On Evaluating Circuits with Inputs Encrypted by Different Fully Homomorphic Encryption Schemes

Zhizhou Li and Ten H. Lai
Department of Computer Science and Engineering
The Ohio State University, Columbus, USA
{lizh, lai}@cse.ohio-state.edu

Abstract

We consider the problem of evaluating circuits whose inputs are encrypted with possibly different encryption schemes. Let $\mathcal{C}$ be any circuit with input $x_1, \ldots, x_t \in \{0,1\}$, and let $\mathcal{E}_i$, $1 \le i \le t$, be (possibly) different fully homomorphic encryption schemes, whose encryption algorithms are Enc_i. Suppose x_i is encrypted with $\mathcal{E}_i$ under a public key pk_i, say $c_i \leftarrow \mathsf{Enc}_i(pk_i, x_i)$. Is there any algorithm $\mathsf{Evaluate}$ such that $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_1, pk_1, c_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t, c_t\rangle)$ returns a ciphertext c that, once decrypted, equals $\mathcal{C}(x_1, \ldots, x_t)$? We propose a solution to this seemingly impossible problem with the number of different schemes and/or keys limited to a small value. Our result also provides a partial solution to the open problem of converting any FHE scheme to a multikey FHE scheme.

Keywords: Fully Homomorphic Encryption, Multi-Scheme FHE, Trivial Encryptions, Ciphertext Trees, Multiparty Computations.

1 Introduction

We are interested in the following question: given ciphertext $\psi_1, \ldots, \psi_t$ of different encryption schemes, is it possible to evaluate a function over these ciphertexts? More precisely, for $1 \le i \le t$, suppose message x_i is encrypted to ψ_i by encryption scheme $\mathcal{E}_i$ under key pk_i; we are interested in an algorithm $\mathsf{Evaluate}$ that given any circuit $\mathcal{C}$, ciphertexts $\psi_1, \ldots, \psi_t$, and their respective encryption methods $\langle \mathcal{E}_i, pk_i\rangle$, will produce a ciphertext ψ which, once decrypted, gives $\mathcal{C}(x_1, \ldots, x_t)$.

This problem is a generation of Fully Homomorphic Encryption (FHE). An FHE scheme is an encryption scheme that allows one to evaluate any circuit with FHE-ciphertexts. Gentry [Gen09a, Gen09b] invented the first such scheme, and since then several other FHE schemes have been proposed (e.g., [BV11a, SV10, vDGHV10]). An FHE scheme consists of four algorithms, $\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}\rangle$, where Eval is an algorithm that given a circuit $\mathcal{C}$, a public key pk and encryptions of $x_1, \ldots, x_t$, evaluates $\mathcal{C}(x_1, \ldots, x_t)$ using only x_i's ciphertexts. Such schemes are *single-keyed*: all $x_1, \ldots, x_t$ are encrypted under a single key. However, a single key FHE does not fit for *Multiparty Computation* (MPC).

Recently, a new concept called Multikey Fully Homomorphic Encryption (MK-FHE) has been proposed [LATV12]. By definition, the evaluation algorithm of an MK-FHE scheme can evaluate any circuit on ciphertexts encrypted under different keys. Such a scheme can be used to construct MPC protocols: each party encrypts its secret data using its own key; then they use the multikey evaluation algorithm to evaluate a circuit over the ciphertexts; the resulting ciphertext can be decrypted to the wanted result of the multiparty computation. In [LATV12], the authors show how to convert an NTRU-based FHE scheme into an *N-key FHE scheme* (which is used for *on-the-fly* MPC), and they point out that several other schemes [BV11a, Gen09a, SV10, vDGHV10] can also be converted. An interesting open problem raised in [LATV12] is whether *every* FHE scheme can be made multikey.

We are interested in a more challenging problem called *Multi-Scheme* FHE. Let $\mathcal{C}$ be any circuit with input $x_1, \ldots, x_t$, and let $\mathcal{E}_i = (\mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i, \mathsf{Eval}_i)$, $1 \leq i \leq t$, be possibly *different* fully homomorphic encryption schemes, whose encryption algorithms are Enc_i. Suppose x_i is encrypted with $\mathcal{E}_i$ under public key pk_i, say $c_i \leftarrow \mathsf{Enc}_i(pk_i, x_i)$. Is it possible to homomorphically evaluate $\mathcal{C}$ on ciphertexts $c_1, \ldots, c_t$? That is, is there any algorithm $\mathsf{Evaluate}$ such that $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_i, pk_i, c_i \rangle_{1 \leq i \leq t})$ returns a ciphertext c that, once decrypted, equals $\mathcal{C}(x_1, \ldots, x_t)$?

This problem is of theoretical as well as practical interest. When applied to (cloud assisted) multiparty computation, our approach is "stateless": no setup phase is required, a participant only needs to upload a ciphertext once and she will not be asked for more information during the computation. Compare to the *on-the-fly* MPC [LATV12], our solution allows a cloud to perform *offline* MPC on any FHE ciphertexts stored on it, (but it still has to ask the ciphertext owners to jointly decrypt the result). In multiparty computation, the participating parties may for various reasons fail to agree on using a same FHE scheme. Participants from different countries, different companies, or different political parties may have their own preferred FHE schemes. In such situations, a multi-scheme $\mathsf{Evaluate}$ will come in handy.

1.1 Related Work

Much research attention has been paid to utilizing FHE in Multiparty Computation Problems. The paper [vDJ10] first shows that it is impossible to construct an MPC protocol in which a subset of the participants can always decode the resulting ciphertext, because such a protocol leads to a construction of program obfuscator, which is proved impossible in [BGI+01].

Literature [AJLA+12] uses the key-homomorphism property of some FHE schemes to make possible multiparty computation. An FHE scheme is key-homomorphic if $pk = f(pk_1, \ldots, pk_t)$ is a valid public key, then there exists f' such that $sk = f'(sk_1, \ldots, sk_t)$ is the corresponding decryption key of pk. The joint public key pk is used for encryption and evaluation. A secure protocol is needed to decrypt the result using secret keys $sk_1, \ldots, sk_t$.

In literature [LATV12], the authors define the MK-FHE scheme as a family of encryption scheme $\{\mathcal{E}^{(N)}\}_{N>0}$ where each scheme can handle up to N distinct keys. The *on-the-fly* MPC protocol based on N-key FHE is then parameterized by N: the parties first jointly determines $\mathcal{E}^{(N)}$ by agreeing on a parameter set $q = q(N)$, then they independently generate public key $(pk, sk) \leftarrow \mathsf{KeyGen}(q)$, encrypt their own messages and use the Eval algorithm to evaluate a circuit. If more parties want to join in the computation, they have to agree on a new parameter set. Also, a secure decryption protocol is needed to decrypt the result.

1.2 Our Results

To the best of our knowledge, this paper is the first to consider multi-scheme FHE. We focus on the problem of evaluating circuits with differently encrypted FHE ciphertexts. We design an algorithm $\mathsf{Evaluate}$ to solve the problem, as stated below:

Theorem 1 (Main Result). *Suppose $\mathcal{E}_i = (\mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i, \mathsf{Eval}_i)$ $(1 \leq i \leq t)$ is a FHE scheme with plaintext space $\{0,1\}$, pk_i is a public key generated by Enc_i; message $x_i \in \{0,1\}$, and $c_i \leftarrow \mathsf{Enc}_i(pk_i, x_i)$ is an encryption of x_i with scheme $\mathcal{E}_i$. There exists an algorithm $\mathsf{Evaluate}$ such that given any circuit $\mathcal{C}$ and ciphertexts $c_1, \ldots, c_t$, $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_i, pk_i, c_i \rangle_{1 \leq i \leq t})$, it returns a ciphertext c which can be decrypted to $\mathcal{C}(x_1, \ldots, x_t)$.*

Our proof of the theorem is constructive: we propose such an algorithm $\mathsf{Evaluate}$, or more precisely, we describe how to construct such an algorithm. A key concept to be introduced to the design of the algorithm is *ciphertext tree*. A ciphertext tree is a representation of a ciphertext that has been encrypted

by multiple schemes/keys. We will formally show that FHE schemes (perhaps with slight changes) are well defined for the tree structures.

The complexity of Evaluate (in terms of the output ciphertext size) is polynomial in the lengths of the input ciphertexts, which implies that it is polynomial in λ, the security parameters of the FHE schemes; however, it could be exponential in t, the number of different $\langle \mathcal{E}_i, pk_i \rangle$ pairs. The reason is that the output of the algorithm is a ciphertext tree, and we know that a fully grown tree has size exponential in its depth (for a ciphertext tree, the depth is t). As a result, the time complexity of algorithm Evaluate is exponential in t in the worst case. It is a open problem whether we can reduce the the output tree to polynomial size. Some partial results are discussed in Section 6.3. Despite this limitation, we believe that our scheme is still useful for many applications where the the number of participants is limited.

Another interesting open problem is how to securely decrypt ciphertexts ψ produced by the multi-scheme $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_i, pk_i, c_i \rangle_{1 \le i \le t})$. (This is also still an open problem for MK-FHE.) In light of recent impossibility results in [vDJ10], it is not hard to see that encrypting ψ entails all the secret keys sk_i corresponding to pk_i. With ψ produced by our proposed Evaluate, a straight way to decrypt it is by decrypting it with decryption algorithm Dec_i and secret key sk_i one by one for i from t to 1. (This will become clear shortly.) This way of decrypting works for some applications, e.g., no decryption key owners are corrupted or colluded against others. Otherwise, a more secure decryption method is needed.

The idea behind our algorithm is simple. Observe that if the evaluation circuit Eval of an FHE can handle any circuit, it can also handle the evaluation circuit of another scheme, or even itself. We call this "evaluation of Eval" the nested evaluation algorithm (Section 6). Intuitively, Eval takes ciphertexts as input; evaluating this Eval requires the encryption of its input, that is, "encryption of ciphertexts." We call them *multiple encryptions*, denoted $\mathsf{Enc}_{pk_n} \ldots \mathsf{Enc}_{pk_1}(\cdot)$. We employ tree structure to succinctly and precisely define multiple encryption, then we show the slightly modified FHE scheme $\mathcal{E}$ is well defined for the *ciphertext trees* (Section 5).

The nested evaluation algorithm takes multiple encryptions as input, however, the Evaluate is only given regular ciphertexts. To convert a regular ciphertext to a multiple encryption (without additional information from the original encryptor), we introduce the concept of *trivial encryptions*, namely, "a message is its own encryption" under the context of FHE (Section 4). Then we show that a regular ciphertext can be made a valid ciphertext tree using trivial encryption (Section 5.3). Finally, evaluating a circuit on these trees yields a ciphertext tree, the desired encryption of the circuit's result (Section 6.3).

2 Preliminary

A homomorphic encryption scheme $\mathcal{E}$ consists of four algorithms, KeyGen, Enc, Dec, and Eval, where KeyGen, Enc, Dec are like that of a regular encryption scheme, and Eval is an *evaluation* algorithm that takes as input a circuit $\mathcal{C}$, a public key pk, a tuple of ciphertexts, $\langle \psi_i \rangle_{i \in I}$ under pk, and outputs a ciphertext under pk. That is,

$$\psi \leftarrow \mathsf{Eval}(pk, \mathcal{C}, \psi_1, \ldots, \psi_t).$$

Eval is said to be *correct* for a circuit collection $\mathbb{C}$ if for any circuit $\mathcal{C} \in \mathbb{C}$, any key pair (pk, sk) output by KeyGen, and any plaintexts $x_1, \ldots, x_t$, it holds that

$$\begin{aligned} &\text{If } \psi_i \leftarrow \mathsf{Enc}_{pk}(x_i) \text{ for } i = 1, \ldots, t \text{ and } \psi \leftarrow \mathsf{Eval}(pk, \mathcal{C}, x_1, \ldots, x_t) \\ &\qquad \text{then } \mathsf{Dec}_{sk}(\psi) = \mathcal{C}(x_1, \ldots, x_t) \end{aligned} \tag{1}$$

Definition 1. (Homomorphic Encryption) *An encryption scheme $\mathcal{E}$ is said to be* homomorphic *for a collection of circuits $\mathbb{C}$ if it has an evaluation algorithm* Eval *that works correctly for circuits in $\mathbb{C}$.*

Definition 2. (Fully Homomorphic Encryption) *An encryption scheme $\mathcal{E}$ fully homomorphic if it is a homomorphic encryption scheme with respect to* all *circuits.*

3 Multi-Scheme FHE

Loosely speaking, in the definition of multikey FHE [LATV12] (see Definition 14 in the appendix), if we allow key pairs (pk_i, sk_i) to be associated with different FHE schemes, then we have a multi-scheme FHE scheme.

Definition 3. (Multi-Scheme FHE or MS-FHE) A *multi-scheme fully homomorphic encryption scheme* is a set of (single-key) fully homomorphic encryption schemes $\mathcal{E}^{(i)} = \langle \mathsf{KeyGen}^{(i)}, \mathsf{Enc}^{(i)}, \mathsf{Dec}^{(i)}, \mathsf{Eval}^{(i)} \rangle$ together with two additional algorithms Decrypt and Evaluate:

- Decrypt is a deterministic algorithm that, on input a ciphertext ψ and scheme-key pairs $\langle s_i, sk_i \rangle$, outputs a plaintext:
$$x \leftarrow \mathsf{Decrypt}(\langle s_1, sk_1 \rangle, \ldots, \langle s_t, sk_t \rangle, \psi)$$
where s_i indicates scheme $\mathcal{E}^{(s_i)}$ and sk_i is a secret key generated by $\mathsf{KeyGen}^{(s_i)}$.

- Evaluate: Takes as input a circuit $\mathcal{C}$, a number of 3-tuples $\langle s_i, pk_i, \psi_i \rangle$, $1 \leq i \leq t$, where s_i is an index indicating scheme $\mathcal{E}^{(s_i)}$, pk_i is a public key generated by $\mathsf{KeyGen}^{(s_i)}$, and ψ_i is a ciphertext under encryption key pk_i and scheme $\mathcal{E}^{(s_i)}$, i.e., $\psi_i \leftarrow \mathsf{Enc}^{(s_i)}(pk_i, x_i)$ for some plaintext x_i. The output of Evaluate is a ciphertext:
$$\psi := \mathsf{Evaluate}(\mathcal{C}, \langle s_1, pk_1, \psi_1 \rangle, \ldots, \langle s_t, pk_t, \psi_t \rangle).$$

It is required that the following conditions hold:

- **Correctness:** The scheme is *correct* if for any $t > 0$, any t-input circuit $\mathcal{C}$, any t key pairs $\langle pk_i, sk_i \rangle$ generated by $\mathsf{KeyGen}^{(s_i)}(1^\lambda)$, and any t ciphertexts ψ_i, where $1 \leq i \leq t$, it holds that
$$\begin{gathered}\text{If } \psi_i \leftarrow \mathsf{Enc}^{(s_i)}(pk_i, x_i) \text{ and } \psi := \mathsf{Evaluate}(\mathcal{C}, \langle s_1, pk_1, \psi_1 \rangle, \ldots, \langle s_t, pk_t, \psi_t \rangle) \\ \text{then } \mathcal{C}(x_1, \ldots, x_t) = \mathsf{Decrypt}(\langle s_1, sk_1 \rangle, \ldots, \langle s_t, sk_t \rangle, \psi).\end{gathered}$$

- **Compactness in security parameter λ:** The MS-FHE scheme is *compact in security parameter* λ if there are polynomials d and e such that, for every value of the security parameter λ, Decrypt and Evaluate can be expressed as circuits of sizes at most $d(\lambda)$ and $e(\lambda)$, respectively. (The underlying FHE schemes $\mathcal{E}^{(i)}$ are implicitly assumed to be compact.)

The following compactness condition is highly desirable for MS-FHE. We did not include it in Definition 3 as a requirement because it is still yet unknown whether this condition can be met by any MS-FHE scheme. For more discussion on schemes that are compact in t, see section 6.3.

Definition 4. (Compactness in t) Let t be the number of different pairs (s_i, pk_i) that are used in evaluating a circuit. An MS-FHE scheme is *compact in number of schemes* if there are polynomials d and e such that, for every value of t, Decrypt and Evaluate can be expressed as circuits of sizes at most $d(t)$ and $e(t)$, respectively.

Remark 1. If a *same* FHE scheme is used (i.e., all s_i are equal, but pk_i may be different), then multi-scheme FHE becomes multikey FHE as defined in [LATV12] (or Definition 14 in the appendix). If in addition all key pairs (pk_i, sk_i) are the same, then it becomes the classic (single-key) FHE as defined in [Gen09b].

4 Trivial Encryptions in FHE

We first show that given any FHE scheme $\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval} \rangle$, we can modify it such that 0 and 1 are valid encryptions of 0 and 1, respectively. We call the ciphertexts $0, 1$ the trivial encryptions and an FHE with trivial encryption TE-FHE.

Definition 5 (Trivial Encryption). *An FHE scheme is said to have the* trivial encryption property *if for every key pair (pk, sk) output by the key generation algorithm, it holds that $b \in \{0,1\}$ is a valid ciphertext of b, i.e., $\mathsf{Dec}(sk, b) = b$.*

Example 1. In [Gen09a], Gentry proposed to use *weakened ciphertexts* to achieve better efficiency in computation. Trivial encryptions are the extreme case of weakened ciphertexts. Roughly speaking, FHE schemes based on ideal lattices tend to have trivial encryptions. The "over integers" FHE [vDGHV10] also has trivial encryptions: at a very high level, the scheme encrypts a bit b as $pq + 2n + b$ where the key p is a large integer, q a randomly sampled number, n a random noise. When $q = 0$ and $n = 0$, the resulting ciphertext b is an encryption of message b under any valid key p.

Although it is not known whether all FHEs have the trivial encryption property, we can convert any FHE to a TE-FHE scheme by adding 0 and 1 to the ciphertext space.

Lemma 1. *An FHE scheme $\mathcal{E}$ with plaintext space $\{0,1\}$ and ciphertext space $\mathcal{R} \subseteq \{0,1\}^+$ can be converted into an FHE scheme $\mathcal{E}'$ having the trivial encryption property. Furthermore, the resulting scheme $\mathcal{E}'$ has ciphertext indistinguishability if $\mathcal{E}$ does.*

Proof. (Sketch only.) The full proof is provided in Section A.1. Let $\mathcal{E}$ be an FHE scheme with plaintext space $\{0,1\}$ and ciphertext space $\mathcal{R} \subseteq \{0,1\}^+$ ($\{0,1\}^+$ denotes the set of all non-empty bit strings). Let $f : \mathcal{R} \mapsto \{0,1\}^{\geq 2}$ be a one-to-one function that maps every ciphertext to a string of length at least 2. Set the new ciphertext space as $f(\mathcal{R}) \cup \{0,1\}$ and set 0, 1 the trivial ciphertext of 0 and 1, respectively (need to change the algorithms in $\mathcal{E}$ accordingly), thus yielding a TE-FHE $\mathcal{E}'$. □

5 Multiple Encryptions

The intuition of our multi-key evaluation algorithm is that, if the input encryptions are of the same *structure* $\mathsf{Enc}_{pk_n} \dots \mathsf{Enc}_{pk_1}(\cdot)$, then we can use a nested evaluation algorithm $\mathsf{Eval}_{pk_n} \dots \mathsf{Eval}_{pk_1}(\cdot)$ to produce a ciphertext of the same structure, which then can be decrypted by $\mathsf{Dec}_{sk_1} \dots \mathsf{Dec}_{sk_{n-1}} \mathsf{Dec}_{sk_n}(\cdot)$ to a plaintext. In the following discussion, denote $\mathsf{Enc}_{pk_i} = \mathsf{Enc}_i(pk_i, \cdot)$ the encryption scheme of $\mathcal{E}_i$ with key pk_i. Dec_{sk_i} and Eval_{pk_i} are similarly defined.

5.1 Multiple Encryptions and Ciphertext Trees

This section introduces the notion of *multiple encryption* and defines two corresponding notations, $\mathsf{Enc}_{pk_n} \dots \mathsf{Enc}_{pk_2} \mathsf{Enc}_{pk_1}(\cdot)$ and $\mathsf{Dec}_{sk_1} \dots \mathsf{Dec}_{sk_{n-1}} \mathsf{Dec}_{sk_n}(\cdot)$. Roughly speaking, multiple encryption is "encrypting a message multiple times". For a plaintext $b \in \{0,1\}$, the multiple encryption of b encrypted under FHE scheme $\mathcal{E}_i$ by keys pk_i for $i = 1, \dots, n$ is the "ciphertext" resulting from the following procedure. Envision a bit string ψ initially consisting of a single bit b. For i from 1 to n, we repeatedly encrypt every bit $x \in \psi$ using scheme $\mathcal{E}_i$ and key pk_i, and substitute that bit x with its ciphertext $\mathsf{Enc}_{pk_i}(x)$. (We denote by Enc_{pk_i} the encryption algorithm of $\mathcal{E}_i$ with key pk_i; similarly, Dec_{sk_i} and Eval_{pk_i} denote the decryption and evaluation of scheme $\mathcal{E}_i$ using key sk_i or pk_i, respectively.) Then we obtain a multi-encryption ciphertext of b under keys $pk_1, \dots, pk_n$ — namely, $\mathsf{Enc}_{pk_n} \dots \mathsf{Enc}_{pk_2} \mathsf{Enc}_{pk_1}(b)$. (This is not a formal definition, which will follow soon.) For instance, suppose b is encrypted by pk_1 into xyz; and

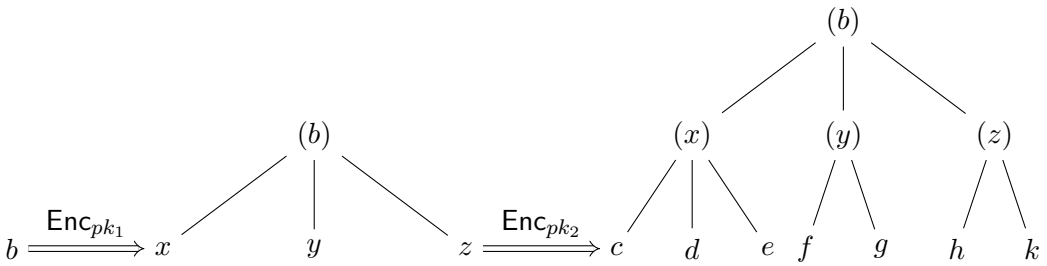

Figure 1: (Multiple) Encryptions Viewed as Ciphertext Trees. From left to right: plaintext b as a single node tree; ciphertext $\mathsf{Enc}_{pk_1}(b)$ as a tree; multiple encryption $\mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(b)$ as a tree. A node in paratheses means this node is unlabeled.

x, y, z are encrypted by pk_2 into cd, efg, hk, respectively; then we have $\psi = cdefghk \leftarrow \mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(b)$, where $b, c, d, e, f, g, h, k, x, y, z \in \{0, 1\}$.

Now, if ψ is a multi-encryption ciphertext of b under different schemes and keys $\langle \mathcal{E}_1, pk_1 \rangle, \ldots, \langle \mathcal{E}_n, pk_n \rangle$, it is naturally required that decrypting ψ in turn with keys $sk_n, \ldots, sk_1$ will yield b. That is, if $\psi \leftarrow \mathsf{Enc}_{pk_n} \ldots \mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(b)$, then we should have $\mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_{n-1}}\mathsf{Dec}_{sk_n}(\psi) = b$, or the scheme will not be correct. For instance, in the above example, we want it to hold that

$$\mathsf{Dec}_{sk_1}\mathsf{Dec}_{sk_2}(cdefghk) = \mathsf{Dec}_{sk_1}\Big(\mathsf{Dec}_{sk_2}(cde)\mathsf{Dec}_{sk_2}(fg)\mathsf{Dec}_{sk_2}(hk)\Big) = \mathsf{Dec}_{sk_1}(xyz) = b.$$

This example illustrates that the ciphertext $cdefghk$ should have some structure such as $(cde)(fg)(hk)$. We choose trees to represent such structures.

Definition 6. [**Ciphertext Tree**] A *ciphertext tree* is a rooted tree with two properties: (1) all leaves are at the same level and (2) each leaf is labeled by a bit (0 or 1) while internal nodes are not labeled.

Given a scheme $\mathcal{E}$, we extend the encryption scheme Enc into a scheme $\widetilde{\mathsf{Enc}}$ whose input is a ciphertext tree.

Definition 7. [Encryption of Ciphertext Tree] If T is a ciphertext tree, we denote by $\widetilde{\mathsf{Enc}}_{pk}(T)$ a ciphertext tree that results from T by, for each leaf x of T, adding $t_x = |\mathsf{Enc}_{pk}(x)|$ children to x and labeling them by the t_x bits of $\mathsf{Enc}_{pk}(x)$, one bit per child, from left to right. The original leaves of T now become internal nodes and unlabeled.

For example, as seen in Figure 1, a plaintext b is a single node tree (on the left); ciphertext $\mathsf{Enc}_{pk_1}(b) = xyz$ is again a tree $\widetilde{\mathsf{Enc}}_{pk_1}(b)$, with root b (unlabeled) and leaves x, y and z. Encrypting the ciphertext tree $\widetilde{\mathsf{Enc}}_{pk_1}(b)$ is encrypting the leaves, e.g., $\mathsf{Enc}_{pk_2}(x) = cde$, attach c, d, e to x then unlabel x.

Note: We emphasize that encrypting a tree T is not hiding the tree structure of T; instead, only the labels of leaves are encrypted. Thus, from $\widetilde{\mathsf{Enc}}_{pk}(T)$, one can tell the tree structure of T without decrypting it.

In practice, a tree is encoded as a bit string. Let $\mathsf{Encoding}$ be an encoding scheme that encodes a ciphertext tree T into a string. Suppose $S := \mathsf{Encoding}(T)$. Note that encrypting every bit in S does *not* yield the encoding of $\widetilde{\mathsf{Enc}}_{pk}(T)$. Instead, we define

$$\widetilde{\mathsf{Enc}}_{pk}(S) \triangleq \mathsf{Encoding}(\widetilde{\mathsf{Enc}}_{pk}(T)).$$

We employ the tree structure to succinctly and precisely define multiple encryption.

Definition 8. [Multiple Encryption] Given schemes and their keys $\langle \mathcal{E}_1, pk_1\rangle, \ldots, \langle \mathcal{E}_n, pk_n\rangle$, and a plaintext $b \in \{0,1\}$, we view b as a single node ciphertext tree labeled by b, and recursively define

$$\widetilde{\mathsf{Enc}}_{pk_n}\widetilde{\mathsf{Enc}}_{pk_{n-1}}\ldots\widetilde{\mathsf{Enc}}_{pk_1}(b) \triangleq \widetilde{\mathsf{Enc}}_{pk_n}\big(\widetilde{\mathsf{Enc}}_{pk_{n-1}}\ldots\widetilde{\mathsf{Enc}}_{pk_1}(b)\big).$$

However, for simplicity of notation (with perhaps a little abuse), we sometimes write $\widetilde{\mathsf{Enc}}_{pk_n}\widetilde{\mathsf{Enc}}_{pk_{n-1}}\ldots\widetilde{\mathsf{Enc}}_{pk_1}(b)$ simply as $\mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\ldots\mathsf{Enc}_{pk_1}(b)$.

Thus, a plaintext b is equivalent to a single-node tree; an ordinary ciphertext $\mathsf{Enc}_{pk}(b)$ may be viewed as a tree of depth 1; and the n-key encryption of b, $\mathsf{Enc}_{pk_n}\ldots\mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(b)$ is a ciphertext tree of depth n.

Since $\mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\cdots\mathsf{Enc}_{pk_1}(\cdot)$ is a randomized algorithm, $\mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\cdots\mathsf{Enc}_{pk_1}(b)$ is not unique. We introduce the following notation for convenience of exposition.

Definition 9. Denote by $\mathbf{T}_{pk_1,\ldots,pk_n}$ the set of all ciphertext trees that may be output by the multiple encryption algorithm $\mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\ldots\mathsf{Enc}_{pk_1}(\cdot)$. That is,

$$\mathbf{T}_{pk_1,\ldots,pk_n} \triangleq \big\{T : T \leftarrow \mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\cdots\mathsf{Enc}_{pk_1}(b), b \in \{0,1\}\big\}.$$

Ciphertext trees in the same $\mathbf{T}_{pk_1,\ldots,pk_n}$ are said to have the *same structure*.

Decrypting a ciphertext tree T is the reverse of encrypting a tree. (See Fig. 1 for illustration.) Given a ciphertext tree $T \in \mathbf{T}_{pk_1,\ldots,pk_n}$ and decryption key sk_n, $\widetilde{\mathsf{Dec}}(sk_n, T)$ yields a tree in $\mathbf{T}_{pk_1,\ldots,pk_{n-1}}$. Recursively define

$$\widetilde{\mathsf{Dec}}_{sk_1}\widetilde{\mathsf{Dec}}_{sk_2}\ldots\widetilde{\mathsf{Dec}}_{sk_n}(T) \triangleq \widetilde{\mathsf{Dec}}_{sk_1}\ldots\widetilde{\mathsf{Dec}}_{sk_{n-1}}\big(\widetilde{\mathsf{Dec}}_{sk_n}(T)\big). \tag{2}$$

Again, we write $\widetilde{\mathsf{Dec}}_{sk_1}\ldots\widetilde{\mathsf{Dec}}_{sk_n}$ simply as $\mathsf{Dec}_{sk_1}\ldots\mathsf{Dec}_{sk_n}$. It is clear that if all $\mathcal{E}_i = \langle \mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i, \mathsf{Eval}_i\rangle$ are correct encryption schemes with plaintext space $\{0,1\}$, then

$$\mathsf{Dec}_{sk_1}\mathsf{Dec}_{sk_2}\ldots\mathsf{Dec}_{sk_n}\big(\mathsf{Enc}_{pk_n}\mathsf{Enc}_{pk_{n-1}}\ldots\mathsf{Enc}_{pk_1}(b)\big) = b. \tag{3}$$

5.2 Evaluating Circuits with Ciphertext Trees as Inputs

Under some circumstances, Eval can be modified to accept ciphertext trees (as opposed to pure ciphertexts) as input and produces a ciphertext tree of the same structure as output. We first define the scope of circuits Γ_i that can be evaluated by such an algorithm. In the following definition, we view a ciphertext tree as a directed graph with each leaf labeled by a bit.

Definition 10. Denote by Γ_i ($i \geq 1$) the set of circuits $\mathcal{C}$ for which the following conditions hold:

1. the input and output of $\mathcal{C}$ are depth i ciphertext trees of the same structure (for example, $T := \mathcal{C}(T_1, \ldots, T_s)$, with $T, T_1, \ldots, T_s \in \mathbf{T}_{pk_1,\ldots,pk_i}$);
2. the *graph* (or the *topology*, the nodes and edges of the ciphertext tree without labels) of output T are determined by the graphs of $T_1, \ldots, T_s$ (without knowing the latter's labels of leaves);
3. T's labels of leaves are computed from the labels of leaves of $T_1, \ldots, T_s$.

Define Γ_0 as the set of circuits whose inputs and output are single-node trees (of depth 0).

We now derive from Eval an evaluation algorithm $\widetilde{\mathsf{Eval}}$ that can evaluate $\mathcal{C} \in \Gamma_i$. Specifically, suppose $T = \mathcal{C}(T_1, \ldots, T_s)$, where $\mathcal{C} \in \Gamma_i$ and, without loss of generality, $T, T_1, \ldots, T_s \in \mathbf{T}_{pk_1,\ldots,pk_i}$. Let $T'_j \leftarrow \mathsf{Enc}(pk, T_j)$ be an encryption of T_j under key pk, where $1 \le j \le s$. Then $\widetilde{\mathsf{Eval}}$ works as follows.

Algorithm/Circuit $\widetilde{\mathsf{Eval}}$

input: $\mathcal{C} \in \Gamma_i$; pk; $T'_1, \ldots, T'_s \in \mathbf{T}_{pk_1,\ldots,pk_i,pk}$, where $T'_j \leftarrow \widetilde{\mathsf{Enc}}(pk, T_j)$ for some $T_j \in \mathbf{T}_{pk_1,\ldots,pk_i}$[1].

output: a ciphertext T' such that $\widetilde{\mathsf{Dec}}_{pk}(T') = \mathcal{C}(T_1, \ldots, T_s)$.

1. Compute from T'_j the graph (tree structure, without labels) of T_j and denote it by $G(T_j)$, $1 \le j \le s$.
2. Determine $G(T)$, the graph of T. (This can be done because of property 2 of Def. 10. By Def. 7, $G(T_j)$ is not hidden in T'_j. It is able to follow the steps of $\mathcal{C}$ to reproduce the $G(T)$ given the graphs of T_j's.)
3. Expand T as follows. For each leaf node z of T, construct the subcircuit $\mathcal{C}_z$ of $\mathcal{C}$ that computes the label of z. The inputs of $\mathcal{C}_z$ involve only leaves of $T_1, \ldots, T_s$, say nodes $y_1, \ldots, y_u$ (by property 3 of Definition 10). That is, $z = \mathcal{C}_z(y_1, \ldots, y_u)$.[2] Call $\psi_z := \mathsf{Eval}(\mathcal{C}_z, pk, \psi_1, \ldots, \psi_u)$ to evaluate $\mathcal{C}_z$, where $\psi_j = \mathsf{Enc}_{pk}(y_j)$ are available in $T'_1, \ldots, T'_s$. Attach the resulting ciphertext ψ_z bit-by-bit to node z as its children.
4. Output the resulting ciphertext tree as T'.

It is evident that the ciphertext tree T' generated above is an encryption of T (under pk); decrypting the leaves of T' then labeling the parents with the resulting plaintexts will produce T. We state this as a lemma for ease of reference.

Lemma 2. *The ciphertext T' output by $\widetilde{\mathsf{Eval}}$ satisfies $\widetilde{\mathsf{Dec}}_{sk}(T') = T \triangleq \mathcal{C}(T_1, \ldots, T_s)$.*

For a given circuit $\mathcal{C}$ and key pk, define

$$\widetilde{\mathsf{Eval}}[\mathcal{C}, pk](T'_1, \ldots, T'_s) \triangleq \widetilde{\mathsf{Eval}}(\mathcal{C}, pk, T'_1, \ldots, T'_s). \tag{4}$$

$\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ is an algorithm that takes ciphertext trees as input and produces a ciphertext tree as output. If $\mathcal{C} \in \Gamma_i$, the input and output ciphertext trees are of the same structure. For example, if $\mathcal{C}$ manipulates ciphertext trees in $\mathbf{T}_{pk_1,\ldots,pk_i}$, $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ will operate on $\mathbf{T}_{pk_1,\ldots,pk_i,pk}$. We will have more discussions on this in Definition 13.

Now we will show $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk] \in \Gamma_{i+1}$ for $\mathcal{C} \in \Gamma_i$ if the regular Eval outputs fixed length ciphertexts. This lemma will play an important role in subsequent sections.

Definition 11. The regular evaluation algorithm Eval outputs fixed length ciphertexts if given a circuit $\mathcal{C}$, a public key pk and any valid ciphertexts $\psi_1, \ldots, \psi_v$, $\psi = \mathsf{Eval}(\mathcal{C}, pk, \psi_1, \ldots, \psi_v)$, it holds that $|\psi| = \mathcal{L}^{\mathcal{C},pk}(|\psi_1|, \ldots, |\psi_v|)$ for some polynomial $\mathcal{L}^{\mathcal{C},pk}$ (in which we may omit the pk for simplicity).

Remark 2. In the above definition, we assume that the regular Eval (given $\mathcal{C}$ and pk) is an algorithm that if the lengths of inputs are fixed, the length of output is fixed. If given a circuit, a public key and the lengths of all input ciphertexts, the regular Eval can be expressed as a boolean circuit that has $\sum_{j=1}^{v} |\psi_j|$ input lines and $|\psi| = \mathcal{L}^{\mathcal{C}}(|\psi_1|, \ldots, |\psi_v|)$ output lines. If the output length is bounded by some polynomial, e.g., $|\psi| \le \mathcal{L}^{\mathcal{C}}(|\psi_1|, \ldots, |\psi_v|)$, we may also satisfy the assumption by, for instance, adding padding bits to ψ until its length reaches $\mathcal{L}^{\mathcal{C}}(|\psi_1|, \ldots, |\psi_v|)$ and/or by changing the encoding of the ciphertexts.

[1] When $i = 0$, $\mathbf{T}_{pk_1,\ldots,pk_i}$ is defined to consist of the single-node trees.

[2] Here we use $z, y_1, \ldots, y_u$ to denote both *nodes* and their *labels*.

Lemma 3. *Let $\mathcal{C} \in \Gamma_i$ be a circuit. Assume that a regular evaluation* Eval *outputs fixed length ciphertexts, then* $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ *(which is derived from* Eval*) is in* Γ_{i+1}.

Proof. The full proof is provided in Section A.2. It is worth mentioning that Γ_i for $i \geq 1$ are not empty. Notice that Γ_0 is the set of all circuits, we have $\widetilde{\mathsf{Eval}}_1[\mathcal{C}, pk_1] \in \Gamma_1$ ($\widetilde{\mathsf{Eval}}_1$ is derived from Eval_1), so Γ_1 is not empty. In particular, define circuit $\mathsf{Eval}^i := \widetilde{\mathsf{Eval}}_i[\mathcal{C}, pk_i] \in \Gamma_i$ for some $\mathcal{C} \in \Gamma_{i-1}$, we have $\widetilde{\mathsf{Eval}}_{i+1}[\mathsf{Eval}^i, pk_{i+1}] \in \Gamma_{i+1}$. The idea of "evaluating $\widetilde{\mathsf{Eval}}$" is used in constructing the nested evaluation in Section 6.2. □

5.3 Making Multiple Encryption From Single Encryption

This section shows how to view an ordinary single-key ciphertext as a multikey ciphertext tree. Let $\langle \mathcal{E}_1, pk_1 \rangle, \ldots, \langle \mathcal{E}_t, pk_t \rangle$ be TE-FHE schemes and their public keys, and $\mathbf{T}_{pk_1,\ldots,pk_t}$ the set of ciphertext trees generated by those pairs. Consider an ordinary encryption of a plaintext b under key pk_i, say, $\psi_i \leftarrow \mathsf{Enc}_{pk_i}(b)$ where $1 \leq i \leq t$. (Note: ψ_i is a bit string.) We show how to construct a ciphertext tree $T(\psi_i) \in \mathbf{T}_{pk_1,\ldots,pk_t}$ that naturally corresponds to ψ_i. This way, an ordinary ciphertext can be uniquely translated into a ciphertext tree. We first explain the translation process by an example.

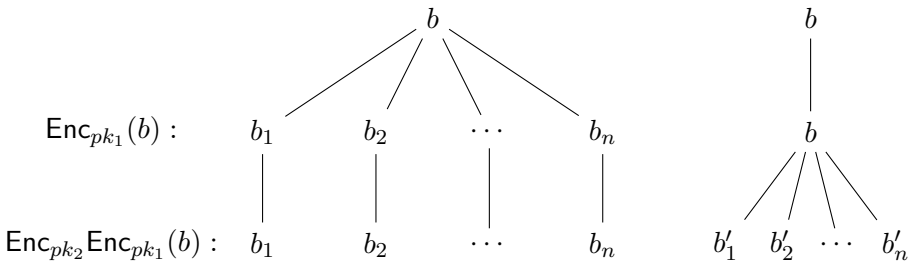

Figure 2: Examples for Making Single Encryption to Multiple Encryption.

Figure 2 shows how to convert ciphertext $\psi_1 \leftarrow \mathsf{Enc}_{pk_1}(b)$ and $\psi_2 \leftarrow \mathsf{Enc}_{pk_2}(b)$ into ciphertext trees $T(\psi_1), T(\psi_2) \in \mathbf{T}_{pk_1,pk_2}$. Suppose $\psi_1 = b_1 b_2 \ldots b_n$. The tree on the left of Fig. 2 is the ciphertext tree corresponding to ψ_1, where a plaintext b is encrypted by pk_1 into $b_1 b_2 \ldots b_n$, each bit of which, b_i, is then encrypted by trivial encryption (using pk_2) into b_i itself. (The internal nodes of ciphertext trees have no labels; we include them in the figure only to illustrate the process.) As another example (the right tree in Fig. 2), suppose $\psi_2 = b'_1 \cdots b'_n \leftarrow \mathsf{Enc}_{pk_2}(b)$. Encrypting b into b (by trivial encryption using key pk_1) and then encrypting b into $b'_1 \ldots b'_n$ using pk_2, we obtain a 2-key ciphertext tree for ψ_2.

In general, to translate an ordinary ciphertext under key pk_i, say $\psi_i = b_1 b_2 \ldots b_n \leftarrow \mathsf{Enc}_{pk_i}(b)$, to a t-key ciphertext tree $\mathbf{T}_{pk_1,\ldots,pk_i}$, we start with a single node representing the plaintext b; encrypt b to b by trivial encryption for $i-1$ times (using keys $pk_1, \ldots, pk_{i-1}$, respectively); then encrypt b to ψ_i using key pk_i; then, encrypt each bit b_j of ψ_i by trivial encryption for $t-i$ times (i.e., using keys $pk_{i+1}, \ldots, pk_t$, respectively). The resulting ciphertext tree $T(\psi_i)$ has depth t; every internal node has a single child — since the bit corresponding to that node is encrypted by trivial encryption — except for the node at depth $i-1$, which has exactly $|\psi_i|$ children; the $|\psi_i|$ leaves of the tree are each labeled by a bit in ψ_i, in the natural left-to-right order. $T(\psi_i)$ is evidently unique in $\mathbf{T}_{pk_1,\ldots,pk_t}$.

Definition 12. For $\psi_i \leftarrow \mathsf{Enc}(pk_i, b)$, denote by $T^{(j)}(\psi_i)$ the j^{th} intermediate ciphertext tree during the construction of $T(\psi_i)$, with $T^{(0)}(\psi_i)$ being the starting tree (with a single node) and $T^{(t)}(\psi_i)$ the final tree. Note that the labels of the leaves of these trees are

$$Leaves(T^{(j)}(\psi_i)) = \begin{cases} b & \text{for } j < i \\ \psi_i & \text{for } j \geq i \end{cases} \tag{5}$$

6 Circuit Evaluation with Multiple Encryptions

Let $\mathcal{E}_i = \langle \mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i, \mathsf{Eval}_i \rangle$, $i = 1, \ldots, t$ be fully homomorphic encryption schemes with the trivial encryption property. Let $\mathcal{C}$ be a circuit and for $1 \le i \le t$, let $\psi_i \leftarrow \mathsf{Enc}_i(pk_i, x_i)$ be an encryption of x_i under key pk_i. We develop an algorithm $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_1 \rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_t \rangle)$ that returns a ciphertext tree $T \in \mathbf{T}_{pk_1,\ldots,pk_t}$ that can be correctly decrypted to $\mathcal{C}(x_1, \ldots, x_t)$ using keys $sk_1, \ldots, sk_t$.

6.1 Basic Idea

$$
\begin{array}{ccccccccc}
X & \xrightarrow{\mathsf{Enc}_{pk_1}} & X^{(1)} & \xrightarrow{\mathsf{Enc}_{pk_2}} & X^{(2)} & \xrightarrow{\mathsf{Enc}_{pk_3}} & \cdots & \xrightarrow{\mathsf{Enc}_{pk_t}} & X^{(t)} \\
\mathcal{C} & \xrightarrow{\mathsf{Eval}_{pk_1}} & \mathcal{C}^{(1)} & \xrightarrow{\mathsf{Eval}_{pk_2}} & \mathcal{C}^{(2)} & \xrightarrow{\mathsf{Eval}_{pk_3}} & \cdots & \xrightarrow{\mathsf{Eval}_{pk_t}} & \mathcal{C}^{(t)} \\
\mathcal{C}(X) & \xleftarrow{\mathsf{Dec}_{pk_1}} & \mathcal{C}^{(1)}(X^{(1)}) & \xleftarrow{\mathsf{Dec}_{pk_2}} & \mathcal{C}^{(2)}(X^{(2)}) & \xleftarrow{\mathsf{Dec}_{pk_3}} & \cdots & \xleftarrow{\mathsf{Dec}_{pk_t}} & \mathcal{C}^{(t)}(X^{(t)})
\end{array}
$$

Figure 3: Ideas behind the multikey evaluation algorithm

Figure 3 shows the basic idea behind our nested evaluation algorithm. Let $\mathcal{C}$ be a circuit and $X = (x_1, \ldots, x_t)$ its input. We wish to compute $\mathcal{C}(X)$ without knowing X but only its ciphertext tree under various schemes and keys $\langle \mathcal{E}_i, pk_i \rangle$, $1 \le i \le t$. Since $\mathcal{E}_i$ are FHE schemes, corresponding to $\mathcal{C}$ there is (at least conceptually) a circuit $\mathcal{C}^{(1)}$ such that if X is bitwise encrypted by pk_1 to $X^{(1)}$, then $\mathcal{C}^{(1)}(X^{(1)})$ will decrypt to $\mathcal{C}(X)$ by sk_1. (We will precisely define $X^{(i)}$ and $\mathcal{C}^{(i)}$ soon.) Now, for $\mathcal{C}^{(1)}$, there is a circuit $\mathcal{C}^{(2)}$ (derived from Eval_2) such that if $X^{(1)}$ is bitwise encrypted by pk_2 to $X^{(2)}$, then $\mathcal{C}^{(2)}(X^{(2)})$ will decrypt to $\mathcal{C}^{(1)}(X^{(1)})$ by sk_2. This reasoning is repeated until we reach $X^{(t)}$ and $\mathcal{C}^{(t)}$. During this process, if we use the ciphertext trees that trivially created from single ciphertexts $\psi_i \leftarrow \mathsf{Enc}_{pk_i}(x_i)$ as described in Section 5.3, we are able to produce a ciphertext tree for $\mathcal{C}(x_1, \ldots, x_t)$.

Definition 13. Let $\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval} \rangle$ be a FHE scheme and $\widetilde{\mathsf{Eval}}$ derived from Eval be the modified evaluation algorithm which evaluates a circuit on ciphertext trees. For a (fixed) circuit $\mathcal{C}$ with t input lines and a (fixed) key pk, let $\mathsf{Eval}[\mathcal{C}, pk]$ denote the following single key algorithm:

Algorithm $\mathsf{Eval}[\mathcal{C}, pk]$

input: ciphertext trees $T_1, \ldots, T_t$, where $T_i \leftarrow \widetilde{\mathsf{Enc}}(pk, x_i)$ for $1 \le i \le t$.

output: a ciphertext tree $T := \widetilde{\mathsf{Eval}}(\mathcal{C}, pk, T_1, \ldots, T_t)$.

That is, $\mathsf{Eval}[\mathcal{C}, pk]$ takes ciphertext trees $T_1, \ldots, T_t$ as input, calls $\widetilde{\mathsf{Eval}}(\mathcal{C}, pk, T_1, \ldots, T_t)$[3] to produce a ciphertext tree T, and returns T. We emphasize that $\mathcal{C}$, pk are "constants" that have been "hardwired" into the algorithm $\mathsf{Eval}[\mathcal{C}, pk]$.

Assumptions: We assume that every regular Eval_i for $1 \le i \le t$ is deterministic (as opposed to probabilistic); also assume that Eval_i outputs fixed length ciphertext given $\mathcal{C}$ and pk_i (see Definition 11), such that $\mathsf{Eval}_i[\mathcal{C}, pk_i] \in \Gamma_i$ if $\mathcal{C} \in \Gamma_{i-1}$ (by Lemma 3). For any circuit $\mathcal{C}$ and encryption key pk_i $\mathsf{Eval}_i[\mathcal{C}, pk_i]$ can be expressed as a circuit, which for simplicity of notation is also denoted by $\mathsf{Eval}_i[\mathcal{C}, pk_i]$. (Note that if the time complexity of the algorithm is polynomially bounded, then the corresponding circuit has a polynomial size.)

[3]The modified Eval that accepts ciphertex trees as input. As discussed in Section 5.2, Eval can be modified such that it accept depth 1 ciphertext as input and the result is a depth 1 ciphertext tree.

6.2 Nested Evaluation Circuits

For a given circuit $\mathcal{C}$ (with t input lines) and keys $pk_1, \ldots, pk_t$, let $\mathcal{C}^{(0)} = \mathcal{C}$, and for $1 \le i \le t$, recursively define $\mathcal{C}^{(i)}$ to be the circuit that evaluates $\mathcal{C}^{(i-1)}$ using key pk_i:

$$\begin{aligned}\mathcal{C}^{(i)} &\triangleq \mathsf{Eval}_i[\mathcal{C}^{(i-1)}, pk_i] \\ &= \mathsf{Eval}_i\big[\mathsf{Eval}_{i-1}[\mathcal{C}^{(i-2)}, pk_{i-1}], pk_i\big] \\ &= \mathsf{Eval}_i\big[\mathsf{Eval}_{i-1}[\cdots[\mathsf{Eval}_1[\mathcal{C}, pk_1], \cdots], pk_{i-1}], pk_i\big].\end{aligned}$$

These circuits $\mathcal{C}^{(i)}$ are "nested" evaluation algorithms. The input of circuit $\mathcal{C}^{(1)} = \mathsf{Eval}_1[\mathcal{C}, pk_1]$ is the bitwise encryption of X under key pk_1 — namely, $\mathsf{Enc}_1(pk_1, x_1), \ldots, \mathsf{Enc}_1(pk_1, x_t)$, which by Definition 9 are each a ciphertext tree in $\mathbf{T}_{pk_1}$. The output of $\mathcal{C}^{(1)}$ is an encryption of $\mathcal{C}(X)$, under key pk_1. $\mathcal{C}^{(2)} = \mathsf{Eval}_2[\mathcal{C}^{(1)}, pk_2]$, again by Definition 13, takes as input the encryptions of $\mathcal{C}^{(1)}$'s input, that is, $\mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_2}\mathsf{Enc}_{pk_1}(x_t) \in \mathbf{T}_{pk_1,pk_2}$. The output, by definition, is an encryption of $\mathcal{C}^{(1)}$'s output, under key pk_2; so it is in $\mathbf{T}_{pk_1,pk_2}$. In general, for $i \le t$, circuit $\mathcal{C}^{(i)}$ takes as input ciphertext trees

$$\mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_j) \in \mathbf{T}_{pk_1,\ldots,pk_i} \text{ for } 1 \le j \le t$$

and outputs a ciphertext tree in $\mathbf{T}_{pk_1,\ldots,pk_i}$. Let $X^{(i)} = (\mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_t))$ denote the array of input ciphertext trees of $\mathcal{C}^{(i)}$. Define the nested evaluation algorithms $\mathcal{C}^{(i)}$ for $i = 1, \ldots, t$ as follow:

Algorithm/Circuit $\mathcal{C}^{(i)}$

input: ciphertext trees $X^{(i)} = (\mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_t))$.

output: ciphertext tree $\psi^{(i)} := \widetilde{\mathsf{Eval}}_i\big(\mathcal{C}^{(i-1)}, pk_i, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_t)\big)$.

The following lemma indicates that the circuits constructed above are well defined.

Lemma 4. *For $1 \le i \le t$, $\psi^{(i)}$ is a ciphertext tree in $\mathbf{T}_{pk_1,\ldots,pk_i}$ and $\mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_i}(\psi^{(i)}) = \mathcal{C}(X)$.*

6.3 Final Evaluation Algorithm

Recall that our goal was $\mathsf{Evaluate}(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_t\rangle)$. Using the circuits developed above, $\mathcal{C}^{(t)}(X^{(t)})$ will give the value (ciphertext tree) of $\mathsf{Evaluate}$ (see Fig. 3), provided that we can obtain $X^{(t)}$ from $(\psi_1, \ldots, \psi_t)$. To this end, we use the method developed in Section 5.3 to construct ciphertext trees $T^{(i)}(\psi_j) \in \mathbf{T}_{pk_1,\ldots,pk_i}$ from single-key ciphertexts ψ_j.

From Definition 12, we see that $\widetilde{X}^{(i)} = (T^{(i)}(\psi_1), \ldots, T^{(i)}(\psi_i), T^{(i)}(x_{i+1}), \ldots, T^{(i)}(x_t))$ is a valid value of $X^{(i)}$. Using $\widetilde{X}^{(i)}$ as input to $\mathcal{C}^{(i)}$, we have the following construction:

The multi-key evaluation algorithm $\mathsf{Evaluate}$ for t keys is based on circuits $\mathcal{C}^{(i)}$ and encryption trees $\tilde{X}^{(i)}$, it is precisely defined as follows:

Algorithm $\mathsf{Evaluate}\big(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_t\rangle\big)$

input: A (description of) circuit $\mathcal{C}$, encryption schemes and their public keys $\langle \mathcal{E}_1, pk_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t\rangle$, ciphertexts $\psi_1, \ldots, \psi_t$, each encrypted under one of $\mathsf{Enc}_{pk_1}, \ldots, \mathsf{Enc}_{pk_t}$.

step 1: Construct $T^{(t)}(\psi_j)$ from ψ_j for $j = 1, \ldots, t$.

step 2: Construct the sequence of circuits $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(t)}$, where $\mathcal{C}^{(i)}$ is the circuit to implement algorithm $\mathsf{Eval}_i[\mathcal{C}^{(i-1)}, pk_i]$.

step 3: Output $\psi := \mathcal{C}^{(t)}\big(T^{(t)}(\psi_1), \ldots, T^{(t)}(\psi_t)\big)$.

We claim that $\mathcal{C}^{(t)}, t \in \mathbb{N}$ is the desired multikey evaluation algorithm for t keys, its output is a ciphertext tree $\psi^{(t)} \in \mathbf{T}_{pk_1,\ldots,pk_t}$ that can be correctly decrypted to $\mathcal{C}(x_1, \ldots, x_t)$ with secret keys $sk_1, \ldots, sk_t$.

Lemma 5 (Correctness)**.** *If* $\psi := \mathsf{Evaluate}\big(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_t\rangle\big)$*, then* $\mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_t}(\psi) = \mathcal{C}(x_1, \ldots, x_t)$.

Remark 3. According to Definition 2.1.2 in [Gen09a], an FHE is a non-trivial scheme if the size of decryption circuit is bounded by a polynomial of the security parameter, but is independent of the size of circuit $\mathcal{C}$. Similarly, a multi-scheme FHE is non-trivial if the size of its $\mathsf{Decrypt}$ circuit is independent of $|\mathcal{C}|$, see the compactness requirement in Definition 3. Because $\mathsf{Evaluate}$ outputs a ciphertext tree, $\mathsf{Decrypt}$ is defined as $\mathsf{Dec}_{sk_1}\mathsf{Dec}_{sk_2} \ldots \mathsf{Dec}_{sk_t}$, it is independent of the size of circuit, and is compact if underlying FHE schemes are all compact.

Performance Analysis. We assume that the regular Eval_i outputs fixed length ciphertexts (see Definition 11). Given ciphertexts ψ_j, $\mathsf{Evaluate}$ constructs trees $T^{(t)}(\psi_j)$, whose nodes are mostly trivial encryptions. The tree $T^1 = \widetilde{\mathsf{Eval}}_1(\mathcal{C}, pk_1, T^{(1)}(\psi_1), T^{(1)}(x_2), \ldots, T^{(1)}(x_t))$ has $\mathcal{L}_1^{\mathcal{C}}(|\psi_1|, 1, \ldots, 1)$ nodes on level 1. For each $i = 2 \ldots, t$, the tree $T^i = \widetilde{\mathsf{Eval}}_i(\mathcal{C}^{(i-1)}, pk_i, T^{(i)}(\psi_i), T^{(i)}(x_{i+1}), \ldots, T^{(i)}(x_t))$ has $\mathcal{L}_i^{\mathcal{C}^{(i-1)}}(|\psi_i|, 1, \ldots, 1)$ leaves on level i (see the proof of Lemma 5 for more details). The total number N of nodes in $\psi = T^t$ will be

$$N = \sum_{j=1}^{t} \Pi_{i=1}^{j} \mathcal{L}^{\mathcal{C}^{(i-1)}}(|\psi_i|, 1, \ldots, 1). \tag{6}$$

This total number is polynomial of lengths of ψ_i's, the length of input ciphertexts (and also the security parameters λ_i of each $\mathcal{E}_i$, which determine the lengths of the input ciphertexts). Number N simplifies to $N = \sum_{j=1}^{t} \Pi_{i=1}^{j} \mathcal{L}(|\psi_i|) = O\big(\Pi_{i=1}^{t} \mathcal{L}(|\psi_i|)\big)$ assuming $\mathcal{L}^{\mathcal{C}}(n, 1, \ldots, 1) = \mathcal{L}(n)$ for any circuit $\mathcal{C}$ of polynomial size, where n is the size of regular input ciphertext. However, it is exponential in t, the number of different schemes/keys in use.

Discussions. It is a open problem whether there exists an algorithm $\mathsf{Evaluate}$ that is polynomial in t. On possible direction is using the trivial encryption to reduce the internal nodes of the output tree. We are interested in some special sets of ciphertext trees of polynomial size (in t). If such trees are closed under multiplication and addition, i.e., multiplying or adding such trees will yield a tree in the same set, the whole evaluation process would be limited to polynomial time. Our point is, although the algorithm in the worst case $\mathcal{C}^{(t)}$ is exponential in t due to the output size is large, it is still possible that the algorithm becomes an efficient one (by limiting the calculation to some special set of ciphertext trees).

Remark 4. In algorithm $\mathsf{Evaluate}\big(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_1\rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_t\rangle\big)$, we assumed for simplicity of presentation that inputs are encrypted by different schemes. If some of the inputs are encrypted by the same scheme and/or key, our method will still work by generalizing algorithm $\mathsf{Evaluate}$ to

$$\mathsf{Evaluate}\big(\mathcal{C}, \langle \mathcal{E}_1, pk_1, \psi_{11}, \ldots, \psi_{1m_1}\rangle, \ldots, \langle \mathcal{E}_t, pk_t, \psi_{t1}, \ldots, \psi_{tm_t}\rangle\big).$$

7 Summary and Future Work

We proposed a generic method to evaluate circuits whose inputs are encrypted with (possibly) different FHE schemes. Our intention was to resolve the solvability issue of the seemingly impossible multi-scheme FHE problem. As a generic method, it did not make use of any algebraic or algorithmic properties of

individual FHE schemes except the trivial ciphertext property. The most important is to improve the efficiency of Evaluate algorithms. As our future work, we will consider specific FHE schemes, especially known multikey FHE schemes, and explore the possibility of merging them into an efficient multi-scheme FHE scheme. Another open problem, as mentioned in the introduction, is to design efficient and secure decryption algorithms or protocols for multi-scheme FHE.

References

[AJLA+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs, *Multiparty computation with low communication, computation and interaction via threshold fhe*, Advances in Cryptology–EUROCRYPT 2012 (2012), 483–501.

[BGI+01] Boaz Barak, Oded Goldreich, Rusell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang, *On the (im)possibility of obfuscating programs*, Advances in Cryptology CRYPTO 2001 (Joe Kilian, ed.), Lecture Notes in Computer Science, vol. 2139, Springer Berlin / Heidelberg, 2001, pp. 1–18.

[BV11a] Zvika Brakerski and Vinod Vaikuntanathan, *Efficient fully homomorphic encryption from (standard) lwe*, FOCS, 2011, pp. 97–106.

[BV11b] Zvika Brakerski and Vinod Vaikuntanathan, *Fully homomorphic encryption from ring-lwe and security for key dependent messages*, Advances in Cryptology CRYPTO 2011 (Phillip Rogaway, ed.), Lecture Notes in Computer Science, vol. 6841, Springer Berlin/Heidelberg, 2011, pp. 505–524.

[CKV10] Kai-Min Chung, Yael Kalai, and Salil Vadhan, *Improved delegation of computation using fully homomorphic encryption*, Advances in Cryptology–CRYPTO 2010 (2010), 483–501.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi, *Batch fully homomorphic encryption over the integers*, Tech. report, Cryptology ePrint Archive, Report 2013/036, 2013.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias, *Multiparty computation from somewhat homomorphic encryption*, CRYPTO (Reihaneh Safavi-Naini and Ran Canetti, eds.), Lecture Notes in Computer Science, vol. 7417, Springer, 2012, pp. 643–662.

[Gen09a] Craig Gentry, *A fully homomorphic encryption scheme*, Ph.D. thesis, Stanford University, 2009.

[Gen09b] Craig Gentry, *Fully Homomorphic Encryption Using Ideal Lattices*, Proceedings of the 41st annual ACM symposium on Theory of computing, STOC '09, 2009, pp. 169–178.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno, *Non-interactive verifiable computing: Outsourcing computation to untrusted workers*, CRYPTO, 2010, pp. 465–482.

[GH11] Craig Gentry and Shai Halevi, *Implementing gentrys fully-homomorphic encryption scheme*, Advances in Cryptology–EUROCRYPT 2011 (2011), 129–148.

[GHS12] Craig Gentry, Shai Halevi, and Nigel Smart, *Fully homomorphic encryption with polylog overhead*, Advances in Cryptology–EUROCRYPT 2012 (2012), 465–482.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson, *How to play any mental game*, Proceedings of the nineteenth annual ACM symposium on Theory of computing (New York, NY, USA), STOC '87, 1987, pp. 218–229.

[Kil88] Joe Kilian, *Founding crytpography on oblivious transfer*, Proceedings of the twentieth annual ACM symposium on Theory of computing, STOC '88, 1988, pp. 20–31.

[KK12] Vladimir Kolesnikov and Ranjit Kumaresan, *Improved secure two-party computation via information-theoretic garbled circuits*, Security and Cryptography for Networks (2012), 205–221.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan, *On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption*, STOC (Howard J. Karloff and Toniann Pitassi, eds.), ACM, 2012, pp. 1219–1234.

[SS10] Damien Stehlé and Ron Steinfeld, *Faster fully homomorphic encryption*, Advances in Cryptology-ASIACRYPT 2010 (2010), 377–394.

[SV10] N. Smart and F. Vercauteren, *Fully homomorphic encryption with relatively small key and ciphertext sizes*, Public Key Cryptography PKC 2010 (Phong Nguyen and David Pointcheval, eds.), Lecture Notes in Computer Science, vol. 6056, Springer Berlin / Heidelberg, 2010, pp. 420–443.

[vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan, *Fully homomorphic encryption over the integers*, EUROCRYPT, 2010, pp. 24–43.

[vDJ10] Marten van Dijk and Ari Juels, *On the impossibility of cryptography alone for privacy-preserving cloud computing*, Proceedings of the 5th USENIX conference on Hot topics in security (Berkeley, CA, USA), HotSec'10, USENIX Association, 2010, pp. 1–8.

[Yao82] Andrew C. Yao, *Protocols for secure computations*, Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (Washington, DC, USA), SFCS '82, IEEE Computer Society, 1982, pp. 160–164.

[Yao86] Andrew Chi-Chih Yao, *How to generate and exchange secrets*, Proceedings of the 27th Annual Symposium on Foundations of Computer Science (Washington, DC, USA), SFCS '86, IEEE Computer Society, 1986, pp. 162–167.

A Proofs

A.1 Proof of Lemma 1

Proof. Let $\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval} \rangle$ be an FHE scheme with plaintext space $\{0,1\}$ and ciphertext space $\mathcal{R} \subseteq \{0,1\}^+$ ($\{0,1\}^+$ denotes the set of all non-empty bit strings). Assume that $\mathcal{E}$ does not have the trivial encryption property, or we are done. We will convert $\mathcal{E}$ to an FHE scheme $\mathcal{E}' = \langle \mathit{Gen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}' \rangle$ with the trivial encryption property. The idea is to append or prepend all ciphertexts in $\mathcal{R}$ with a fixed string of $l \geq 0$ such that every ciphertext now has length at least 2. Then add 0 and 1 to R as the trivial ciphertext of 0 and 1, respectively.

Specifically, let $f(r) = 0 \| r$ be a function that prepends $r \in \mathcal{R}$ with a 0 bit. (Every string in $f(R)$ now has at least two bits.) Construct an FHE scheme $\mathcal{E}' = \langle \mathsf{KeyGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}' \rangle$ as follows.

- KeyGen': Same as the KeyGen of $\mathcal{E}$.

- Enc': On input a plaintext b and a public key pk, output b with probability p, and run $c \leftarrow \mathsf{Enc}(pk, b)$ and output $f(c)$ with probability $1-p$, where p is a negligible value, say $p = 2^{-\lambda}$ with λ being the security parameter. Thus,

$$\mathsf{Enc}'(pk, b) \triangleq \begin{cases} b & \text{with a negligible probability } p \\ f\big(\mathsf{Enc}(pk, b)\big) & \text{with probability } 1-p \end{cases}$$

  (Note: 0, 1 are now valid ciphertexts of 0, 1, respectively.)

- Dec': On input a ciphertext c' and decryption key sk, if $c' \in \{0,1\}$, return c'; if $c' \in f(\mathcal{R})$, return $\mathsf{Dec}_{sk}\big(f^{-1}(c')\big)$; otherwise, return *FALSE*.

  (Note: the new ciphertext space is $\mathcal{R}' = \{0,1\} \cup f(\mathcal{R})$. In the rest of the proof, the ciphertexts in $\mathcal{R}'$ and those in $\mathcal{R}$ will be referred to as $\mathcal{E}'$-ciphertexts and $\mathcal{E}$-ciphertexts, respectively.)

- Eval': Given as input a public key pk, a circuit $\mathcal{C}$, and $\mathcal{E}'$-ciphertexts $c'_1, c'_2, \ldots, c'_t \in \mathcal{R}'$, where t is the number of input lines of $\mathcal{C}$, we will convert $\mathcal{E}'$-ciphertexts c'_i into $\mathcal{E}$-ciphertexts c_i, call the Eval of $\mathcal{E}$ to evaluate the circuit, convert the resulting $\mathcal{E}$-ciphertext into an $\mathcal{E}'$-ciphertext, and output the latter. To that end, let x_i denote the plaintext of c'_i, $1 \le i \le t$. (So, Eval' is to evaluate $\mathcal{C}(x_1, \ldots, x_t)$ homomorphically.) Let $\psi_b \leftarrow \mathsf{Enc}_{pk}(b)$ be any $\mathcal{E}$-ciphertext of $b \in \{0,1\}$. For $1 \le i \le t$, compute a $\mathcal{E}$-ciphertext c_i of x_i as follows:

$$c_i := \begin{cases} \psi_{c'_i} & \text{if } c'_i \in \{0,1\}, \text{ i.e., a trivial encryption under } \mathcal{E}' \\ f^{-1}(c'_i) & \text{otherwise} \end{cases}$$

  Now, evaluate circuit $\mathcal{C}$ using the Eval algorithm of $\mathcal{E}$ and convert the resulting $\mathcal{E}$-ciphertext into an $\mathcal{E}'$-ciphertext, which is the output of $\mathsf{Eval}'(pk, \mathcal{C}, c'_1, \ldots, c'_t)$. That is,

$$c := \mathsf{Eval}(pk, \mathcal{C}, c_1, \ldots, c_t) \quad \text{and} \quad \mathsf{Eval}'(pk, \mathcal{C}, c'_1, \ldots, c'_t) := c' := f(c).$$

Trivial Encryption: The scheme $\mathcal{E}'$ thus constructed clearly has the trivial encryption property.

Correctness: We need to show $\mathsf{Dec}'_{sk}(c') = \mathcal{C}(x_1, \ldots, x_t)$, where $c' := f(c)$ and $c := \mathsf{Eval}(pk, \mathcal{C}, c_1, \ldots, c_t)$. Since c_i is an $\mathcal{E}$-ciphertext of x_i and $\mathcal{E}$ is fully homomorphic, we have $\mathcal{C}(x_1, \ldots, x_t) = \mathsf{Dec}_{sk}(c)$. On the other hand, by definition of Dec', $\mathsf{Dec}'_{sk}(c') = \mathsf{Dec}_{sk}(f^{-1}(c')) = \mathsf{Dec}_{sk}(c)$. Hence, $\mathsf{Dec}'_{sk}(c') = \mathcal{C}(x_1, \ldots, x_t)$.

Compactness: The circuit complexity of Dec' is the complexity of Dec plus that of f^{-1}. The latter is polynomial in λ since $\mathcal{E}$ is a compact FHE. (Note that the complexity of f^{-1} is *not* higher than Dec.) Hence, $\mathcal{E}$ is compact.

Security: Since Eval' can never produce 0 or 1 (trivial ciphertexts), a challenge $\mathcal{E}'$-ciphertext must be produced by Enc'. Since there is a one-to-one correspondence between $\mathcal{E}$-ciphertexts and nontrivial $\mathcal{E}'$-ciphertexts (defined by f) and since the probability of trivial $\mathcal{E}'$-ciphertexts being produced is negligible, the probability that an adversary can distinguish $\mathcal{E}'$-ciphertexts is at best *negligibly* larger than that of distinguishing $\mathcal{E}$-ciphertexts. Hence $\mathcal{E}'$ is as secure as $\mathcal{E}$ in terms of ciphertext indistinguishability. □

A.2 Proof of Lemma 3

Proof. To show $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk] \in \Gamma_{i+1}$ we need to show the three conditions in Definition 10 hold for $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$: (1) the input and output are ciphertext trees of depth $i+1$; (2) the graph of T' is determined alone by the graphs of $T'_1, \ldots, T'_s$; (3) the labels of leaves of T' are determined by those of $T'_1, \ldots, T'_s$.

Checking the input/output trees of $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$, we readily see that condition (1) holds for all $\mathcal{C} \in \Gamma_i$: suppose, w.l.o.g., $\mathcal{C} \in \Gamma_i$ takes input and output in $\mathbf{T}_{pk_1,\ldots,pk_i}$, $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ will operate on the encryptions (under pk_i) of trees in $\mathbf{T}_{pk_1,\ldots,pk_i}$, that is, $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$'s input and output are in $\mathbf{T}_{pk_1,\ldots,pk_i,pk}$, trees of depth $i+1$. We need to check condition (2) and (3) in the definition.

First consider $i = 0$. The set of Γ_0 is the set of all circuits whose inputs and output are single node tree; circuits that output a single bit fall in this category. Suppose $T = \mathcal{C}(T_1, \ldots, T_s)$, where $\mathcal{C} \in \Gamma_0$, $T, T_1, \ldots, T_s$ are single nodes. $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ accepts ciphertext trees $T'_1, \ldots, T'_s \in \mathbf{T}_{pk}$ as input and outputs a ciphertext tree $T' \in \mathbf{T}_{pk}$. It will do the following to produce T': $\mathcal{C}$ outputs a single node, so $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ knows that $G(T)$ is a single node (see step 1 and 2 of the $\widetilde{\mathsf{Eval}}$). The *label* of T is calculated as $T = \mathcal{C}(T_1, \ldots, T_s)$; in the step 3 of $\widetilde{\mathsf{Eval}}$, the algorithm will call Eval to evaluate $\mathcal{C}$, producing an encryption of the label:

$$\psi := \mathsf{Eval}(\mathcal{C}, pk, \psi_1, \ldots, \psi_s)$$

where $\psi_j := \mathsf{Enc}(pk, T_j)$ is a ciphertext (rather than a tree), whose bits are the labels of T'_j. $\widetilde{\mathsf{Eval}}$ will then attach all bits of ψ to the leaf node in $G(T)$, resulting in the tree T'. Now we check condition (2): $G(T')$ is a root with $|\psi|$ leaves; because Eval is an algorithm that outputs fixed length ciphertexts, so the length of ψ is determined by the lengths of ψ_j:

$$|\psi| = \mathcal{L}^{\mathcal{C},pk}(|\psi_1|, \ldots, |\psi_s|),$$

since $|\psi_j|$ is the number of leaves of T'_j, thus once the graphs of $T'_1, \ldots, T'_s$ are known, $|\psi|$ (hence the graph of T) is determined. Condition (2) is satisfied. To see condition (3), the labels of T' is computed from the labels of the input trees. Therefore, $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk] \in \Gamma_1$ for $\mathcal{C} \in \Gamma_0$.

We will show if the lemma holds for some $i \geq 1$, it will also hold for $i = i+1$. Suppose $T = \mathcal{C}(T_1, \ldots, T_s)$ where $T, T_1, \ldots, T_s \in \mathbf{T}_{pk_1,\ldots,pk_i}$. Given ciphertexts $T'_j \leftarrow \widetilde{\mathsf{Enc}}(pk, T_j)$ for $1 \leq j \leq s$, $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ will do the following to produce T'. It repeats what $\mathcal{C}$ does to produce the graph of T (see step 2 in $\widetilde{\mathsf{Eval}}$'s algorithm). Next it deals with the "labels" of T. $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ evaluates the label for a leaf z of T: suppose $z = \mathcal{C}_z(y_1, \ldots, y_u)$ where $y_1, \ldots, y_u$ are labels of all leaves in $T_1, \ldots, T_s$; $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ calls

$$\psi_z = \mathsf{Eval}(\mathcal{C}_z, pk, \psi_1, \ldots, \psi_u)$$

where $\psi_k = \mathsf{Enc}(pk, y_k)$ for $1 \leq k \leq u$. Then $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ attaches the bits of ψ_z to node z; repeatedly doing so for all leaves yields the tree T'.

We shall see that ciphertexts $\psi_1, \ldots, \psi_u$ can be found in $T'_1, \ldots, T'_s$: suppose y_k $(1 \leq k \leq u)$ is a leaf/label of input tree T_j $(1 \leq j \leq s)$; by Definition 7 of $\widetilde{\mathsf{Enc}}$, if ciphertext tree T'_j is an encryption of T_j, then T'_j contains T_j (or more precisely, $G(T_j)$ is isomorphic to a sub-graph of $G(T'_j)$, this sub-graph contains all internal nodes and their edges of $G(T'_j)$, but does not include any leaf node of $G(T'_j)$), each leaf/label y_k of T_j has children in T'_j, the labels of these children compose a ciphertext ψ_k of y_k.

Now we can check the condition (2). The non-leaf part of graph $G(T')$ is isomorphic to $G(T)$, because $G(T)$ is determined by the graphs of input trees, so the non-leaf sub-graph of $G(T')$ is determined by the non-leaf subgraphs of $T'_1, \ldots, T'_s$ (see step 2 of the algorithm); each leaf z in T has an "image" z' in T' (z' is a unlabeled intermal node in T'), z' has $|\psi_z|$ leaves, where

$$|\psi_z| = \mathcal{L}^{\mathcal{C}_z,pk}(|\psi_1|, \ldots, |\psi_u|).$$

Given $\mathcal{C}$, $|\psi_z|$ is only determined by $|\psi_1|, \ldots, |\psi_u|$, and each $|\psi_k|$ is the number of children of leaf y_k from the input tree T'_j, this information can be obtained from the graph of T'_j; it needs not to know the values of ψ_k before determining $|\psi_z|$, the number of children of z in T. That means $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk]$ is able to produce

the graph of T' when given the graphs of $T'_1, \ldots, T'_s$, it satisfies property (2) in the definition. To see the condition (3): the labels of T' are calculated from the labels of the $T'_1, \ldots, T'_s$ by evaluating each $\mathcal{C}_z$. So we have $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk] \in \Gamma_{i+1}$, the lemma holds.

It is worth mentioning that Γ_i for $i \geq 1$ are not empty. We have shown that Γ_0 is the set of all circuits and $\widetilde{\mathsf{Eval}}[\mathcal{C}, pk] \in \Gamma_1$, Γ_1 is not empty. In particular, suppose the circuit $\mathsf{Eval}^i \triangleq \widetilde{\mathsf{Eval}}_i[\mathcal{C}, pk_i] \in \Gamma_i$, so $\widetilde{\mathsf{Eval}}_{i+1}[\mathsf{Eval}^i, pk_{i+1}] \in \Gamma_{i+1}$. The idea of "evaluating $\widetilde{\mathsf{Eval}}$" is used in constructing the nested evaluation in Section 6.2. □

A.3 Proof of Lemma 4

Proof. By induction. For $i = 1$, algorithm $\mathcal{C}^{(1)}$ takes a tuple of ciphertext trees $X^{(1)} = \langle \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_1}(x_s) \rangle$ as input, it calls the modified Eval (see Lemma 3) and gets a ciphertext tree $\psi^{(1)} := \widetilde{\mathsf{Eval}}_1\big(\mathcal{C}, pk_1, \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}pk_1(x_t)\big)$, where $\widetilde{\mathsf{Dec}}_{sk_1}(\psi^{(1)}) = \mathcal{C}(X)$ (by Lemma 2).

Now consider $\mathcal{C}^{(2)}$. Since $\mathcal{C}^{(1)} = \mathsf{Eval}_1[\mathcal{C}, pk_1] \in \Gamma_1$, $\mathcal{C}^{(2)}$ can use $\widetilde{\mathsf{Eval}}$ to evaluate $\mathcal{C}^{(1)}$ and output a ciphertext tree $\psi^{(2)}$. By Lemma 2, $\psi^{(2)} \in \mathbf{T}_{pk_1,pk_2}$ and $\mathsf{Dec}(sk_2, \psi^{(2)}) = \psi^{(1)}$ and thus $\widetilde{\mathsf{Dec}}_{sk_1}\widetilde{\mathsf{Dec}}_{sk_2}(\psi^{(2)}) = \mathcal{C}(X)$. Also, by Definition 10 and Lemma 3, $\mathcal{C}^{(2)} = \mathsf{Eval}_2[\mathcal{C}^{(1)}, pk_2] \in \Gamma_2$.

Assume $\psi^{(i-1)} = \mathcal{C}^{(i-1)}(X^{(i-1)})$ is a ciphertext in $\mathbf{T}_{pk_1,\ldots,pk_{i-1}}$ and $\mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_{i-1}}(\psi^{(i-1)}) = \mathcal{C}(X)$ for $i \leq t$. Again, by lemma 3, $\mathcal{C}^{(i-1)} \in \Gamma_{i-1}$. The circuit $\mathcal{C}^{(i)}$ has ciphertext trees $X^{(i)}$, it calls

$$\psi^{(i)} = \widetilde{\mathsf{Eval}}_i i\big(\mathcal{C}^{(i-1)}, pk_i, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_1), \ldots, \mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_t)\big),$$

where $\mathsf{Enc}_{pk_i} \ldots \mathsf{Enc}_{pk_1}(x_j) = X_j^{(i)}$ comes from $X^{(i)}$. By Lemma 2, the output $\psi^{(i)}$ is an encryption of $\mathcal{C}^{(i-1)}(X^{(i-1)})$, that is, $\widetilde{\mathsf{Dec}}(pk_i, \psi) = \mathcal{C}^{(i-1)}(X^{(i-1)}) = \psi^{(i-1)}$. By the induction hypothesis,

$$\mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_i}(\psi^{(i)}) = \mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_{i-1}}\big(\widetilde{\mathsf{Dec}}_{sk_i}(\psi^{(i)})\big) = \mathsf{Dec}_{sk_1} \ldots \mathsf{Dec}_{sk_{i-1}}(\psi^{(i-1)}) = \mathcal{C}(X).$$

By Definition 10 and Lemma 3, $\mathcal{C}^{(i)} = \mathsf{Eval}[\mathcal{C}^{(i-1)}, pk_i] \in \Gamma_i$. It shows that the lemma is true for i if it is true for $i-1$, by induction, the lemma is true for $i = 1, \ldots, t$. □

A.4 Proof of Lemma 5

Proof. In the first step of $\mathsf{Evaluate}$, the algorithm constructs $T^{(t)}(\psi_j) \in \mathbf{T}_{pk_1,\ldots,pk_t}$. As a result, the inputs of $\mathcal{C}^{(i)}$ are $T^{(i)}(\psi_1), \ldots, T^{(i)}(\psi_i)$ and $T^{(i)}(x_{i+1}), \ldots, T^{(i)}(x_t)$, the trivial trees of variables $x_{i+1}, \ldots, x_t$. We will study how to evaluating a circuit with such trees.

$\mathcal{C}^{(1)}$ calls $\widetilde{\mathsf{Eval}}$ with $T^{(1)}(\psi_1), T^{(1)}(x_2), \ldots, T^{(1)}(x_t)$ then outputs a tree T^1. Notice that $T^{(1)}(x_j)$ is a tree with single leaf labeled x_j (for $j = 2, \ldots, t$); $\widetilde{\mathsf{Eval}}$ computes the leaves labels by $\mathsf{Eval}\big(\mathcal{C}, pk_1, \psi_1, x_2, \ldots, x_t\big)$, which, given the algorithm Eval, circuit $\mathcal{C}$ and value of ψ_1, is a function (circuit) of $x_2, \ldots, x_t$:

$$\mathsf{Eval}\big(\mathcal{C}, pk_1, \psi_1, x_2, \ldots, x_t\big) = f^{\mathsf{Eval},\mathcal{C},\psi_1}(x_2, \ldots, x_t),$$

denote the function $f^{\mathsf{Eval},\mathcal{C},\psi_1} \triangleq f^1$. By assumption, Eval outputs ciphertext of length $\mathcal{L}^{\mathcal{C}}(|\psi_1|, 1, \ldots, 1)$, it is also the number of leaves in T^1; denote f^1_k the sub-circuit of f^1 which generates the k^{th} bit. Label the k^{th} leaf in T^1 by f^1_k. Now T^1 is a tree whose leaves are labeled by functions, it decrypts to $\mathcal{C}(x_1, \ldots, x_t)$ given any value of $x_2, \ldots, x_t$.

$\mathcal{C}^{(2)}$ will in turn evaluate $f^1 = f^{\mathsf{Eval},\mathcal{C},\psi_1}$ given ciphertexts trees $T^{(2)}(\psi_1), T^{(2)}(\psi_2), T^{(2)}(x_3), \ldots, T^{(2)}(x_t)$. then outputs a tree T^2. Since the value of ψ_1 is known (the leaves of $T^{(2)}(\psi_1)$ are the bit-wise trivial encryptions of ψ_1), $\mathcal{C}^{(2)}$ is able to construct the function $f^{\mathsf{Eval},\mathcal{C},\psi_1}$. As shown in the proof of Lemma 3, the graph of T^1 is known if given encryptions of input trees of $\mathcal{C}^{(1)}$; the leaves of T^1 are labeled by

f^1. $\mathcal{C}^{(2)}$ evaluates these labels using $\mathsf{Eval}(f_k^1, pk_2, \psi_2, x_3, \ldots, x_t)$ ($\psi_2, x_3, \ldots, x_t$ are encryptions of f^1's input) then attaches the result to the leaves of T^1. Again, $\mathsf{Eval}(f^1, pk_2, \psi_2, x_3, \ldots, x_t)$ is a function $f^{\mathsf{Eval},f^1,\psi_2}(x_3, \ldots, x_t) = f^{\mathsf{Eval},\mathcal{C},\psi_1,\psi_2}(x_3, \ldots, x_t)$, we denote it by $f^2(x_3, \ldots, x_t)$. By assumption, the k^{th} leaf in T^2 has $\mathcal{L}^{f_k^1}$ children, each children is labeled by a sub-function of f^2. According to Lemma 2, T^2 is an encryption of T^1 in the sense that substitute variables $x_3, \ldots, x_t$ with any values, it holds that $\widetilde{\mathsf{Dec}}_{sk_2}(T^2) = T^1$, where we have $\mathsf{Dec}_{sk_1}\mathsf{Dec}_{sk_2}(T^2) = \mathcal{C}(x_1, \ldots, x_t)$.

This argument of $\mathcal{C}^{(i)}$ can be repeated for $i = 3, \ldots, t$. In the final stage, $\mathcal{C}^{(t)}$ evaluates the labels $f^{t-1}(x_t) = f^{\mathsf{Eval},\mathcal{C},\psi_1,\ldots,\psi_{t-1}}(x_t)$ with x_t's ciphertext ψ_t. The output tree T^t is a ciphertext tree, decrypting it with sk_t results in T^{t-1}, which, by induction, decrypts to $\mathcal{C}(x_1, \ldots, x_t)$ by secret keys $sk_{t-1}, \ldots, sk_1$. $\mathsf{Evaluate}$ outputs $\psi := T^t$, we have $\mathsf{Dec}_{pk_1} \ldots \mathsf{Dec}_{pk_t}(\psi) = \mathcal{C}(x_1, \ldots, x_t)$, that completes the proof. □

B Definition of Multi-Key FHE (From [LATV12])

We are interested in an encryption scheme with an evaluation algorithm Eval that allows the input ciphertexts ψ_i to be encryptions under *different* keys pk_i:

$$\psi \leftarrow \mathsf{Eval}(\mathcal{C}, pk_1, \ldots, pk_t, \psi_1, \ldots, \psi_t) \tag{7}$$

where $\psi_i \leftarrow \mathsf{Enc}_{pk_i}(x_i)$. Such an encryption scheme is a multikey fully homomorphic encryption scheme.

The concept of multikey FHE was first proposed in [LATV12], where a multikey FHE scheme is defined as *a family of encryption schemes* $\{\mathcal{E}^{(N)} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}\rangle\}_{N>0}$, where each $\mathcal{E}^{(N)}$ is an N-key FHE scheme.

Our definition of multikey FHE captures the essence of multikey FHE as defined in [LATV12], but is simpler.

Definition 14. (MultiKey FHE or MK-FHE) A *multikey fully homomorphic encryption scheme* consists of four algorithms: $\mathcal{E} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval}\rangle$.

- KeyGen is a randomized algorithm that, on input a security parameter λ, outputs a pair of public and secret keys: $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- Enc is a randomized algorithm that, on input a public key pk and a plaintext x, outputs a ciphertext: $\psi \leftarrow \mathsf{Enc}(pk, x)$.
- Dec is a deterministic algorithm that, on input a ciphertext ψ and secret keys $sk_1, \ldots, sk_t$, outputs a plaintext: $x \leftarrow \mathsf{Dec}(sk_1, \ldots, sk_t, \psi)$.
- Eval is an algorithm that on input a circuit $\mathcal{C}$, public keys $pk_1, \ldots, pk_t$ and ciphertexts $\psi_1, \ldots, \psi_t$ where $\psi_i \leftarrow \mathsf{Enc}(pk_i, x_i)$ for $i = 1, \ldots, t$, outputs a ciphertext:

$$\psi := \mathsf{Eval}(\mathcal{C}, pk_1, \ldots, pk_t, \psi_1, \ldots, \psi_t).$$

It is required that the following conditions hold:

- **Correctness:** For any $t > 0$, any t-input circuit $\mathcal{C}$, any t-tuple of key pairs $\{\langle pk_i, sk_i\rangle\}_{i=1}^t$ generated by $\mathsf{KeyGen}(1^\lambda)$, and any t-tuple of ciphertexts $\{\psi_i\}_{i=1}^t$, it holds that

$$\text{If } \psi_i \leftarrow \mathsf{Enc}(pk_i, x_i) \text{ for } i = 1, \ldots, t \text{ and } \psi \leftarrow \mathsf{Eval}(\mathcal{C}, pk_1, \ldots, pk_t, \psi_1, \ldots, \psi_t)$$
$$\text{then } \mathcal{C}(x_1, \ldots, x_t) = \mathsf{Dec}(sk_1, \ldots, sk_t, \psi).$$

- **Compactness:** The MK-FHE scheme is *compact* if there are polynomials d and e such that, for every value of the security parameter λ and every value of t (number of distinct keys), Dec and Eval can be expressed as circuits of sizes at most $d(\lambda, t)$ and $e(\lambda, t)$, respectively.

The notion of N-key FHE [LATV12] can be defined as a multikey FHE with no more than N different keys used in evaluating a circuit:

Definition 15. (N-key FHE) An N-key FHE scheme $\mathcal{E}^{(N)} = \langle \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval} \rangle$ is a multikey FHE with the restriction that no more than N different keys may be used in evaluating a circuit (i.e., the set of key pairs $\{\langle pk_i, sk_i \rangle\}_{i=1}^{t}$ in the correctness condition consists of no more than N different pairs).

Note that the compactness condition of N-key FHE in [LATV12] requires $|\psi| \leq P(\lambda, N)$ for some polynomial P.