheme that is IND-CPA secure and INT-CTXT secure. Then $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ is KH-CCA secure and leveled circuit private, where the level function ℓ used in the leveled circuit privacy is defined by $\ell(C) = 1$ for any level-1 ciphertext C and $\ell(C) = (2, (|D(C)| - 1)/2)$ for any level-2 ciphertext C.*

6.3 Proof of Theorem 3

In this section, we prove Theorem 3. First, we show that $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ is leveled circuit private. We are going to construct a simulator Sim' for the leveled circuit privacy of $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$. Let Sim be the simulator for the circuit privacy of $\mathcal{E}$.

For the case of level-1 ciphertexts, i.e., $\ell(C) = 1$, we simply define $\mathsf{Sim}'(1^\lambda, \mathsf{ek}', 1, m) := \mathsf{Sim}(1^\lambda, \mathsf{ek}, m)$. Then the condition for Sim implies that Sim' also satisfies the desired condition in this case.

For the case of level-2 ciphertexts, i.e., $\ell(C) = (2, n)$ for some integer $n \geq 1$, we define $\mathsf{Sim}'(1^\lambda, \mathsf{ek}', (2, n), m)$ as follows.

For each $i = 1, 2$ and $j = 1, \dots, n$, it chooses $m_{ij} \leftarrow \mathcal{M}$ uniformly at random and puts

$$\beta_{ij} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, m_{ij}) \ .$$

It moreover sets

$$\alpha \leftarrow \mathsf{Sim}\left(1^\lambda, \mathsf{ek}, m - \sum_{j=1}^{n} m_{1j} \cdot m_{2j}\right)$$

and outputs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, \alpha || \beta_{11} || \beta_{21} || \cdots || \beta_{1n} || \beta_{2n})$.

By the construction of the algorithm Rerand used in the evaluation algorithm of $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ and the condition for Sim, it follows that the output distribution of Sim' is indistinguishable from the correct output distribution of the evaluation algorithm. Hence $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ is leveled circuit private, as desired.

The remaining task is to show that $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ is KH-CCA secure. Let $\mathcal{A}_{\mathsf{CF}'}$ be any PPT adversary for the KH-CCA game for $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$. To show that the advantage of $\mathcal{A}_{\mathsf{CF}'}$ is negligible, we construct a PPT adversary $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ ($\nu \in \{0, 1\}$) for the KH-CCA game for $\mathcal{E}$ that executes $\mathcal{A}_{\mathsf{CF}'}$ internally. Here we use the following lists and functions:

- DList : the list used in the KH-CCA game for $\mathcal{E}$ (note that $\mathcal{B}_{\mathcal{E}}$ can know the current content of DList by simulating the updates of DList by the challenger)
- $\mathsf{DList}_{\mathsf{CF}'}$: the list used in the KH-CCA game for $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$, maintained by $\mathcal{B}_{\mathcal{E}}$; $\mathsf{DList}_{\mathsf{CF}'} := \emptyset$ at the beginning ($\mathcal{B}_{\mathcal{E}}$ will be constructed in a way that any level-1 ciphertext in $\mathsf{DList}_{\mathsf{CF}'}$ also belongs to DList; **below we write this property as (*)**)
- IList : a list of pairs (χ, m) of a ciphertext χ in $\mathcal{E}$ and a plaintext m satisfying that χ is known to have plaintext m from the viewpoint of $\mathcal{B}_{\mathcal{E}}$; $\mathsf{IList} := \emptyset$ at the beginning
- ICList : the list of the first components of the pairs in IList ($\mathsf{ICList} = \emptyset$ at the beginning)
- F_i ($i \in \{0, 1\}$) : the function with domain $\mathsf{DList} \cup \mathsf{ICList}$, satisfying that any $\chi \in \mathsf{DList} \cup \mathsf{ICList}$ would have plaintext $F_i(\chi)$ if among the two challenge plaintexts M_0^* and M_1^* the challenger chose M_i^* to generate the challenge ciphertext; we suppose that the list IList will be automatically updated in a way that a pair (χ, m) belongs to IList whenever $F_0(\chi) = F_1(\chi) = m$

The adversary $\mathcal{B}_{\mathcal{E}}$ performs as follows.

- Given a public key pk for $\mathcal{E}$, $\mathcal{B}_{\mathcal{E}}$ sends pk to $\mathcal{A}_{\mathsf{CF}'}$. Moreover, $\mathcal{B}_{\mathcal{E}}$ generates $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathcal{SE}}(1^\lambda)$.
- When $\mathcal{A}_{\mathsf{CF}'}$ makes a query to RevEK, $\mathcal{B}_{\mathcal{E}}$ makes a query to the own oracle RevEK to obtain ek, and sends $\mathsf{ek}' := (\mathsf{ek}, \mathsf{K})$ back to $\mathcal{A}_{\mathsf{CF}'}$.
- When $\mathcal{A}_{\mathsf{CF}'}$ makes a decryption query or an evaluation query, if at least one input ciphertext C is of level 2 and satisfies that $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$ cannot be appropriately parsed into level-1 ciphertexts, then $\mathcal{B}_{\mathcal{E}}$ returns $\perp$ to $\mathcal{A}_{\mathsf{CF}'}$. Otherwise:
  - If at least one of the input ciphertexts C of the query satisfies that $C \notin \mathsf{DList}_{\mathsf{CF}'}$ and $D(C) \cap (\mathsf{DList} \setminus \mathsf{ICList}) \neq \emptyset$, **we write this event as T, and let C_T be the first such ciphertext and χ_T be the first element of $D(C_T) \cap (\mathsf{DList} \setminus \mathsf{ICList}) \neq \emptyset$; we call C_T *the critical ciphertext* and call χ_T *the critical component*.** Now $\mathcal{B}_{\mathcal{E}}$ makes a query to RevEK to obtain ek, and by using ek, computes $\chi_{T,i} \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, F_i(\chi_T))$ for $i \in \{0, 1\}$. Then
    - if $\chi_T = \chi_{T,i}$ with $i \in \{0, 1\}$, then $\mathcal{B}_{\mathcal{E}}$ outputs $b' := i$ to the challenger and finishes the game;
    - otherwise, $\mathcal{B}_{\mathcal{E}}$ outputs a uniformly random bit b' to the challenger and finishes the game.
  - Otherwise, for each input ciphertext C not belonging to $\mathsf{DList}_{\mathsf{CF}'}$ (note that now $D(C) \cap (\mathsf{DList} \setminus \mathsf{ICList}) = \emptyset$ and hence $D(C) \cap \mathsf{DList} \subset \mathsf{ICList}$), for each $\chi \in D(C) \setminus \mathsf{DList}$, $\mathcal{B}_{\mathcal{E}}$ makes a decryption query with input χ, obtains the response m, and

- $*$ appends (χ, m) to IList and sets $F_0(\chi) = F_1(\chi) := m$ if $m \neq \perp$;
- $*$ returns $\perp$ to $\mathcal{A}_{\mathsf{CF}'}$ if $m = \perp$.

Note that after this process (when $\perp$ has not been returned to $\mathcal{A}_{\mathsf{CF}'}$), any input ciphertext C not belonging to $\mathsf{DList}_{\mathsf{CF}'}$ satisfies that $D(C) \subset \mathsf{ICList}$. In particular, any level-1 input ciphertext belongs either to $\mathsf{DList}_{\mathsf{CF}'}$ (hence to DList as well, by the property (*) mentioned above) or to ICList.

When the query is a decryption query with input ciphertext C, if $C \in \mathsf{DList}_{\mathsf{CF}'}$, then $\mathcal{B}_{\mathcal{E}}$ returns $\perp$ to $\mathcal{A}_{\mathsf{CF}'}$. Otherwise:

- $*$ When C is of level 1, note that $C \in \mathsf{ICList}$ as mentioned above; $\mathcal{B}_{\mathcal{E}}$ returns the plaintext m with $(C, m) \in \mathsf{IList}$ to $\mathcal{A}_{\mathsf{CF}'}$.
- $*$ When C is of level 2, $D(C) \subset \mathsf{ICList}$ as mentioned above; $\mathcal{B}_{\mathcal{E}}$ parses $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$ as

$$\alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n} \ ,$$

  takes the plaintext m_χ with (χ, m_χ) for each $\chi \in D(C)$, and returns $m_\alpha + \sum_{i=1}^{n} m_{\beta_{1i}} \cdot m_{\beta_{2i}}$ to $\mathcal{A}_{\mathsf{CF}'}$.

On the other hand, when the query is an evaluation query, $\mathcal{B}_{\mathcal{E}}$ proceeds as follows. Here, whenever $\mathcal{B}_{\mathcal{E}}$ makes an evaluation query of the form $\mathsf{Add}(\chi_1, \chi_2)$ with $\chi_1, \chi_2 \in \mathsf{DList} \cup \mathsf{ICList}$ and obtains the response χ, $\mathcal{B}_{\mathcal{E}}$ defines $F_i(\chi) := F_i(\chi_1) + F_i(\chi_2)$ for $i \in \{0, 1\}$. Similarly, whenever $\mathcal{B}_{\mathcal{E}}$ makes an evaluation query of the form $\mathsf{cMult}(m, \chi_0)$ with $\chi_0 \in \mathsf{DList} \cup \mathsf{ICList}$ and obtains the response χ, $\mathcal{B}_{\mathcal{E}}$ defines $F_i(\chi) := m \cdot F_i(\chi_0)$ for $i \in \{0, 1\}$.

- $*$ When the query is an evaluation query $\mathsf{Add}_1(C_1, C_2)$, $\mathcal{B}_{\mathcal{E}}$ makes an evaluation query $\mathsf{Add}(C_1, C_2)$ and obtains the response C, and returns C to $\mathcal{A}_{\mathsf{CF}'}$. If C_1 or C_2 belongs to $\mathsf{DList}_{\mathsf{CF}'}$, then $\mathcal{B}_{\mathcal{E}}$ appends C to $\mathsf{DList}_{\mathsf{CF}'}$ (note that now C_1 or C_2 belongs to DList by the property (*), therefore C also belongs to DList and the property (*) is preserved).
- $*$ When the query is an evaluation query $\mathsf{cMult}_1(m, C_0)$, $\mathcal{B}_{\mathcal{E}}$ makes an evaluation query $\mathsf{cMult}(m, C_0)$ and obtains the response C, and returns C to $\mathcal{A}_{\mathsf{CF}'}$. If C_0 belongs to $\mathsf{DList}_{\mathsf{CF}'}$, then $\mathcal{B}_{\mathcal{E}}$ appends C to $\mathsf{DList}_{\mathsf{CF}'}$ (note that now C_0 belongs to DList by the property (*), therefore C also belongs to DList and the property (*) is preserved).
- $*$ When the query is an evaluation query $\mathsf{Mult}_1(C_1, C_2)$, note that $C_1, C_2 \in \mathsf{DList} \cup \mathsf{ICList}$ as mentioned above. Then:
  1. $\mathcal{B}_{\mathcal{E}}$ chooses $m_1, m_2 \leftarrow \mathcal{M}$ uniformly at random, and generates $C_1' \leftarrow \mathsf{Enc}(-m_1)$, $C_2' \leftarrow \mathsf{Enc}(-m_2)$, and $\alpha \leftarrow \mathsf{Enc}(m_1 \cdot m_2)$. Moreover, $\mathcal{B}_{\mathcal{E}}$ defines $F_0(C_j') = F_1(C_j') := -m_j$ for $j \in \{1, 2\}$ and defines $F_0(\alpha) = F_1(\alpha) := m_1 \cdot m_2$.
  2. By making evaluation queries sequentially, $\mathcal{B}_{\mathcal{E}}$ obtains $\beta_j \leftarrow C_j \boxplus C_j'$ for $j \in \{1, 2\}$ and obtains $\gamma \leftarrow \alpha \boxplus (m_2 \boxdot \beta_1) \boxplus (m_1 \boxdot \beta_2)$.
  3. $\mathcal{B}_{\mathcal{E}}$ generates $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, \gamma||\beta_1||\beta_2)$ and returns C to $\mathcal{A}_{\mathsf{CF}'}$. If C_1 or C_2 belongs to $\mathsf{DList}_{\mathsf{CF}'}$, then $\mathcal{B}_{\mathcal{E}}$ appends C to $\mathsf{DList}_{\mathsf{CF}'}$.
- $*$ When the query is an evaluation query $\mathsf{Add}_2(C_1, C_2)$, $\mathcal{B}_{\mathcal{E}}$ computes $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C_1)$ and $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C_2)$, and parses them as $\alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n_1}||\beta_{2n_2}$ and $\gamma||\delta_{11}||\delta_{21}||\cdots||\delta_{1n_2}||\delta_{2n_2}$, respectively. Then, by using the own evaluation queries, $\mathcal{B}_{\mathcal{E}}$ obtains $\epsilon \leftarrow \mathsf{Add}(\alpha, \gamma)$, sets

$$C' = \epsilon||\beta_{11}||\beta_{21}||\cdots||\beta_{1n_1}||\beta_{2n_1}||\delta_{11}||\delta_{21}||\cdots||\delta_{1n_2}||\delta_{2n_2} \ ,$$

  and obtains $C'' \leftarrow \mathsf{Rerand}(\mathsf{ek}, C')$. Finally, $\mathcal{B}_{\mathcal{E}}$ returns $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C'')$ back to $\mathcal{A}_{\mathsf{CF}'}$.
- $*$ When the query is an evaluation query $\mathsf{cMult}_2(m, C_0)$, $\mathcal{B}_{\mathcal{E}}$ computes $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C_0)$ and parses it as $\alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. Then, by using the own evaluation queries, $\mathcal{B}_{\mathcal{E}}$ obtains $\alpha' \leftarrow \mathsf{cMult}(m, \alpha)$; for each $k = 1, \ldots, n$, obtains $\beta_{1k}' \leftarrow \mathsf{cMult}(m, \beta_{1k})$ and puts $\beta_{2k}' = \beta_{2k}$; puts $C' = \alpha'||\beta_{11}'||\beta_{21}'||\cdots||\beta_{1n}'||\beta_{2n}'$ and obtains $C'' \leftarrow \mathsf{Rerand}(\mathsf{ek}, C')$. Finally, $\mathcal{B}_{\mathcal{E}}$ returns $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C'')$ back to $\mathcal{A}_{\mathsf{CF}'}$.

- When $\mathcal{A}_{\mathsf{CF}'}$ gives challenge plaintexts M_0^* and M_1^*, $\mathcal{B}_{\mathcal{E}}$ sends them to the challenger and obtains the challenge ciphertext C^*, appends C^* to $\mathsf{DList}_{\mathsf{CF}'}$, sets $F_i(C^*) := M_i^*$ for $i \in \{0,1\}$, and returns C^* to $\mathcal{A}_{\mathsf{CF}'}$.
- When $\mathcal{A}_{\mathsf{CF}'}$ finally outputs a bit $\widehat{b}$,
  - if $\nu = 0$, then $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ outputs $b' := \widehat{b}$ to the challenger;
  - if $\nu = 1$, then $\mathcal{B}_{\mathcal{E}} = \mathcal{B}_{\mathcal{E}}^{(\nu)}$ outputs a uniformly random bit b' to the challenger.

If the event T does not occur during an execution of $\mathcal{B}_{\mathcal{E}}$, then $\mathcal{B}_{\mathcal{E}}$ perfectly simulates the KH-CCA game for $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ from the viewpoint of $\mathcal{A}_{\mathsf{CF}'}$. This and Difference Lemma imply

$$\left| \Pr_{\mathcal{B}_{\mathcal{E}}^{(0)}}[b' = b] - \Pr_{\mathcal{A}_{\mathsf{CF}'}}[b' = b] \right| \leq \Pr[T]$$

(note that $\Pr[T]$ in $\mathcal{B}_{\mathcal{E}}^{(0)}$ and in $\mathcal{B}_{\mathcal{E}}^{(1)}$ are equal), therefore by the triangle inequality, the difference of the advantages of $\mathcal{A}_{\mathsf{CF}'}$ and $\mathcal{B}_{\mathcal{E}}^{(0)}$ is at most $\Pr[T]$. Hence by the KH-CCA security of $\mathcal{E}$, our task is reduced to showing that $\Pr[T]$ is negligible. As $\Pr[T]$ in $\mathcal{B}_{\mathcal{E}}^{(0)}$ and in $\mathcal{B}_{\mathcal{E}}^{(1)}$ are equal, in what follows we focus on $\mathcal{B}_{\mathcal{E}}^{(1)}$ instead of $\mathcal{B}_{\mathcal{E}}^{(0)}$.

We construct a PPT adversary $\mathcal{B}'_{\mathcal{E}}$ by modifying $\mathcal{B}_{\mathcal{E}}^{(1)}$ in the following manner:

- When $\mathcal{A}_{\mathsf{CF}'}$ makes a RevEK query, $\mathcal{B}'_{\mathcal{E}}$ sends any two challenge plaintexts to the challenger and obtains the challenge ciphertext if $\mathcal{B}'_{\mathcal{E}}$ has not done it, and then outputs a uniformly random bit to the challenger.

As the event T does not occur in an execution of $\mathcal{B}_{\mathcal{E}}^{(1)}$ if $\mathcal{A}_{\mathsf{CF}'}$ makes a RevEK query, it follows that $\Pr[T]$ in $\mathcal{B}_{\mathcal{E}}^{(1)}$ and in $\mathcal{B}'_{\mathcal{E}}$ are equal. Hence our task is reduced to showing that $\Pr_{\mathcal{B}'_{\mathcal{E}}}[T]$ is negligible.

We construct a PPT adversary $\mathcal{B}''_{\mathcal{E}}$ by modifying $\mathcal{B}'_{\mathcal{E}}$ in the following manner:

- $\mathcal{B}''_{\mathcal{E}}$ initializes a list RList to $\emptyset$ at the beginning of the KH-CCA game. Whenever $\mathcal{B}''_{\mathcal{E}}$ performs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$ for some c, $\mathcal{B}''_{\mathcal{E}}$ appends (C, c) to RList. Let RCList denote the set of the first components C of the elements in RList.
- Whenever $\mathcal{B}''_{\mathcal{E}}$ is given from $\mathcal{A}_{\mathsf{CF}'}$ a level-2 ciphertext C as an input of a query, if $C \notin \mathsf{RCList}$ then $\mathcal{B}''_{\mathcal{E}}$ returns $\perp$ back to $\mathcal{A}_{\mathsf{CF}'}$. Otherwise, $\mathcal{B}''_{\mathcal{E}}$ performs by using the element c with $(C, c) \in \mathsf{RList}$ instead of $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$.

Then we have the following property.

Lemma 3. *In the setting above,* $\left|\Pr_{\mathcal{B}'_{\mathcal{E}}}[T] - \Pr_{\mathcal{B}''_{\mathcal{E}}}[T]\right|$ *is negligible.*

Proof. Let E denote the event that $\mathcal{B}''_{\mathcal{E}}$ receives from $\mathcal{A}_{\mathsf{CF}'}$ a level-2 ciphertext C satisfying that $C \notin \mathsf{RCList}$ and $\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C) \neq \perp$. If E does not occur, then the behavior of $\mathcal{B}''_{\mathcal{E}}$ is identical to that of $\mathcal{B}'_{\mathcal{E}}$. Therefore, Difference Lemma implies that $\left|\Pr_{\mathcal{B}'_{\mathcal{E}}}[T] - \Pr_{\mathcal{B}''_{\mathcal{E}}}[T]\right| \leq \Pr_{\mathcal{B}''_{\mathcal{E}}}[E]$. Now our task is reduced to showing that $\Pr_{\mathcal{B}''_{\mathcal{E}}}[E]$ is negligible.

We construct a PPT adversary $\mathcal{A}_{\mathcal{SE}}$ for the INT-CTXT game for $\mathcal{SE}$ as follows.

- $\mathcal{A}_{\mathcal{SE}}$ simulates both $\mathcal{B}''_{\mathcal{E}}$ and the challenger in the KH-CCA game for $\mathcal{E}$, where the simulated $\mathcal{B}''_{\mathcal{E}}$ does not generate $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathcal{SE}}(1^{\lambda})$ and instead performs as follows:
  - Whenever the simulated $\mathcal{B}''_{\mathcal{E}}$ has to perform $\mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$, $\mathcal{A}_{\mathcal{SE}}$ makes the own encryption query in the INT-CTXT game and obtains the result of $\mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$.

- Whenever the simulated $\mathcal{B}''_{\mathcal{E}}$ receives from the internally executed $\mathcal{A}_{\mathsf{CF}'}$ a level-2 ciphertext C with $C \notin \mathsf{RCList}$, $\mathcal{A}_{\mathcal{SE}}$ sends C to the challenger of the INT-CTXT game. If $\mathcal{A}_{\mathcal{SE}}$ does not win the game with this ciphertext, then the challenger always returns "invalid" by the definition of RCList; now $\mathcal{A}_{\mathcal{SE}}$ lets the simulated $\mathcal{B}''_{\mathcal{E}}$ return $\perp$ to $\mathcal{A}_{\mathcal{SE}}$.

Now the simulation of $\mathcal{A}_{\mathcal{SE}}$ is perfect unless $\mathcal{A}_{\mathcal{SE}}$ wins the game, and $\mathcal{A}_{\mathcal{SE}}$ wins the game if and only if the event E occurs in the simulated KH-CCA game. As the winning probability of $\mathcal{A}_{\mathcal{SE}}$ is negligible by the INT-CTXT security of $\mathcal{SE}$, it follows that $\Pr_{\mathcal{B}''_{\mathcal{E}}}[E]$ is also negligible, as desired. Hence Lemma 3 holds. □

By the lemma, our task is reduced to showing that $\Pr_{\mathcal{B}''_{\mathcal{E}}}[T]$ is negligible.

We construct a PPT adversary $\mathcal{B}^{\dagger}_{\mathcal{E}}$ by modifying $\mathcal{B}''_{\mathcal{E}}$ in the following manner:

- Whenever $\mathcal{B}^{\dagger}_{\mathcal{E}}$ performs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$ for some c, $\mathcal{B}^{\dagger}_{\mathcal{E}}$ instead chooses c' with $|c'| = |c|$ uniformly at random, generates $C' \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c')$, and appends (C', c) (instead of (C, c)) to RList if C' has not belonged to RCList. In this case, we define $D(C')$ to be the set of ciphertexts for $\mathcal{E}$ appearing in c.

Then we have the following property.

Lemma 4. *In the setting above,* $\left|\Pr_{\mathcal{B}''_{\mathcal{E}}}[T] - \Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T]\right|$ *is negligible.*

Proof. We construct a PPT adversary $\mathcal{A}_{\mathcal{SE}}$ for the IND-CPA game for $\mathcal{SE}$ as follows.

- $\mathcal{A}_{\mathcal{SE}}$ simulates both $\mathcal{B}''_{\mathcal{E}}$ and the challenger in the KH-CCA game for $\mathcal{E}$, where the simulated $\mathcal{B}''_{\mathcal{E}}$ does not generate $\mathsf{K} \leftarrow \mathsf{Gen}_{\mathcal{SE}}(1^{\lambda})$ and instead performs as follows:
  - Whenever the simulated $\mathcal{B}''_{\mathcal{E}}$ has to perform $\mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$, $\mathcal{A}_{\mathcal{SE}}$ sets $m_0 := c$ and chooses m_1 with $|m_0| = |m_1|$ uniformly at random. $\mathcal{A}_{\mathcal{SE}}$ sends (m_0, m_1) to the challenger of the IND-CPA game and obtains the response C. Then the simulated $\mathcal{B}''_{\mathcal{E}}$ uses C instead of $\mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, c)$ and appends (C, c) to RList if C has not belonged to RCList. In this case, we define $D(C)$ to be the set of ciphertexts for $\mathcal{E}$ appearing in c.
- If the event T occurs during the simulation of $\mathcal{B}''_{\mathcal{E}}$, then $\mathcal{A}_{\mathcal{SE}}$ outputs 0 and finishes the game. If the simulation of $\mathcal{B}''_{\mathcal{E}}$ terminates before the event T occurs, then $\mathcal{A}_{\mathcal{SE}}$ outputs 1 and finishes the game.

Now the behavior of the simulated $\mathcal{B}''_{\mathcal{E}}$ coincides with the original $\mathcal{B}''_{\mathcal{E}}$ if the challenge bit b in the IND-CPA game is $b = 0$, while it coincides with the behavior of $\mathcal{B}^{\dagger}_{\mathcal{E}}$ if $b = 1$. Therefore, the advantage of $\mathcal{A}_{\mathcal{SE}}$, which is negligible by the IND-CPA security of $\mathcal{SE}$, is equal to

$$\left|\Pr[b=0]\Pr_{\mathcal{B}''_{\mathcal{E}}}[T] + \Pr[b=1]\left(1 - \Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T]\right) - \frac{1}{2}\right| = \frac{1}{2}\left|\Pr_{\mathcal{B}''_{\mathcal{E}}}[T] - \Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T]\right| \enspace .$$

Hence Lemma 4 holds. □

By the lemma, our task is reduced to showing that $\Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T]$ is negligible. Moreover, in an execution of $\mathcal{B}^{\dagger}_{\mathcal{E}}$, $\mathcal{B}^{\dagger}_{\mathcal{E}}$ wins the KH-CCA game with conditional probability 1 if the event T occurs and $\chi_T = \chi_{T,i}$, and otherwise $\mathcal{B}^{\dagger}_{\mathcal{E}}$ wins with conditional probability 1/2. Hence we have

$$\Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[b' = b] - \frac{1}{2} = \frac{1}{2}\Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T \wedge \chi_T = \chi_{T,b}] \enspace .$$

The left-hand side is negligible by the KH-CCA security of $\mathcal{E}$, therefore $\Pr_{\mathcal{B}^{\dagger}_{\mathcal{E}}}[T \wedge \chi_T = \chi_{T,b}]$ is negligible as well.

Let G_0 denote the original KH-CCA game for $\mathcal{E}$ between $\mathcal{B}^{\dagger}_{\mathcal{E}}$ and the challenger. As $\mathcal{B}^{\dagger}_{\mathcal{E}}$ is PPT, the number of computations of the evaluation algorithm by the challenger, say $C \leftarrow \mathsf{Eval}(\mathsf{ek}, f, \boldsymbol{C})$, is bounded by a positive polynomial, say $P(\lambda)$. For $1 \le k \le P(\lambda)$, let G_k be the game modifying G_0 in a way that for the first k computations of Eval as above, the challenger instead computes $\boldsymbol{m} \leftarrow \mathsf{Dec}(\mathsf{sk}, \boldsymbol{C})$ and $C \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, f(\boldsymbol{m}))$. By the condition of Sim in the circuit privacy, the behaviors of G_{k-1} and G_k have only negligible statistical distance, so do the behaviors of G_0 and $G := G_{P(\lambda)}$. Hence our task is reduced to showing that $\Pr_G[T]$ is negligible, while it follows that $\Pr_G[T \wedge \chi_T = \chi_{T,b}]$ is negligible.

Now we have the following property.

Lemma 5. *In the game G, if the event T occurs with critical ciphertext C_T, then C_T is a level-1 ciphertext.*

Proof. Assume for the contrary that C_T is a level-2 ciphertext. If C_T were not in RCList, then $\mathcal{B}_{\mathcal{E}}^{\dagger}$ would just return $\perp$ to $\mathcal{A}_{\mathsf{CF}'}$ and hence the event T would not occur. Therefore we have $C_T \in \mathsf{RCList}$. This implies that C_T was sent to $\mathcal{A}_{\mathsf{CF}'}$ as the response to a previous evaluation query, say q.

As C_T was not appended to $\mathsf{DList}_{\mathsf{CF}'}$ at the query q, each input ciphertext C of the query q was not in $\mathsf{DList}_{\mathsf{CF}'}$ at the time of the query q. Now by the construction of $\mathcal{B}_{\mathcal{E}}^{\dagger}$, the fact that the event T did not occur at the time of the query q implies that every element of $D(C)$ is appended to ICList during the query q. This also implies that $D(C_T) \subset \mathsf{ICList}$ at the end of the query q, contradicting the condition of the critical ciphertext C_T. Hence Lemma 5 holds. □

We note that in the game G, all the computations of Eval by the challenger have been replaced by Sim, and all level-2 ciphertexts sent to $\mathcal{A}_{\mathsf{CF}'}$ have been replaced by encryption results by $\mathcal{SE}$ of random plaintexts with appropriate lengths. As a result, the behavior of the internal $\mathcal{A}_{\mathsf{CF}'}$ at some step is independent of the previously generated components of level-2 ciphertexts. As $\mathcal{B}_{\mathcal{E}}^{\dagger}$ is PPT, the number of level-1 ciphertexts sent from the internal $\mathcal{A}_{\mathsf{CF}'}$ to $\mathcal{B}_{\mathcal{E}}^{\dagger}$ at some query is bounded by a positive polynomial, say $Q(\lambda)$. For each $1 \le h \le Q(\lambda)$, let $C_{\mathcal{A}}^{(h)}$ denote the h-th level-1 ciphertext sent from $\mathcal{A}_{\mathsf{CF}'}$ to $\mathcal{B}_{\mathcal{E}}^{\dagger}$. Now the event T occurs if and only if for some $1 \le h \le Q(\lambda)$ and some level-1 ciphertext C, we have $C_{\mathcal{A}}^{(h)} = C$, $C_{\mathcal{A}}^{(h)}$ is a critical ciphertext, and $C_{\mathcal{A}}^{(h')}$ is not a critical ciphertext for any $h' < h$. As these cases are disjoint, we have

$$\Pr_G[T] = \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]$$

where $E(\vec{C})$ denotes the event that $C_{\mathcal{A}}^{(h')} = C_{h'}$ for any $1 \le h' \le h$, and $E'(\vec{C})$ denotes the event that $C_1, \dots, C_{h-1}$ are not critical and C_h is critical. On the other hand, under the two events $E(\vec{C})$ and $E'(\vec{C})$, we have $C_T = \chi_T = C_h$, therefore the conditional probability of $T \wedge \chi_T = \chi_{T,b}$ is equal to $\Pr[C_h \leftarrow \mathsf{Sim}] := \Pr[C_h \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{ek}, m_h)]$ where m_h is the plaintext for C_h. Hence we have

$$\Pr_G[T \wedge \chi_T = \chi_{T,b}] = \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})] \cdot \Pr[C_h \leftarrow \mathsf{Sim}] \enspace .$$

Moreover, under the event $E(\vec{C})$, the event $E'(\vec{C})$ occurs only if at least one of the components of a level-2 ciphertext generated during some query is equal to C_h, which occurs with probability at most $P(\lambda) \cdot \Pr[C_h \leftarrow \mathsf{Sim}]$. Hence we have $\Pr_G[E'(\vec{C}) \mid E(\vec{C})] \le P(\lambda) \cdot \Pr[C_h \leftarrow \mathsf{Sim}]$, therefore

$$\begin{aligned} Q(\lambda)P(\lambda) \cdot \Pr_G[T \wedge \chi_T = \chi_{T,b}] &\ge Q(\lambda) \cdot \sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]^2 \\ &\ge \left(\sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \right) \cdot \left(\sum_{h=1}^{Q(\lambda)} \sum_{\vec{C}=(C_1,\dots,C_h)} \Pr_G[E(\vec{C})] \cdot \Pr_G[E'(\vec{C}) \mid E(\vec{C})]^2 \right) \enspace . \end{aligned}$$

By applying Cauchy–Schwarz inequality to two vectors $(\Pr_G[E(\vec{C})]^{1/2})_{h,\vec{C}}$ and $(\Pr_G[E(\vec{C})]^{1/2}\Pr_G[E'(\vec{C}) \mid E(\vec{C})])_{h,\vec{C}}$ in the right-hand side, we have

$$\begin{aligned}\sqrt{Q(\lambda)P(\lambda)\cdot\Pr_G[T\wedge\chi_T=\chi_{T,b}]} &\geq \sum_{h=1}^{Q(\lambda)}\sum_{\vec{C}=(C_1,\dots,C_h)}\Pr_G[E(\vec{C})]^{1/2}\cdot\Pr_G[E(\vec{C})]^{1/2}\Pr_G[E'(\vec{C})\mid E(\vec{C})]\\ &= \sum_{h=1}^{Q(\lambda)}\sum_{\vec{C}=(C_1,\dots,C_h)}\Pr_G[E(\vec{C})]\cdot\Pr_G[E'(\vec{C})\mid E(\vec{C})]\\ &= \Pr_G[T]\ .\end{aligned}$$

Now the left-hand side is negligible, as $\Pr_G[T\wedge\chi_T=\chi_{T,b}]$ is negligible as shown above and $P(\lambda)$ and $Q(\lambda)$ are polynomials. Hence it follows that $\Pr_G[T]$ is also negligible, as desired. This completes the proof of Theorem 3.

6.4 For the Generalized Catalano–Fiore Conversion

For the generalized Catalano–Fiore conversion from a d-level HE scheme into a $2d$-level HE scheme, we can also extend it to the case of KH-PKE schemes. Our conversion method is described as follows.

Definition 13 (Our Proposed Conversion for Leveled KH-PKE). Let $\mathcal{E}=(\mathsf{Gen},\mathsf{Enc},\mathsf{Dec},\mathsf{Eval})$ be a d-level KH-PKE scheme that is public-space, and let $\mathcal{SE}=(\mathsf{Gen}_{\mathcal{SE}},\mathsf{Enc}_{\mathcal{SE}},\mathsf{Dec}_{\mathcal{SE}})$ be an SKE scheme. For the algorithm Eval, we write its special case of addition as Add, of scalar multiplication as cMult, and of multiplication as Mult; and abbreviate Add and cMult as the operators $\boxplus$ and $\boxdot$, respectively. Then we define a new $2d$-level KH-PKE scheme $\mathsf{CF}'(\mathcal{E},\mathcal{SE})=(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}'_{\leq d},\mathsf{Dec}'_{>d},\mathsf{Eval}')$ as follows. Here $\mathsf{Dec}'_{\leq d}$ and $\mathsf{Dec}'_{>d}$ are decryption algorithms for ciphertexts of level $\leq d$ and level $>d$, respectively.

- $\mathsf{Gen}'(1^\lambda)$: It generates $(\mathsf{pk},\mathsf{sk},\mathsf{ek})$ by $\mathsf{Gen}(1^\lambda)$ and K by $\mathsf{Gen}_{\mathcal{SE}}(1^\lambda)$, and outputs $(\mathsf{pk},\mathsf{sk},\mathsf{ek}')$ where $\mathsf{ek}'=(\mathsf{ek},\mathsf{K})$.
- $\mathsf{Enc}'(\mathsf{pk},m)$: The same as $\mathsf{Enc}(\mathsf{pk},m)$.
- $\mathsf{Eval}'(\mathsf{ek}',f,\boldsymbol{C})$: We suppose that $\boldsymbol{C}$ is a pair of two ciphertexts when f is addition or multiplication, and $\boldsymbol{C}$ is a single ciphertext when f is scalar multiplication. We write $\mathsf{Add}_{\leq d}$ to mean addition for ciphertexts of level $\leq d$, $\mathsf{Add}_{>d}$ to mean addition for ciphertexts of level $>d$, $\mathsf{cMult}_{\leq d}$ to mean scalar multiplication for ciphertext of level $\leq d$, $\mathsf{cMult}_{>d}$ to mean scalar multiplication for ciphertext of level $>d$, $\mathsf{Mult}_{\leq d}$ to mean multiplication for two ciphertexts for which the sum of levels is $\leq d$, and $\mathsf{Mult}_{>d}$ to mean multiplication for two ciphertexts for which the sum of levels is $>d$. These algorithms are defined as follows.
  - $\mathsf{Add}_{\leq d}(\mathsf{ek}',C_1,C_2)$, $\mathsf{cMult}_{\leq d}(\mathsf{ek}',m,C)$, and $\mathsf{Mult}_{\leq d}(\mathsf{ek}',C_1,C_2)$: The same as $\mathsf{Add}(\mathsf{ek},C_1,C_2)$, $\mathsf{cMult}(\mathsf{ek},m,C)$, and $\mathsf{Mult}(\mathsf{ek},C_1,C_2)$.
  - $\mathsf{Mult}_{>d}(\mathsf{ek}',C_1,C_2)$: Suppose that C_1 and C_2 are of levels $d_1\leq d$ and $d_2\leq d$, respectively. It chooses plaintexts $m_1,m_2\leftarrow\mathcal{M}$ uniformly at random, sets $\alpha\leftarrow\mathsf{Enc}(\mathsf{pk},m_1\cdot m_2)$, and for each $i=1,2$, sets $C'_i\leftarrow\mathsf{Enc}(\mathsf{pk},-m_i)$ and $\beta_i\leftarrow C_i\boxplus C'_i$. Then it computes

    $$\gamma\leftarrow\alpha\boxplus(m_2\boxdot\beta_1)\boxplus(m_1\boxdot\beta_2)$$

    (where the $\boxplus$'s are calculated from left to right), and outputs $C\leftarrow\mathsf{Enc}_{\mathcal{SE}}(\mathsf{K},\gamma||\beta_1||\beta_2)$.
  - $\mathsf{Add}_{>d}(\mathsf{ek}',C_1,C_2)$: First, it computes $C'_1\leftarrow\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K},C_1)$ and $C'_2\leftarrow\mathsf{Dec}_{\mathcal{SE}}(\mathsf{K},C_2)$, and parses them as

    $$\begin{aligned}C'_1&=\alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}\ ,\\ C'_2&=\gamma||\delta_{11}||\delta_{21}||\cdots||\delta_{1m}||\delta_{2m}\ .\end{aligned}$$

Then it sets $\epsilon \leftarrow \alpha \boxplus \gamma$ and puts

$$C' = \epsilon||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}||\delta_{11}||\delta_{21}||\cdots||\delta_{1m}||\delta_{2m} \ .$$

Finally, it computes $C'' \leftarrow \mathsf{Rerand}(\mathsf{ek}, C')$ and outputs $C \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C'')$ where Rerand is as defined later.

– $\mathsf{cMult}_{>d}(\mathsf{ek}', m, C)$: First, it computes $C' \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$ and parses it as

$$C' = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n} \ .$$

Then it sets $\alpha' \leftarrow m \boxdot \alpha$ and for each $k = 1, \dots, n$, sets $\beta'_{1k} \leftarrow m \boxdot \beta_{1k}$, $\beta'_{2k} = \beta_{2k}$, and puts $C'_0 = \alpha||\beta'_{11}||\beta'_{21}||\cdots||\beta'_{1n}||\beta'_{2n}$. Finally, it computes $C''_0 \leftarrow \mathsf{Rerand}(\mathsf{ek}, C'_0)$ and outputs $C_0 \leftarrow \mathsf{Enc}_{\mathcal{SE}}(\mathsf{K}, C''_0)$ where Rerand is as defined later.

- $\mathsf{Dec}'_{\le d}(\mathsf{sk}, C)$: It outputs $m \leftarrow \mathsf{Dec}(\mathsf{sk}, C)$.
- $\mathsf{Dec}'_{>d}(\mathsf{sk}, C)$: First, it computes $C' \leftarrow \mathsf{Dec}_{\mathcal{SE}}(\mathsf{K}, C)$, and parses it as $C' = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. It computes

$$m = \mathsf{Dec}(\mathsf{sk}, \alpha) + \sum_{i=1}^{n} \mathsf{Dec}(\mathsf{sk}, \beta_{1i}) \cdot \mathsf{Dec}(\mathsf{sk}, \beta_{2i})$$

and outputs m.

Now the algorithm Rerand used in the construction of Eval' is given as follows.

- $\mathsf{Rerand}(\mathsf{ek}, S)$: First, it parses the input S as $S = \alpha||\beta_{11}||\beta_{21}||\cdots||\beta_{1n}||\beta_{2n}$. For each $i = 1, 2$ and $j = 1, \dots, n$, it chooses $m_{ij} \leftarrow \mathcal{M}$ uniformly at random and sets $\gamma_{ij} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_{ij})$. Moreover, it sets $\beta'_{ij} \leftarrow \beta_{ij} \boxplus \gamma_{ij}$ and $\delta_j \leftarrow \mathsf{Enc}(\mathsf{pk}, -(m_{1j} \cdot m_{2j}))$. Then it sets

$$\epsilon_j \leftarrow \delta_j \boxplus ((-m_{2j}) \boxdot \beta_{1j}) \boxplus ((-m_{1j}) \boxdot \beta_{2j})$$

and $\alpha' \leftarrow \alpha \boxplus \epsilon_1 \boxplus \cdots \boxplus \epsilon_n$, and outputs $\alpha'||\beta'_{11}||\beta'_{21}||\cdots||\beta'_{1n}||\beta'_{2n}$.

Then we have the following result corresponding to Theorem 3. The proof is also similar to Theorem 3 and is omitted here.

Theorem 4. *Let $\mathcal{E}$ be a d-level KH-PKE scheme that is KH-CCA secure and leveled circuit private, and let $\mathcal{SE}$ be an SKE scheme that is IND-CPA secure and INT-CTXT secure. Then $\mathsf{CF}'(\mathcal{E}, \mathcal{SE})$ is KH-CCA secure and leveled circuit private.*

7 Conclusion

In this paper, first we showed that when extending the number of inputs for the homomorphic evaluation algorithm in a KH-PKE scheme, the KH-CCA security is not necessarily preserved; while the KH-CCA security is preserved when the original scheme also satisfies circuit privacy. The latter is applicable to the existing KH-PKE schemes in the literature. Secondly, we extended the Catalano–Fiore conversion (and its generalized version) to the case of KH-PKE schemes, which results in conversion from linearly KH-PKE schemes to two-level KH-PKE schemes and from d-level KH-PKE schemes to $2d$-level KH-PKE schemes. This conversion is applicable to KH-PKE schemes with various security assumptions such as the DDH and the DCR assumptions (for linearly KH-PKE schemes) and the SXDH assumption (for two-level KH-PKE schemes).

A drawback of our proposed conversion method (which is common to the original Catalano–Fiore conversion) is that in the resulting scheme, the homomorphic evaluation for higher-level ciphertexts increases the size of the ciphertext. In the original paper [14] of Catalano and Fiore, they proposed a primitive called

2S-DCED (two-server delegation of computation on encrypted data), and based on it, they constructed a two-server protocol for resolving the issue of non-compact ciphertexts. It is a future research topic to investigate possible extensions of their technique to our case of KH-PKE schemes. On the other hand, the original Catalano–Fiore conversion is known to preserve some more properties in addition to the IND-CPA security and circuit privacy. Studying similar properties in the case of our proposed conversion method is also a future research topic.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 19H01109, Japan, and by JST CREST Grant Number JPMJCR2113, Japan.

References

[1] N. Attrapadung, G. Hanaoka, S. Mitsunari, Y. Sakai, K. Shimizu, and T. Teruya, "Efficient two-level homomorphic encryption in prime-order bilinear groups and a fast implementation in WebAssembly," ASIACCS 2018, pp.685-697, 2018.

[2] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, " On the (im) possibility of obfuscating programs," CRYPTO 2001, pp.1-18, 2001.

[3] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," FOCS 1997, pp.394-403, 1997.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," CRYPTO 1998, pp.26-45, 1998.

[5] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," ASIACRYPT 2000, pp.531-545, 2000.

[6] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1," CRYPTO 1998, pp.1-12, 1998.

[7] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," CRYPTO 2004, vol. 3152, pp.41-55, 2004.

[8] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," TCC 2005, pp.325-341, 2005.

[9] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," CRYPTO 2012, pp.868-886, 2012.

[10] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(Leveled) fully homomorphic encryption without bootstrapping," ITCS 2012, pp.309-325, 2012.

[11] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from Ring-LWE and security for key dependent messages," CRYPTO 2011, pp.505-524, 2011.

[12] R. Canetti, S. Raghuraman, S. Richelson, and V. Vaikuntanathan, "Chosen-ciphertext secure fully homomorphic encryption," PKC 2017, pp.213-240, 2017.

[13] D. Catalano and D. Fiore, "Boosting linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data," Cryptology ePrint Archive, 2014/813, 2014.

[14] D. Catalano and D. Fiore, "Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data," ACM CCS 2015, pp.1518-1529, 2015.

[15] R. Cramer, R. Gennaro, and B. Schoenmakers, "A secure and optimally efficient multi-authority election scheme," EUROCRYPT 1997, vol. 1233 of LNCS, pp.103-118, 1997.

[16] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," EUROCRYPT 2010, pp.24-43, 2010.

[17] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," IEEE Transactions on Information Theory, vol. 31, no. 4, pp.469-472, 1985.

[18] K. Emura, "On the security of keyed-homomorphic PKE: preventing key recovery attacks and ciphertext validity attacks," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E104.A, no. 1, pp.310-314, 2021.

[19] K. Emura, G. Hanaoka, K. Nuida, G. Ohtake, T. Matsuda, and S. Yamada, "Chosen ciphertext secure keyed homomorphic public key cryptosystems," Designs, Codes and Cryptography, vol. 86, no. 8, pp.1623-1683, 2018.

[20] K. Emura, G. Hanaoka, G. Ohtake, T. Matsuda, and S. Yamada, "Chosen ciphertext secure keyed-homomorphic public-key encryption," PKC 2013, pp.32-50, 2013.

[21] K. Emura, T. Hayashi, N. Kunihiro, and J. Sakuma, "Mis-operation resistant searchable homomorphic encryption," ASIACCS 2017, pp.215-229, 2017.

[22] D. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," EUROCRYPT 2010, pp.44-61, 2010.

[23] C. Gentry, "Fully homomorphic encryption using ideal lattices," STOC 2009, pp.169-178, 2009.

[24] C. Gentry, A. Sahai, and B. Waters, "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based," CRYPTO 2013, pp.75-92, 2013.

[25] S. Goldwasser and S. Micali, "Probabilistic encryption," Journal of Computer and System Sciences, vol. 28, pp.270-299, 1984.

[26] G. Herold, J. Hesse, D. Hofheinz, C. Rafols, and A. Rupp, "Polynomial spaces: A new framework for composite-to-prime-order transformations," CRYPTO 2014, pp.261-279, 2014.

[27] C. Jutla and A. Roy, "Dual-system simulation-soundness with applications to UC-PAKE and more," ASIACRYPT 2015, pp.630-655, 2015.

[28] J. Lai, R. H. Deng, C. Ma, K. Sakurai, and J. Weng, "CCA-secure keyed-fully homomorphic encryption," PKC 2016, pp.70-98, 2016.

[29] B. Libert, T. Peters, M. Joye, and M. Yung, "Nonmalleability from malleability: Simulation-sound quasi-adaptive NIZK proofs and CCA2-secure encryption from homomorphic signatures," EUROCRYPT 2014, pp.514-532, 2014.

[30] Y. Maeda and K. Nuida, "Chosen ciphertext secure keyed two-level homomorphic encryption," Cryptology ePrint Archive, 2021/722, 2021.

[31] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," EUROCRYPT 1998, vol. 1403, pp.308-318, 1998.

[32] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," EUROCRYPT 1999, vol. 1592, pp.223-238, 1999.

[33] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," Foundations of Secure Computation, vol. 4, no.11, pp.169-180, 1978.

[34] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp.120-126, 1978.

[35] S. Sato, K. Emura, A. Takayasu, "Keyed-fully homomorphic encryption without indistinguishability obfuscation," Cryptology ePrint Archive, 2022/017, 2022.

[36] S. Sato, K. Emura, A. Takayasu, "Keyed-fully homomorphic encryption without indistinguishability obfuscation," ACNS 2022, to appear.

[37] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, 2004/332, 2004.