Certified Everlasting Functional Encryption

Taiga Hiroka[1], Tomoyuki Morimae[1], Ryo Nishimaki[2], Takashi Yamakawa[1,2]

[1]Yukawa Institute for Theoretical Physics, Kyoto University, Kyoto, Japan
{taiga.hiroka,tomoyuki.morimae}@yukawa.kyoto-u.ac.jp
[2]NTT Corporation, Tokyo, Japan
{ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp

July 28, 2022

Abstract

Computational security in cryptography has a risk that computational assumptions underlying the security are broken in the future. One solution is to construct information-theoretically-secure protocols, but many cryptographic primitives are known to be impossible (or unlikely) to have information-theoretical security even in the quantum world. A nice compromise (intrinsic to quantum) is certified everlasting security, which roughly means the following. A receiver with possession of quantum encrypted data can issue a certificate that shows that the receiver has deleted the encrypted data. If the certificate is valid, the security is guaranteed even if the receiver becomes computationally unbounded. Although several cryptographic primitives, such as commitments and zero-knowledge, have been made certified everlasting secure, there are many other important primitives that are not known to be certified everlasting secure.

In this paper, we introduce certified everlasting FE. In this primitive, the receiver with the ciphertext of a message m and the functional decryption key of a function f can obtain $f(m)$ and nothing else. The security holds even if the adversary becomes computationally unbounded after issuing a valid certificate. We, first, construct certified everlasting FE for $\mathbf{P}/\mathbf{poly}$ circuits where only a single key query is allowed for the adversary. We, then, extend it to q-bounded one for $\mathbf{NC}^1$ circuits where q-bounded means that q key queries are allowed for the adversary with an a priori bounded polynomial q. For the construction of certified everlasting FE, we introduce and construct certified everlasting versions of secret-key encryption, public-key encryption, receiver non-committing encryption, and a garbling scheme, which are of independent interest.

Contents

1 Introduction

1.1 Background

Computational security in cryptography relies on assumptions that some problems are hard to solve. It, however, has a risk that the assumptions could be broken in a future when revolutionary novel algorithms are discovered or computing devices are drastically improved. One solution to the problem is to construct information-theoretically-secure protocols [Sha79, BB84], but even in the quantum world, many cryptographic primitives are known to be impossible (or unlikely) to have information-theoretical security [LC97, May97, MW18].

Good compromises (intrinsic to quantum!) have been studied recently [Unr15, BI20, KT20, HMNY21b, HMNY21a, Por22]. In particular, certified everlasting security, which was introduced in [HMNY21a] based on [Unr15, BI20], achieves the following security: a receiver with possession of quantum encrypted date issues a certificate which shows that the receiver has deleted its quantum encrypted data. If the certificate is valid, the security is guaranteed even if the receiver becomes computationally unbounded later (and even if some secret information like the secret key is leaked). This security notion is weaker than the information-theoretical security. (For example, a malicious receiver may refuse to issue a valid certificate.) It is, however, still a useful security notion, because, for example, a sender can penalize receivers who do not issue valid certificates. Moreover, certified everlasting security is intrinsically quantum property, because it implies information-theoretical security in the classical world. [1]

Certified everlasting security can bypass the impossibility of information-theoretical security. In fact, several cryptographic primitives have been shown to have certified everlasting security, such as commitments and zero-knowledge [HMNY21a]. An important open problem in this direction is

Which cryptographic primitives can have certified everlasting security?

Functional encryption (FE) is one of the most advanced cryptographic primitives that achieves large flexibility in controlling encrypted data [BSW11]. In FE, an owner of a master secret key MSK can generate a functional decryption key sk_f that hardwires a function f. When a ciphertext $\mathsf{CT}(m)$ of a message m is decrypted by sk_f, we obtain $f(m)$. No information beyond $f(m)$ is obtained. Information-theoretically-secure FE seems to be unlikely, and in fact all known constructions are computationally secure ones [GVW12, GGH$^+$13, GGHZ16, AS17, LT17, AJL$^+$19, AJS18, Agr18, LM18]. Hence we have the following open problem:

Is it possible to construct certified everlasting secure FE?

We remark that certified everlasting FE is particularly useful compared with certified everlasting public key encryption (PKE) (or more generally "all-or-nothing encryption" [GMM17] such as identity-based encryption (IBE), attribute-based encryption (ABE), or witness encryption (WE)) because it ensures security even against an honest receiver who holds a decryption key. That is, we can ensure that a receiver who holds a decryption key sk_f with respect to a function f cannot learn more than $f(m)$ even if the receiver can run unbounded-time computation after issuing a valid certificate. In contrast, certified everlasting PKE does not ensure any security against an honest receiver since the receiver can simply copy an encrypted message after honestly decrypting a ciphertext and then no security remains.

1.2 Our Results

We partially solve the above questions affirmatively. Our contributions are as follows:

1. We formally define certified everlasting versions of secret-key encryption (SKE) (Section 3.1), public-key encryption (PKE) (Section 4.1), receiver non-committing encryption (RNCE) (Section 5.1), a garbling scheme (Section 6.1), and FE (Section 7.1), respectively.

2. We present two constructions of certified everlasting SKE (resp. PKE). An advantage of the first construction is that the certificate is classical, but a disadvantage is that the security proof relies on the quantum random oracle

[1] This is because a malicious receiver can copy the encrypted data freely, and thus the encrypted data must be secure against an unbounded malicious receiver at the point when the receiver obtains the encrypted data. On the other hand, in the quantum world, the same discussion does not go through, because even a malicious receiver cannot copy the quantum encrypted data due to the quantum no-cloning theorem.

model (QROM) [BDF$^+$11]. On the other hand, in the second construction, the security holds without relying on the QROM, but the certificate is quantum.

3. We construct certified everlasting RNCE from certified everlasting PKE in a black-box way (Section 5.2).
4. We construct a certified everlasting garbling scheme for all **P**/**poly** circuits from certified everlasting SKE in a black-box way (Section 6.2).
5. We construct 1-bounded certified everlasting FE with adaptive security for all **P**/**poly** circuits. The adaptive security means that the adversary can call key queries before and after seeing the challenge ciphertext. The 1-bounded means that only a single key query is allowed for the adversary. The construction is done in the following two steps. First, we construct 1-bounded certified everlasting FE with *non-adaptive* security for all **P**/**poly** circuits from a certified everlasting garbling scheme and certified everlasting PKE in a black-box way (Section 7.2). Second, we change it to the *adaptively-secure* one by using certified everlasting RNCE in a black-box way (Section 7.3).
6. We construct q-bounded certified everlasting FE with adaptive security for all $\mathbf{NC}^1$ circuits, where q-bounded means that the total number of key queries is bounded by an a priori fixed polynomial q. This is constructed from the 1-bounded one constructed in Step. 5 by using multi-party computation in a black-box way (Section 7.4).

1.3 Related Works

Unruh [Unr15] introduced the concept of revocable quantum time-released encryption. In this primitive, a receiver with possession of quantum encrypted data can obtain its plaintext after predetermined time T. The sender can revoke the quantum encrypted data before time T. If the revocation succeeds, the receiver cannot obtain the information of the plaintext even if its computing power becomes unbounded.

Broadbent and Islam [BI20] constructed one-time SKE with certified deletion. This is ordinary one-time SKE, but once the receiver issues a valid classical certificate, the receiver cannot obtain the information of plaintext even if it later obtains the secret key of the ciphertext. (See also [KT20]).

Hiroka, Morimae, Nishimaki, and Yamakawa [HMNY21b] constructed reusable SKE, PKE, and attribute-based encryption (ABE) with certified deletion. These reusable SKE, PKE, and ABE with certified deletion are ordinary reusable SKE, PKE, and ABE, respectively. However, once the receiver issues a valid classical certificate, the receiver cannot obtain the information of plaintext even if it obtains some secret information (e.g. the master secret key of ABE). Note that, in these primitives, the security holds against computationally bounded adversaries unlike the present paper.

Hiroka, Morimae, Nishimaki, and Yamakawa [HMNY21a] constructed commitments with statistical binding and certified everlasting hiding. From it, they also constructed certified everlasting zero-knowledge proof for QMA based on the zero-knowledge protocol of [BG20].

Poremba [Por22] constructed fully homomorphic encryption (FHE) with certified deletion where the security holds against only semi-honest adversaries that behaves maliciously only after outputting a certificate.

1.4 Concurrent and Independent Work

There is a concurrent and independent work. Recently, Bartusek and Khurana have uploaded their paper on arXiv [BK22] where a generic compiler is introduced. The generic compiler can change many cryptographic primitives into ones with certified deletion, such as PKE, ABE, FHE, witness encryption, and timed-release encryption.

Their constructions via the generic compiler achieve classical certificates without QROM, which is an advantage of their results. On the other hand, we construct certified everlasting garbling schemes and FE, which is not done in their work. In fact, it is not clear how to construct certified everlasting garbling schemes and certified everlasting FE via their generic compiler. For example, a natural construction of FE via their generic compiler would be as follows. The ciphertext consists of a classical part and a quantum part. The classical part is the ciphertext of ordinary FE whose plaintext is $m \oplus r$, and the quantum part is random BB84 states whose computational basis states encode r. The decryption key of the function f consists of a functional decryption key sk_f and the basis of the BB84 states. However, in this construction, a receiver with the ciphertext and the decryption key cannot obtain $f(m)$. This is because the receiver can obtain only $f(m \oplus r)$ and r, which cannot recover $f(m)$.

2 Preliminaries

2.1 Notations

Here we introduce basic notations we will use in this paper. $x \leftarrow X$ denotes selecting an element x from a finite set X uniformly at random, and $y \leftarrow A(x)$ denotes assigning to y the output of a quantum or probabilistic or deterministic algorithm A on an input x. When we explicitly show that A uses randomness r, we write $y \leftarrow A(x; r)$. When D is a distribution, $x \leftarrow D$ denotes sampling an element x from D. $y := z$ denotes that y is set, defined, or substituted by z. Let $[n] := \{1, \ldots, n\}$. Let λ be a security parameter. By $[N]_p$ we denote the set of all size-p subsets of $\{1, 2 \cdots, N\}$. For classical strings x and y, $x||y$ denotes the concatenation of x and y. For a bit string $s \in \{0,1\}^n$, s_i and $s[i]$ denotes the i-th bit of s. QPT stands for quantum polynomial time. PPT stands for (classical) probabilistic polynomial time. A function $f : \mathbb{N} \to \mathbb{R}$ is a negligible function if for any constant c, there exists $\lambda_0 \in \mathbb{N}$ such that for any $\lambda > \lambda_0$, $f(\lambda) < \lambda^{-c}$. We write $f(\lambda) \leq \mathsf{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function.

2.2 Quantum Computations

We assume familiarity with the basics of quantum computation and use standard notations. Let $\mathcal{Q}$ be the state space of a single qubit. I is the two-dimensional identity operator. X and Z are the Pauli X and Z operators, respectively. For an operator A acting on a single qubit and a bit string $x \in \{0,1\}^n$, we write A^x as $A^{x_1} \otimes A^{x_2} \otimes \cdots A^{x_n}$. The trace distance between two states ρ and σ is given by $\frac{1}{2}\|\rho - \sigma\|_{\mathrm{tr}}$, where $\|A\|_{\mathrm{tr}} := \mathrm{tr}\sqrt{A^\dagger A}$ is the trace norm. If $\frac{1}{2}\|\rho - \sigma\|_{\mathrm{tr}} \leq \epsilon$, we say that ρ and σ are ϵ-close. If $\epsilon = \mathsf{negl}(\lambda)$, then we say that ρ and σ are statistically indistinguishable.

Quantum Random Oracle. We use the quantum random oracle model (QROM) [BDF+11] to construct certified everlasting SKE and certified everlasting PKE in Sections 3.2 and 4.2, respectively. In the QROM, a uniformly random function with a certain domain and range is chosen at the beginning, and quantum access to this function is given to all parties including an adversary. Zhandry showed that quantum access to random functions can be efficiently simulatable by using so-called compressed random oracle technique [Zha19].

We review the one-way to hiding lemma [Unr15, AHU19], which is useful when analyzing schemes in the QROM. The following form of the lemma is based on [AHU19].

Lemma 2.1 (One-Way to Hiding Lemma [AHU19]). *Let $S \subseteq \mathcal{X}$ be a random subset of $\mathcal{X}$. Let $G, H : \mathcal{X} \to \mathcal{Y}$ be random functions satisfying $\forall x \notin S\ [G(x) = H(x)]$. Let z be a random classical bit string. (S, G, H, z may have an arbitrary joint distribution.) Let $\mathcal{A}$ be an oracle-aided quantum algorithm that makes at most q quantum queries. Let $\mathcal{B}$ be an algorithm that on input z chooses $i \leftarrow [q]$, runs $\mathcal{A}^H(z)$, measures $\mathcal{A}$'s i-th query, and outputs the measurement outcome. Then we have $\left|\Pr[\mathcal{A}^G(z) = 1] - \Pr[\mathcal{A}^H(z) = 1]\right| \leq 2q\sqrt{\Pr[\mathcal{B}^H(z) \in S]}$.*

Quantum Teleportation. We use quantum teleportation to prove that our construction of the FE scheme in Section 7.3 satisfies adaptive security.

Lemma 2.2 (Quantum Teleportation). *Suppose that we have N Bell pairs between registers A and B, i.e., $\frac{1}{\sqrt{2^N}}\sum_{s\in\{0,1\}^N} |s\rangle_A \otimes |s\rangle_B$, and let ρ be an arbitrary N-qubit quantum state in register C. Suppose that we measure j-th qubits of C and A in the Bell basis and let $(x_j, z_j) \in \{0,1\} \times \{0,1\}$ be the measurement outcome for all $j \in [N]$. Let $x := x_1||x_2||\cdots||x_N$ and $z := z_1||z_2||\cdots||z_N$. Then (x, z) is uniformly distributed over $\{0,1\}^N \times \{0,1\}^N$. Moreover, conditioned on the measurement outcome (x, z), the resulting state in B is $X^x Z^z \rho Z^z X^x$.*

CSS code. We explain basics of CSS codes. CSS codes are used only in the constructions of certified everlasting SKE and PKE (Section 3.3 and Section 4.3), and therefore readers who are not interested in these constructions can skip this paragraph. A CSS code with parameters q, k_1, k_2, t consists of two classical linear binary codes. One is a $[q, k_1]$ code C_1 [2] and the other is a $[q, k_2]$ code. Both C_1 and $C_2^\perp$ can correct up to t errors, and they satisfy $C_2 \subseteq C_1$. We require that the parity check matrices of C_1, C_2 are computable in polynomial time, and that error correction can be

[2] A $[q, k]$ code is a code consisting of 2^k codewords, each of length q. That is, a k-dimensional subspace of $\{0,1\}^q = \mathrm{GF}(2)^q$.

performed in polynomial time. Given two binary codes $C \subseteq D$, let $D/C := \{x \bmod C : x \in D\}$. Here, mod C is a linear polynomial-time operation on $\{0,1\}^q$ with the following three properties. First, $x \bmod C = x' \bmod C$ if and only if $x - x' \in C$ for any $x, x' \in \{0,1\}^q$. Second, for any binary code D such that $C \subseteq D$, $x \bmod C \in D$ for any $x \in D$. Third, (x mod C) mod C= x mod C for any $x \in \{0,1\}^q$.

2.3 Cryptographic Tools

In this section, we review the cryptographic tools used in this paper.

Lemma 2.3 (Difference Lemma [Sho04]). *Let A, B, F be events defined in some probability distribution, and suppose $\Pr[A \wedge \overline{F}] = \Pr[B \wedge \overline{F}]$. Then $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

Encryption with Certified Deletion. Broadbent and Islam [BI20] introduced the notion of encryption with certified deletion.

Definition 2.4 (One-Time SKE with Certified Deletion (Syntax) [BI20, HMNY21b]). *Let λ be a security parameter and let p, q and r be some polynomials. A one-time secret key encryption scheme with certified deletion is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)}$, key space $\mathcal{K} := \{0,1\}^{q(\lambda)}$ and deletion certificate space $\mathcal{D} := \{0,1\}^{r(\lambda)}$.*

$\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$: *The key generation algorithm takes as input the security parameter 1^λ, and outputs a secret key $\mathsf{sk} \in \mathcal{K}$.*

$\mathsf{Enc}(\mathsf{sk}, m) \to \mathsf{CT}$: *The encryption algorithm takes as input sk and a plaintext $m \in \mathcal{M}$, and outputs a ciphertext $\mathsf{CT} \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \to m'$ *or* $\perp$: *The decryption algorithm takes as input sk and CT, and outputs a plaintext $m' \in \mathcal{M}$ or $\perp$.*

$\mathsf{Del}(\mathsf{CT}) \to \mathsf{cert}$: *The deletion algorithm takes as input CT, and outputs a certification $\mathsf{cert} \in \mathcal{D}$.*

$\mathsf{Vrfy}(\mathsf{sk}, \mathsf{cert}) \to \top$ *or* $\perp$: *The verification algorithm takes sk and cert as input, and outputs $\top$ or $\perp$.*

We require that a one-time SKE scheme with certified deletion satisfies correctness defined below.

Definition 2.5 (Correctness for One-Time SKE with Certified Deletion). *There are three types of correctness, namely, decryption correctness, verification correctness, and modification correctness.*

Decryption Correctness: *There exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[m' \neq m \,\middle|\, \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{sk}, \mathsf{cert}) = \perp \,\middle|\, \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Modification Correctness: *There exists a negligible function* negl *and a QPT algorithm* Modify *such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{sk}, \mathsf{cert}^*) = \perp \,\middle|\, \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \mathsf{cert} \leftarrow \mathsf{Del}(Z^b X^a \mathsf{CT} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Modify}(a, b, \mathsf{cert}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 2.6. The original definition [BI20, HMNY21b] only considers decryption correctness and verification correctness. In this paper, we additionally require modification correctness. This is because we need modification correctness for the construction of FE in Section 7.3. In fact, the construction of [BI20] satisfies modification correctness as well.

We require that a one-time SKE with certified deletion satisfies certified deletion security defined below.

Definition 2.7 (Certified Deletion Security for One-Time SKE with Certified Deletion). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a one-time SKE scheme with certified deletion. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{otsk\text{-}cert\text{-}del}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against an unbounded adversary* $\mathcal{A}$.

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
3. *The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$ *and sends* CT *to* $\mathcal{A}$.
4. $\mathcal{A}$ *sends* cert *to the challenger.*
5. *The challenger computes* $\mathsf{Vrfy}(\mathsf{sk}, \mathsf{cert})$. *If the output is* $\bot$, *the challenger sends* $\bot$ *to* $\mathcal{A}$. *If the output is* $\top$, *the challenger sends* sk *to* $\mathcal{A}$.
6. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$. *This is the output of the experiment.*

We say that Σ *is OT-CD secure if, for any unbounded* $\mathcal{A}$, *it holds that*

$$\mathsf{Adv}^{\mathsf{otsk\text{-}cert\text{-}del}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{otsk\text{-}cert\text{-}del}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{otsk\text{-}cert\text{-}del}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Broadbent and Islam [BI20] showed that a one-time SKE scheme with certified deletion that satisfies the above correctness and security exists unconditionally.

Secret Key Encryption (SKE).

Definition 2.8 (Secret Key Encryption (Syntax)). *Let* λ *be a security parameter and let* p, q, r *and* s *be some polynomials. A secret key encryption scheme is a tuple of algorithms* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with plaintext space* $\mathcal{M} := \{0, 1\}^n$, *ciphertext space* $\mathcal{C} := \{0, 1\}^{p(\lambda)}$, *and secret key space* $\mathcal{SK} := \{0, 1\}^{q(\lambda)}$.

$\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$**:** *The key generation algorithm takes the security parameter* 1^λ *as input and outputs a secret key* $\mathsf{sk} \in \mathcal{SK}$.

$\mathsf{Enc}(\mathsf{sk}, m) \to \mathsf{CT}$**:** *The encryption algorithm takes* sk *and a plaintext* $m \in \mathcal{M}$ *as input, and outputs a ciphertext* $\mathsf{CT} \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \to m'$ *or* $\bot$**:** *The decryption algorithm takes* sk *and* CT *as input, and outputs a plaintext* $m' \in \mathcal{M}$ *or* $\bot$.

We require that a SKE scheme satisfies correctness defined below.

Definition 2.9 (Correctness for SKE). *There are two types of correctness, namely, decryption correctness and special correctness.*

Decryption Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \neq m \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Special Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_2, \mathsf{CT}) \neq \bot \;\middle|\; \begin{array}{l} \mathsf{sk}_2, \mathsf{sk}_1 \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 2.10. In the original definition of SKE schemes, only decryption correctness is required. In this paper, however, we additionally require special correctness. This is because we need special correctness for the construction of FE in Section 6.2. In fact, special correctness can be easily satisfied as well.

As security of SKE schemes, we consider OW-CPA security or IND-CPA security defined below.

Definition 2.11 (OW-CPA Security for SKE). *Let* ℓ *be a polynomial of the security parameter* λ*. Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a SKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ow\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda)$ *against a QPT adversary* $\mathcal{A}$*.*

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*
2. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ *and returns* CT *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
3. *The challenger samples* $(m^1, \cdots, m^\ell) \leftarrow \mathcal{M}^\ell$*, computes* $\mathsf{CT}^i \leftarrow \mathsf{Enc}(\mathsf{sk}, m^i)$ *for all* $i \in [\ell]$ *and sends* $\{\mathsf{CT}^i\}_{i\in[\ell]}$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ *and returns* CT *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
5. $\mathcal{A}$ *outputs* m'*.*
6. *The output of the experiment is* 1 *if* $m' = m^i$ *for some* $i \in [\ell]$*. Otherwise, the output of the experiment is* 0*.*

We say that the Σ *is OW-CPA secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ow\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr\left[\mathsf{Exp}^{\mathsf{ow\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Note that we assume $1/|\mathcal{M}|$ *is negligible.*

Definition 2.12 (IND-CPA Security for SKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a SKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$*.*

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*
2. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ *and returns* CT *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
3. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
4. *The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$ *and sends* CT *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$ *and returns* CT *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
6. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$*. This is the output of the experiment.*

We say that Σ *is IND-CPA secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

It is well-known that IND-CPA security implies OW-CPA security. A SKE scheme exists if there exists a pseudorandom function.

Public Key Encryption (PKE).

Definition 2.13 (Public Key Encryption (Syntax)). *Let λ be a security parameter and let p, q and r be some polynomials. A public key encryption scheme is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \{0,1\}^{p(\lambda)}$, public key space $\mathcal{PK} := \{0,1\}^{q(\lambda)}$ and secret key space $\mathcal{SK} := \{0,1\}^{r(\lambda)}$.*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:** *The key generation algorithm takes as input the security parameter 1^λ and outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.*

$\mathsf{Enc}(\mathsf{pk}, m) \to \mathsf{CT}$**:** *The encryption algorithm takes as input pk and a plaintext $m \in \mathcal{M}$, and outputs a ciphertext $\mathsf{CT} \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \to m'$ **or** $\perp$**:** *The decryption algorithm takes as input sk and CT, and outputs a plaintext m' or $\perp$.*

We require that a PKE scheme satisfies decryption correctness defined below.

Definition 2.14 (Decryption Correctness for PKE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \neq m \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

As security, we consider OW-CPA security or IND-CPA security defined below.

Definition 2.15 (OW-CPA Security for PKE). *Let ℓ be a polynomial of the security parameter λ. Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ow\text{-}cpa}}(\lambda)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.*
2. *The challenger samples $(m^1, \cdots, m^\ell) \leftarrow \mathcal{M}^\ell$, computes $\mathsf{CT}^i \leftarrow \mathsf{Enc}(\mathsf{pk}, m^i)$ for all $i \in [\ell]$ and sends $\{\mathsf{CT}^i\}_{i \in [\ell]}$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs m'.*
4. *The output of the experiment is 1 if $m' = m^i$ for some $i \in [\ell]$. Otherwise, the output of the experiment is 0.*

We say that Σ is OW-CPA secure if, for any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{ow\text{-}cpa}}(\lambda) := \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ow\text{-}cpa}}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda).$$

Note that we assume $1/|\mathcal{M}|$ is negligible.

Definition 2.16 (IND-CPA Security for PKE). *Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and sends pk to $\mathcal{A}$.*
2. *$\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.*
3. *The challenger computes $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$, and sends CT to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. This is the output of the experiment.*

We say that Σ is IND-CPA secure if, for any QPT $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(\lambda) := \left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ind\text{-}cpa}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

It is well known that IND-CPA security implies OW-CPA security. There are many IND-CPA secure PKE schemes against QPT adversaries under standard cryptographic assumptions. A famous one is Regev PKE scheme, which is IND-CPA secure if the learning with errors (LWE) assumption holds against QPT adversaries [Reg09]. See [Reg09, GPV08] for the LWE assumption and constructions of post-quantum PKE.

3 Certified Everlasting Secret Key Encryption

In Section 3.1, we define certified everlasting SKE. In Section 3.2 and Section 3.3, we construct a certified everlasting SKE scheme with and without QROM, respectively.

3.1 Definition

Definition 3.1 (Certified Everlasting SKE (Syntax)). *Let λ be a security parameter and let p, q, r and s be some polynomials. A certified everlasting SKE scheme is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)}$, secret key space $\mathcal{SK} := \{0,1\}^{q(\lambda)}$, verification key space $\mathcal{VK} := \{0,1\}^{r(\lambda)}$, and deletion certificate space $\mathcal{D} := \mathcal{Q}^{\otimes s(\lambda)}$.*

$\mathsf{KeyGen}(1^\lambda) \to \mathsf{sk}$**:** *The key generation algorithm takes the security parameter 1^λ as input and outputs a secret key $\mathsf{sk} \in \mathcal{SK}$.*

$\mathsf{Enc}(\mathsf{sk}, m) \to (\mathsf{vk}, \mathsf{CT})$**:** *The encryption algorithm takes sk and a plaintext $m \in \mathcal{M}$ as input, and outputs a verification key $\mathsf{vk} \in \mathcal{VK}$ and a ciphertext $\mathsf{CT} \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \to m'$ *or* $\bot$**:** *The decryption algorithm takes sk and CT as input, and outputs a plaintext $m' \in \mathcal{M}$ or $\bot$.*

$\mathsf{Del}(\mathsf{CT}) \to \mathsf{cert}$**:** *The deletion algorithm takes CT as input, and outputs a certification $\mathsf{cert} \in \mathcal{D}$.*

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \to \top$ **or** $\bot$**:** *The verification algorithm takes vk and cert as input, and outputs $\top$ or $\bot$.*

We require that a certified everlasting SKE scheme satisfies correctness defined below.

Definition 3.2 (Correctness for Certified Everlasting SKE). *There are four types of correctness, namely, decryption correctness, verification correctness, special correctness, and modification correctness.*

Decryption Correctness: *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[m' \neq m \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \bot \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\mathsf{CT}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Special Correctness: *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_2, \mathsf{CT}) \neq \bot \;\middle|\; \begin{array}{l} \mathsf{sk}_2, \mathsf{sk}_1 \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}_1, m) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Modification Correctness: *There exists a negligible function negl and a QPT algorithm Modify such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \mathsf{cert} \leftarrow \mathsf{Del}(Z^b X^a \mathsf{CT} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Modify}(a, b, \mathsf{cert}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Remark 3.3. Minimum requirements for correctness are decryption correctness and verification correctness. In this paper, however, we also require special correctness and modification correctness, because we need special correctness for the construction of the garbling scheme in Section 6.2, and modification correctness for the construction of functional encryption in Section 7.3.

As security, we consider two definitions, Definition 3.4 and Definition 3.5 given below. The former is just the ordinal IND-CPA security and the latter is the certified everlasting security that we newly define in this paper. Roughly, the everlasting security guarantees that any QPT adversary cannot obtain plaintext information even if it becomes computationally unbounded and obtains the secret key after it issues a valid certificate.

Definition 3.4 (IND-CPA Security for Certified Everlasting SKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting SKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$*.*

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*
2. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$*, and returns* $(\mathsf{vk}, \mathsf{CT})$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
3. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
4. *The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$*, and sends* CT *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$*, and returns* $(\mathsf{vk}, \mathsf{CT})$ *to* $\mathcal{A}$*.* $\mathcal{A}$ *can repeat this process polynomially many times.*
6. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$*. This is the output of the experiment.*

We say that Σ *is IND-CPA secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 3.5 (Certified Everlasting IND-CPA Security for Certified Everlasting SKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting SKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}_1$ *and an unbounded adversary* $\mathcal{A}_2$*.*

1. *The challenger computes* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda)$*.*
2. $\mathcal{A}_1$ *sends an encryption query* m *to the challenger. The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$*, and returns* $(\mathsf{vk}, \mathsf{CT})$ *to* $\mathcal{A}_1$*.* $\mathcal{A}_1$ *can repeat this process polynomially many times.*
3. $\mathcal{A}_1$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
4. *The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m_b)$*, and sends* CT *to* $\mathcal{A}_1$*.*
5. $\mathcal{A}_1$ *sends an encryption query* m *to the challenger. The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$*, and returns* $(\mathsf{vk}, \mathsf{CT})$ *to* $\mathcal{A}_1$*.* $\mathcal{A}_1$ *can repeat this process polynomially many times.*
6. *At some point,* $\mathcal{A}_1$ *sends* cert *to the challenger and sends the internal state to* $\mathcal{A}_2$*.*
7. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$*. If the output is* $\bot$*, the challenger outputs* $\bot$*, and sends* $\bot$ *to* $\mathcal{A}_2$*. Otherwise, the challenger outputs* $\top$*, and sends* sk *to* $\mathcal{A}_2$*.*
8. $\mathcal{A}_2$ *outputs* $b' \in \{0, 1\}$*.*
9. *If the challenger outputs* $\top$*, then the output of the experiment is* b'*. Otherwise, the output of the experiment is* $\bot$*.*

We say that Σ *is certified everlasting IND-CPA secure if, for any QPT* $\mathcal{A}_1$ *and any unbounded* $\mathcal{A}_2$*, it holds that*

$$\mathsf{Adv}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

3.2 Construction with QROM

In this section, we construct a certified everlasting SKE scheme with QROM. Our construction is similar to that of the certified everlasting commitment scheme in [HMNY21a]. The difference is that we use SKE instead of commitment.

Our certified everlasting SKE scheme. We construct a certified everlasting SKE scheme $\Sigma_{\mathsf{cesk}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ from the following primitives.

- A one-time SKE with certified deletion scheme (Definition 2.4) $\Sigma_{\mathsf{skcd}} = \mathsf{CD}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$.
- A SKE scheme (Definition 2.8) $\Sigma_{\mathsf{sk}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\{0,1\}^\lambda$.
- A hash function H modeled as a quantum random oracle.

$\mathsf{KeyGen}(1^\lambda)$**:**

- Generate $\mathsf{ske.sk} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- Output $\mathsf{sk} := \mathsf{ske.sk}$.

$\mathsf{Enc}(\mathsf{sk}, m)$**:**

- Parse $\mathsf{sk} = \mathsf{ske.sk}$.
- Generate $\mathsf{cd.sk} \leftarrow \mathsf{CD.KeyGen}(1^\lambda)$ and $R \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{ske.CT} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, R)$.
- Compute $h := H(R) \oplus \mathsf{cd.sk}$ and $\mathsf{cd.CT} \leftarrow \mathsf{CD.Enc}(\mathsf{cd.sk}, m)$.
- Output $\mathsf{CT} := (h, \mathsf{ske.CT}, \mathsf{cd.CT})$ and $\mathsf{vk} := \mathsf{cd.sk}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$**:**

- Parse $\mathsf{sk} = \mathsf{ske.sk}$ and $\mathsf{CT} = (h, \mathsf{ske.CT}, \mathsf{cd.CT})$.
- Compute R' **or** $\perp \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}, \mathsf{ske.CT})$. If it outputs $\perp$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$ outputs $\perp$.
- Compute $\mathsf{cd.sk}' := H(R') \oplus h$.
- Compute $m' \leftarrow \mathsf{CD.Dec}(\mathsf{cd.sk}', \mathsf{cd.CT})$.
- Output m'.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = (h, \mathsf{ske.CT}, \mathsf{cd.CT})$.
- Compute $\mathsf{cd.cert} \leftarrow \mathsf{CD.Del}(\mathsf{cd.CT})$.
- Output $\mathsf{cert} := \mathsf{cd.cert}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = \mathsf{cd.sk}$ and $\mathsf{cert} = \mathsf{cd.cert}$.
- Compute $b \leftarrow \mathsf{SKE.Vrfy}(\mathsf{cd.sk}, \mathsf{cd.cert})$.
- Output b.

Correctness: It is easy to see that correctness of Σ_{cesk} comes from those of Σ_{sk} and Σ_{skcd}.

Security: The following two theorems hold.

Theorem 3.6. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.11) and Σ_{skcd} satisfies the OT-CD security (Definition 2.4), Σ_{cesk} satisfies the IND-CPA security (Definition 3.4).*

Its proof is similar to that of Theorem 3.7, and therefore we omit it.

Theorem 3.7. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.11) and Σ_{skcd} satisfies the OT-CD security (Definition 2.4), Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition 3.5).*

Its proof is similar to that of [HMNY21a, Theorem 5.8].

3.3 Construction without QROM

In this section, we construct a certified everlasting SKE scheme without QROM. Note that unlike the construction with QROM (Section 3.2), in this construction the plaintext space is of constant size. However, the size can be extended to the polynomial size via the standard hybrid argument. Our construction is similar to that of revocable quantum timed-release encryption in [Unr15]. The difference is that we use SKE instead of timed-release encryption.

Our certified everlasting SKE scheme without QROM. Let k_1 and k_2 be constants such that $k_1 > k_2$. Let p, q, r, s and t be polynomials. Let (C_1, C_2) be a CSS code with parameters q, k_1, k_2, t. We construct a certified everlasting SKE scheme $\Sigma_{\mathsf{cesk}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ with plaintext space $\mathcal{M} = C_1/C_2$ (isomorphic to $\{0,1\}^{k_1-k_2}$), ciphertext space $\mathcal{C} = \mathcal{Q}^{\otimes(p(\lambda)+q(\lambda))} \times \{0,1\}^{r(\lambda)} \times \{0,1\}^{q(\lambda)}/C_1 \times C_1/C_2$, secret key space $\mathcal{SK} = \{0,1\}^{s(\lambda)}$, verification key space $\mathcal{VK} = \{0,1\}^{p(\lambda)} \times [p(\lambda)+q(\lambda)]_{p(\lambda)} \times \{0,1\}^{p(\lambda)}$ and deletion certificate space $\mathcal{D} = \mathcal{Q}^{\otimes(p(\lambda)+q(\lambda))}$ from the following primitive.

- A SKE scheme (Definition 2.8) $\Sigma_{\mathsf{sk}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} = \{0,1\}^{p(\lambda)} \times [p(\lambda) + q(\lambda)]_{p(\lambda)} \times \{0,1\}^{p(\lambda)} \times C_1/C_2$, secret key space $\mathcal{SK} = \{0,1\}^{s(\lambda)}$ and ciphertext space $\mathcal{C} = \{0,1\}^{r(\lambda)}$.

The construction is as follows. (We will omit the security parameter below.)

$\mathsf{KeyGen}(1^\lambda)$:

- Generate $\mathsf{ske.sk} \leftarrow \mathsf{SKE}.\mathsf{KeyGen}(1^\lambda)$.
- Output $\mathsf{sk} := \mathsf{ske.sk}$.

$\mathsf{Enc}(\mathsf{sk}, m)$:

- Parse $\mathsf{sk} = \mathsf{ske.sk}$.
- Generate $B \leftarrow \{0,1\}^p$, $Q \leftarrow [p+q]_p$, $y \leftarrow C_1/C_2$, $u \leftarrow \{0,1\}^q/C_1$, $r \leftarrow \{0,1\}^p$, $x \leftarrow C_1/C_2$, $w \leftarrow C_2$.
- Compute $\mathsf{ske.CT} \leftarrow \mathsf{SKE}.\mathsf{Enc}(\mathsf{ske.sk}, (B, Q, r, y))$.
- Let U_Q be the unitary that permutes the qubits in Q into the first half of the system. (I.e., $U_Q \left|x_1 x_2 \cdots x_{p+q}\right\rangle = \left|x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\right\rangle$ with $Q := \{a_1, a_2, \cdots, a_p\}$ and $\{1, 2, \cdots, p+q\} \backslash Q := \{b_1, b_2, \cdots, b_q\}$.)
- Construct a quantum state $|\Psi\rangle := U_Q^\dagger (H^B \otimes I^{\otimes q})(|r\rangle \otimes |x \oplus w \oplus u\rangle)$.
- Compute $h := m \oplus x \oplus y$.
- Output $\mathsf{CT} := (|\Psi\rangle, \mathsf{ske.CT}, u, h)$ and $\mathsf{vk} := (B, Q, r)$.

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

- Parse $\mathsf{sk} = \mathsf{ske.sk}$, $\mathsf{CT} = (|\Psi\rangle, \mathsf{ske.CT}, u, h)$.
- Compute $(B, Q, r, y)/\bot \leftarrow \mathsf{SKE}.\mathsf{Dec}(\mathsf{ske.sk}, \mathsf{ske.CT})$. If $\bot \leftarrow \mathsf{SKE}.\mathsf{Dec}(\mathsf{ske.sk}, \mathsf{ske.CT})$, $\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$ outputs $\bot$ and aborts.

- Apply U_Q to $|\Psi\rangle$, measure the last q-qubits in the computational basis and obtain the measurement outcome $\gamma \in \{0,1\}^q$.
- Compute $x := \gamma \oplus u \bmod C_2$.
- Output $m' := h \oplus x \oplus y$.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = (|\Psi\rangle, \mathsf{ske.CT}, u, h)$.
- Output $\mathsf{cert} := |\Psi\rangle$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = (B, Q, r)$ and $\mathsf{cert} = |\Psi\rangle$.
- Apply $(H^B \otimes I^{\otimes q})U_Q$ to $|\Psi\rangle$, measure the first p-qubits in the computational basis and obtain the measurement outcome $r' \in \{0,1\}^p$.
- Output $\top$ if $r = r'$ and output $\bot$ otherwise.

Correctness. Correctness easily follows from that of Σ_{sk}.

Security. The following two theorems hold.

Theorem 3.8. *If Σ_{sk} is IND-CPA secure (Definition 2.12), then Σ_{cesk} is IND-CPA secure (Definition 3.4).*

Its proof is straightforward, so we omit it.

Theorem 3.9. *If Σ_{sk} is IND-CPA secure (Definition 2.12) and $tp/(p+q) - 4(k_1 - k_2)\ln 2$ is superlogarithmic, then Σ_{cesk} is certified everlasting IND-CPA secure (Definition 3.5).*

Its proof is similar to that of [Unr15, Theorem 3].

Note that the plaintext space is of constant size in our construction. However, via the standard hybrid argument , we can extend it to the polynomial size.

4 Certified Everlasting Public Key Encryption

In Section 4.1, we define certified everlasting PKE. In Section 4.2 and Section 4.3, we construct a certified everlasting PKE scheme with and without QROM, respectively.

4.1 Definition

Definition 4.1 (Certified Everlasting PKE). *Let λ be a security parameter and let p, q, r, s and t be polynomials. A certified everlasting PKE scheme is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)}$, public key space $\mathcal{PK} := \{0,1\}^{q(\lambda)}$, secret key space $\mathcal{SK} := \{0,1\}^{r(\lambda)}$, verification key space $\mathcal{VK} := \{0,1\}^{s(\lambda)}$ and deletion certificate space $\mathcal{D} := \mathcal{Q}^{\otimes t(\lambda)}$.*

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$**:** *The key generation algorithm takes the security parameter 1^λ as input and outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.*

$\mathsf{Enc}(\mathsf{pk}, m) \to (\mathsf{vk}, \mathsf{CT})$**:** *The encryption algorithm takes pk and a plaintext $m \in \mathcal{M}$ as input, and outputs a verification key $\mathsf{vk} \in \mathcal{VK}$ and a ciphertext $\mathsf{CT} \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \to m'$ *or* $\bot$**:** *The decryption algorithm takes sk and CT as input, and outputs a plaintext $m' \in \mathcal{M}$ or $\bot$.*

$\mathsf{Del}(\mathsf{CT}) \to \mathsf{cert}$**:** *The deletion algorithm takes CT as input and outputs a certification $\mathsf{cert} \in \mathcal{D}$.*

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \to \top$ **or** $\bot$**:** *The verification algorithm takes* vk *and* cert *as input, and outputs* $\top$ *or* $\bot$.

We require that a certified everlasting PKE scheme satisfies correctness defined below.

Definition 4.2 (Correctness for Certified Everlasting PKE). *There are three types of correctness, namely, decryption correctness, verification correctness, and modification correctness.*

Decryption Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[m' \neq m \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \bot \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Modification Correctness: *There exists a negligible function* negl *and a QPT algorithm* Modify *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \mathsf{cert} \leftarrow \mathsf{Del}(Z^b X^a \mathsf{CT} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Modify}(a, b, \mathsf{cert}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 4.3. Minimum requirements for correctness are decryption correctness and verification correctness. In this paper, however, we also require modification correctness, because we need modification correctness for the construction of functional encryption in Section 7.3.

As security, we consider two definitions, Definition 4.4 and Definition 4.5 given below. The former is just the ordinal IND-CPA security and the latter is the certified everlasting security that we newly define in this paper. Roughly, the everlasting security guarantees that any QPT adversary cannot obtain plaintext information even if it becomes computationally unbounded and obtains the secret key after it issues a valid certificate.

Definition 4.4 (IND-CPA Security for Certified Everlasting PKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting PKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$.

1. *The challenger generates* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
3. *The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$*, and sends* CT *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$. *This is the output of the experiment.*

We say that the Σ *is IND-CPA secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 4.5 (Certified Everlasting IND-CPA Security for Certified Everlasting PKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting PKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}_1$ *and an unbounded adversary* $\mathcal{A}_2$.

1. *The challenger computes* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, and sends* pk *to* $\mathcal{A}_1$.

2. $\mathcal{A}_1$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*

3. *The challenger computes* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$*, and sends* CT *to* $\mathcal{A}_1$.

4. *At some point,* $\mathcal{A}_1$ *sends* cert *to the challenger, and sends the internal state to* $\mathcal{A}_2$.

5. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$*. If the output is* $\perp$*, the challenger outputs* $\perp$*, and sends* $\perp$ *to* $\mathcal{A}_2$*. Otherwise, the challenger outputs* $\top$*, and sends* sk *to* $\mathcal{A}_2$.

6. $\mathcal{A}_2$ *outputs* $b' \in \{0,1\}$.

7. *If the challenger outputs* $\top$*, then the output of the experiment is* b'*. Otherwise, the output of the experiment is* $\perp$.

We say that the Σ *is certified everlasting IND-CPA secure if for any QPT* $\mathcal{A}_1$ *and any unbounded* $\mathcal{A}_2$*, it holds that*

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}(\lambda) := \left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

4.2 Construction with QROM

In this section, we construct a certified everlasting PKE scheme with QROM. Our construction is similar to that of the certified everlasting commitment scheme in [HMNY21a]. The difference is that we use PKE instead of commitment.

Our certified everlasting PKE scheme. We construct a certified everlasting PKE scheme $\Sigma_{\mathsf{cepk}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ from a one-time SKE with certified deletion scheme $\Sigma_{\mathsf{skcd}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ (Definition 2.4), a PKE scheme $\Sigma_{\mathsf{pk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\{0,1\}^\lambda$ (Definition 2.13) and a hash function H modeled as quantum random oracle.

$\mathsf{KeyGen}(1^\lambda)$**:**

- Generate $(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- Output $\mathsf{pk} := \mathsf{pke.pk}$ and $\mathsf{sk} := \mathsf{pke.sk}$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:**

- Parse $\mathsf{pk} = \mathsf{pke.pk}$.
- Generate $\mathsf{ske.sk} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- Randomly generate $R \leftarrow \{0,1\}^\lambda$.
- Compute $\mathsf{pke.CT} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}, R)$.
- Compute $h := H(R) \oplus \mathsf{ske.sk}$ and $\mathsf{ske.CT} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, m)$.
- Output $\mathsf{CT} := (h, \mathsf{ske.CT}, \mathsf{pke.CT})$ and $\mathsf{vk} := \mathsf{ske.sk}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$**:**

- Parse $\mathsf{sk} = \mathsf{pke.sk}$ and $\mathsf{CT} = (h, \mathsf{ske.CT}, \mathsf{pke.CT})$.
- Compute $R' \leftarrow \mathsf{PKE.Dec}(\mathsf{pke.sk}, \mathsf{pke.CT})$.
- Compute $\mathsf{ske.sk}' := h \oplus H(R')$.
- Compute $m' \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}', \mathsf{ske.CT})$.
- Output m'.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = (h, \mathsf{ske.CT}, \mathsf{pke.CT})$.
- Compute $\mathsf{ske.cert} \leftarrow \mathsf{SKE.Del}(\mathsf{ske.CT})$.
- Output $\mathsf{cert} := \mathsf{ske.cert}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = \mathsf{ske.sk}$ and $\mathsf{cert} = \mathsf{ske.cert}$.
- Compute $b \leftarrow \mathsf{SKE.Vrfy}(\mathsf{ske.sk}, \mathsf{ske.cert})$.
- Output b.

Correctness: Correctness easily follows from those of Σ_{pk} and Σ_{skcd}.

Security: The following two theorems hold. Their proofs are similar to those of Theorems 3.6 and 3.7, and therefore we omit them.

Theorem 4.6. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.15) and Σ_{skcd} satisfies the OT-CD security (Definition 2.7), Σ_{cepk} is IND-CPA secure (Definition 4.4).*

Theorem 4.7. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.15) and Σ_{skcd} satisfies the OT-CD security (Definition 2.7), Σ_{cepk} is certified everlasting IND-CPA secure (Definition 4.5).*

4.3 Construction without QROM

In this section, we construct a certified everlasting PKE scheme without QROM. Our construction is similar to that of quantum timed-release encryption presented in [Unr15]. The difference is that we use PKE instead of timed-release encryption.

Our certified everlasting PKE scheme without QROM. Let k_1 and k_2 be some constant such that $k_1 > k_2$. Let p, q, r, s, t and u be some polynomials. Let (C_1, C_2) be a CSS code with parameters q, k_1, k_2, t. We construct a certified everlasting PKE scheme $\Sigma_{\mathsf{cepk}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$, with plaintext space $\mathcal{M} = C_1/C_2$ (isomorphic $\{0,1\}^{(k_1-k_2)}$), ciphertext space $\mathcal{C} = \mathcal{Q}^{\otimes(p(\lambda)+q(\lambda))} \times \{0,1\}^{r(\lambda)} \times \{0,1\}^{q(\lambda)}/C_1 \times C_1/C_2$, public key space $\mathcal{PK} = \{0,1\}^{u(\lambda)}$, secret key space $\mathcal{SK} = \{0,1\}^{s(\lambda)}$, verification key space $\mathcal{VK} = \{0,1\}^{p(\lambda)} \times [p(\lambda)+q(\lambda)]_{p(\lambda)} \times \{0,1\}^{p(\lambda)}$ and deletion certificate space $\mathcal{D} = \mathcal{Q}^{\otimes(p(\lambda)+q(\lambda))}$ from the following primitive.

- A PKE scheme (Definition 2.13)$\Sigma_{\mathsf{pk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M} = \{0,1\}^{p(\lambda)} \times [p(\lambda)+q(\lambda)]_{p(\lambda)} \times \{0,1\}^{p(\lambda)} \times C_1/C_2$, public key space $\mathcal{PK} = \{0,1\}^{u(\lambda)}$, secret key space $\mathcal{SK} = \{0,1\}^{s(\lambda)}$ and ciphertext space $\mathcal{C} = \{0,1\}^{r(\lambda)}$.

The construction is as follows. (We will omit the security parameter below.)

$\mathsf{KeyGen}(1^\lambda)$**:**

- Generate $(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$.
- Output $\mathsf{pk} := \mathsf{pke.pk}$ and $\mathsf{sk} := \mathsf{pke.sk}$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:**

- Parse $\mathsf{pk} = \mathsf{pke.pk}$.
- Generate $B \leftarrow \{0,1\}^p$, $Q \leftarrow [p+q]_p$, $y \leftarrow C_1/C_2$, $u \leftarrow \{0,1\}^q/C_1$, $r \leftarrow \{0,1\}^p$, $x \leftarrow C_1/C_2$, $w \leftarrow C_2$.
- Compute $\mathsf{pke.CT} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}, (B, Q, r, y))$.
- Let U_Q be the unitary that permutes the qubits in Q into the first half of the system. (I.e., $U_Q |x_1 x_2 \cdots x_{p+q}\rangle = |x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\rangle$ with $Q := \{a_1, a_2, \cdots, a_p\}$ and $\{1, 2, \cdots, p+q\} \setminus Q := \{b_1, b_2, \cdots, b_q\}$.)

- Generate a quantum state $|\Psi\rangle := U_Q^{\dagger}(H^B \otimes I^{\otimes q})(|r\rangle \otimes |x \oplus w \oplus u\rangle)$.
- Compute $h := m \oplus x \oplus y$.
- Output $\mathsf{CT} := (|\Psi\rangle, \mathsf{pke.CT}, u, h)$ and $\mathsf{vk} := (B, Q, r)$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$**:**

- Parse $\mathsf{sk} = \mathsf{pke.sk}$ and $\mathsf{CT} = (|\Psi\rangle, \mathsf{pke.CT}, u, h)$.
- Compute $(B, Q, r, y) \leftarrow \mathsf{PKE.Dec}(\mathsf{pke.sk}, \mathsf{pke.CT})$.
- Apply U_Q to $|\Psi\rangle$, measure the last q-qubits in the computational basis and obtain the measurement outcome γ.
- Compute $x := \gamma \oplus u \bmod C_2$.
- Output $m' := h \oplus x \oplus y$.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = (|\Psi\rangle, \mathsf{pke.CT}, u, h)$.
- Output $\mathsf{cert} := |\Psi\rangle$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = (B, Q, r)$ and $\mathsf{cert} = |\Psi\rangle$.
- Apply $(H^B \otimes I^{\otimes q})U_Q$ to $|\Psi\rangle$, measure the first p-qubits in the computational basis and obtain the measurement outcome r'.
- Output $\top$ if $r = r'$ and output $\bot$ otherwise.

Correctness. Correctness easily follows from that of Σ_{pk}.

Security. The following two theorems hold.

Theorem 4.8. *If Σ_{pk} is IND-CPA secure (Definition 2.16), then Σ_{cepk} is IND-CPA secure (Definition 4.4).*

Its proof is straightforward, and therefore we omit it.

Theorem 4.9. *If Σ_{pk} is IND-CPA secure (Definition 2.16) and $tp/(p+q) - 4(k_1 - k_2)\ln 2$ is superlogarithmic, then Σ_{cepk} is certified everlasting IND-CPA secure (Definition 4.5).*

Its proof is similar to that of [Unr15, Theorem 3].

Note that the plaintext space is of constant size in our construction. However, via the standard hybrid argument, we can extend it to the polynomial size.

5 Certified Everlasting Receiver Non-Committing Encryption

In this section, we define and construct certified everlasting receiver non-committing encryption. In Section 5.1, we define certified everlasting RNCE. In Section 5.2, we construct a certified everlasting RNCE scheme from certified everlasting PKE (Section 4).

5.1 Definition

Definition 5.1 (Certified Everlasting RNCE (Syntax)). *Let λ be the security parameter and let p, q, r, s, t, u, and v be polynomials. A certified everlasting RNCE scheme is a tuple of algorithms* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal}, \mathsf{Del}, \mathsf{Vrfy})$ *with plaintext space* $\mathcal{M} := \{0,1\}^n$, *ciphertext space* $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)}$, *public key space* $\mathcal{PK} := \{0,1\}^{q(\lambda)}$, *master secret key space* $\mathcal{MSK} := \{0,1\}^{r(\lambda)}$, *secret key space* $\mathcal{SK} := \{0,1\}^{s(\lambda)}$, *verification key space* $\mathcal{VK} := \{0,1\}^{t(\lambda)}$, *deletion certificate space* $\mathcal{D} := \mathcal{Q}^{u(\lambda)}$, *and auxiliary state space* $\mathcal{AUX} := \{0,1\}^{v(\lambda)}$.

$\mathsf{Setup}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{MSK})$**:** *The setup algorithm takes the security parameter* 1^λ *as input, and outputs a public key* $\mathsf{pk} \in \mathcal{PK}$ *and a master secret key* $\mathsf{MSK} \in \mathcal{MSK}$.

$\mathsf{KeyGen}(\mathsf{MSK}) \rightarrow \mathsf{sk}$**:** *The key generation algorithm takes the master secret key* MSK *as input, and outputs a secret key* $\mathsf{sk} \in \mathcal{SK}$.

$\mathsf{Enc}(\mathsf{pk}, m) \rightarrow (\mathsf{vk}, \mathsf{CT})$**:** *The encryption algorithm takes* pk *and a plaintext* $m \in \mathcal{M}$ *as input, and outputs a verification key* $\mathsf{vk} \in \mathcal{VK}$ *and a ciphertext* $\mathsf{CT} \in \mathcal{C}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \rightarrow m'$ *or* $\bot$**:** *The decryption algorithm takes* sk *and* CT *as input, and outputs a plaintext* $m' \in \mathcal{M}$ *or* $\bot$.

$\mathsf{Fake}(\mathsf{pk}) \rightarrow (\mathsf{vk}, \widetilde{\mathsf{CT}}, \mathsf{aux})$**:** *The fake encryption algorithm takes* pk *as input, and outputs a verification key* $\mathsf{vk} \in \mathcal{VK}$, *a fake ciphertext* $\widetilde{\mathsf{CT}} \in \mathcal{C}$ *and an auxiliary state* $\mathsf{aux} \in \mathcal{AUX}$.

$\mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m) \rightarrow \widetilde{\mathsf{sk}}$**:** *The reveal algorithm takes* $\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}$ *and* m *as input, and outputs a fake secret key* $\widetilde{\mathsf{sk}} \in \mathcal{SK}$.

$\mathsf{Del}(\mathsf{CT}) \rightarrow \mathsf{cert}$**:** *The deletion algorithm takes* CT *as input and outputs a certification* $\mathsf{cert} \in \mathcal{D}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \rightarrow \top$ **or** $\bot$**:** *The verification algorithm takes* vk *and* cert *as input, and outputs* $\top$ *or* $\bot$.

We require that a certified everlasting RNCE scheme satisfies correctness defined below.

Definition 5.2 (Correctness for Certified Everlasting RNCE). *There are two types of correctness, namely, decryption correctness and verification correctness.*

Decryption Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[m' \neq m \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}) \\ m' \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \bot \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\mathsf{CT}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

As security, we consider two definitions, Definition 5.3 and Definition 5.4 given below. The former is just the ordinal receiver non-committing security and the latter is the certified everlasting security that we newly define in this paper. Roughly, the everlasting security guarantees that any QPT adversary cannot distinguish whether the ciphertext and the secret key are properly generated or not even if it becomes computationally unbounded and obtains the master secret key after it issues a valid certificate.

Definition 5.3 (Receiver Non-Committing (RNC) Security for Certified Everlasting RNCE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting RNCE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{rec}\text{-}\mathsf{nc}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$.

1. *The challenger runs* $(\mathsf{pk}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends* $m \in \mathcal{M}$ *to the challenger.*
3. *The challenger does the following:*
   - *If* $b = 0$*, the challenger generates* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ *and* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK})$*, and sends* $(\mathsf{CT}, \mathsf{sk})$ *to* $\mathcal{A}$.
   - *If* $b = 1$*, the challenger generates* $(\mathsf{vk}, \widetilde{\mathsf{CT}}, \mathsf{aux}) \leftarrow \mathsf{Fake}(\mathsf{pk})$ *and* $\widetilde{\mathsf{sk}} \leftarrow \mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m)$*, and sends* $(\widetilde{\mathsf{CT}}, \widetilde{\mathsf{sk}})$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$.

We say that Σ *is RNC secure if, for any QPT* $\mathcal{A}$*, it holds that*

$$\mathsf{Adv}^{\mathsf{rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 5.4 (Certified Everlasting RNC Security for Certified Everlasting RNCE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting RNCE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}_1$ *and an unbounded adversary* $\mathcal{A}_2$.

1. *The challenger runs* $(\mathsf{pk}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}_1$.
2. $\mathcal{A}_1$ *sends* $m \in \mathcal{M}$ *to the challenger.*
3. *The challenger does the following:*
   - *If* $b = 0$*, the challenger generates* $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ *and* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK})$*, and sends* $(\mathsf{CT}, \mathsf{sk})$ *to* $\mathcal{A}_1$.
   - *If* $b = 1$*, the challenger generates* $(\mathsf{vk}, \widetilde{\mathsf{CT}}, \mathsf{aux}) \leftarrow \mathsf{Fake}(\mathsf{pk})$ *and* $\widetilde{\mathsf{sk}} \leftarrow \mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m)$*, and sends* $(\widetilde{\mathsf{CT}}, \widetilde{\mathsf{sk}})$ *to* $\mathcal{A}_1$.
4. *At some point,* $\mathcal{A}_1$ *sends* cert *to the challenger and its internal state to* $\mathcal{A}_2$.
5. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$*. If the output is* $\top$*, the challenger outputs* $\top$ *and sends* MSK *to* $\mathcal{A}_2$*. If the output is* $\bot$*, the challenger outputs* $\bot$ *and sends* $\bot$ *to* $\mathcal{A}_2$.
6. $\mathcal{A}_2$ *outputs* $b' \in \{0, 1\}$.
7. *If the challenger outputs* $\top$*, then the output of the experiment is* b'*. Otherwise, the output of the experiment is* $\bot$.

We say that Σ *is certified everlasting RNC secure if for any QPT* $\mathcal{A}_1$ *and any unbounded* $\mathcal{A}_2$*, it holds that*

$$\mathsf{Adv}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

5.2 Construction

In this section, we construct a certified everlasting RNCE scheme from a certified everlasting PKE scheme (Definition 4.1). Our construction is similar to that of the secret-key RNCE scheme presented in [KNTY19]. The difference is that we use a certified everlasting PKE scheme instead of an ordinary SKE scheme.

Our certified everlasting RNCE scheme. We construct a certified everlasting RNCE scheme $\Sigma_{\mathsf{cence}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal}, \mathsf{Del}, \mathsf{Vrfy})$ from a certified everlasting PKE scheme $\Sigma_{\mathsf{cepk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$, which was introduced in Definition 4.1.

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

- Generate $(\mathsf{pke.pk}_{i,\alpha}, \mathsf{pke.sk}_{i,\alpha}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for all $i \in [n]$ and $\alpha \in \{0,1\}$.
- Output $\mathsf{pk} := \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ and $\mathsf{MSK} := \{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.

$\mathsf{KeyGen}(\mathsf{MSK})$**:**

- Parse $\mathsf{MSK} = \{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Generate $x \leftarrow \{0,1\}^n$.
- Output $\mathsf{sk} := (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]})$.

$\mathsf{Enc}(\mathsf{pk}, m)$**:**

- Parse $\mathsf{pk} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.CT}_{i,\alpha}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,\alpha}, m[i])$ for all $i \in [n]$ and $\alpha \in \{0,1\}$.
- Output $\mathsf{vk} := \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ and $\mathsf{CT} := \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{CT})$**:**

- Parse $\mathsf{sk} = (x, \{\mathsf{pke.sk}_i\}_{i\in[n]})$ and $\mathsf{CT} = \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $m[i] \leftarrow \mathsf{PKE.Dec}(\mathsf{pke.sk}_i, \mathsf{pke.CT}_{i,x[i]})$ for all $i \in [n]$.
- Output $m := m[1]||m[2]||\cdots||m[n]$.

$\mathsf{Fake}(\mathsf{pk})$**:**

- Parse $\mathsf{pk} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Generate $x^* \leftarrow \{0,1\}^n$.
- Compute $(\mathsf{pke.vk}_{i,x^*[i]}, \mathsf{pke.CT}_{i,x^*[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x^*[i]}, 0)$ and $(\mathsf{pke.vk}_{i,x^*[i]\oplus 1}, \mathsf{pke.CT}_{i,x^*[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x^*[i]\oplus 1}, 1)$ for all $i \in [n]$.
- Output $\mathsf{vk} := \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$, $\widetilde{\mathsf{CT}} := \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ and $\mathsf{aux} = x^*$.

$\mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m)$**:**

- Parse $\mathsf{pk} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$, $\mathsf{MSK} = \{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ and $\mathsf{aux} = x^*$.
- Output $\widetilde{\mathsf{sk}} := \left(x^* \oplus m, \{\mathsf{pke.sk}_{i,x^*[i]\oplus m[i]}\}_{i\in[n]}\right)$.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $\mathsf{pke.cert}_{i,\alpha} \leftarrow \mathsf{PKE.Del}(\mathsf{pke.CT}_{i,\alpha})$ for all $i \in [n]$ and $\alpha \in \{0,1\}$.
- Output $\mathsf{cert} := \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ and $\mathsf{cert} = \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $\top/\bot \leftarrow \mathsf{PKE.Vrfy}(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.cert}_{i,\alpha})$ for all $i \in [n]$ and $\alpha \in \{0,1\}$. If all results are $\top$, $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$ outputs $\top$. Otherwise, it outputs $\bot$.

Correctness: Correctness easily follows from that of Σ_{cepk}.

Security: The following two theorems hold.

Theorem 5.5. *If Σ_{cepk} is IND-CPA secure (Definition 4.4), Σ_{cence} is RNC secure (Definition 5.3).*

Its proof is similar to that of Theorem 5.6, and therefore we omit it.

Theorem 5.6. *If Σ_{cepk} is certified everlasting IND-CPA secure (Definition 4.5), Σ_{cence} is certified everlasting RNC secure (Definition 5.4).*

Its proof is given in Appendix A.

6 Certified Everlasting Garbling Scheme

In Section 6.1, we define certified everlasting garbling scheme. In Section 6.2, we construct a certified everlasting garbling scheme from a certified everlasting SKE scheme.

6.1 Definition

We define certified everlasting garbling schemes below. An important difference from ordinal classical garbling schemes is that the garbled circuit $\tilde{C}$ (i.e., an output of Grbl) is a quantum state.

Definition 6.1 (Certified Everlasting Garbling Scheme (Syntax)). *Let λ be a security parameter and p, q, r and s be polynomials. Let $\mathcal{C}_n$ be a family of circuits that take n-bit inputs. A certified everlasting garbling scheme is a tuple of algorithms $\Sigma = (\mathsf{Samp}, \mathsf{Grbl}, \mathsf{Eval}, \mathsf{Del}, \mathsf{Vrfy})$ with label space $\mathcal{L} := \{0,1\}^{p(\lambda)}$, garbled circuit space $\mathcal{C} := \mathcal{Q}^{\otimes q(\lambda)}$, verification key space $\mathcal{VK} := \{0,1\}^{r(\lambda)}$ and deletion certificate space $\mathcal{D} := \mathcal{Q}^{\otimes s(\lambda)}$.*

$\mathsf{Samp}(1^\lambda) \to \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$**:** *The sampling algorithm takes a security parameter 1^λ as input, and outputs $2n$ labels $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ with $L_{i,\alpha} \in \mathcal{L}$ for each $i \in [n]$ and $\alpha \in \{0,1\}$.*

$\mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}) \to (\widetilde{C}, \mathsf{vk})$**:** *The garbling algorithm takes 1^λ, a circuit $C \in \mathcal{C}_n$ and $2n$ labels $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ as input, and outputs a garbled circuit $\widetilde{C} \in \mathcal{C}$ and a verification key $\mathsf{vk} \in \mathcal{VK}$.*

$\mathsf{Eval}(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]}) \to y$**:** *The evaluation algorithm takes $\widetilde{C}$ and n labels $\{L_{i,x_i}\}_{i\in[n]}$ where $x_i \in \{0,1\}$ as input, and outputs y.*

$\mathsf{Del}(\widetilde{C}) \to \mathsf{cert}$**:** *The deletion algorithm takes $\widetilde{C}$ as input, and outputs a certificate $\mathsf{cert} \in \mathcal{D}$.*

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \to \top$ **or** $\bot$**:** *The verification algorithm takes vk and cert as input, and outputs $\top$ or $\bot$.*

We require that a certified everlasting garbling scheme satisfies correctness defined below.

Definition 6.2 (Correctness for Certified Everlasting Garbling Scheme). *There are three types of correctness, namely, evaluation correctness, verification correctness, and modification correctness.*

Evaluation Correctness: *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$,*

$$\Pr\left[y \neq C(x) \,\middle|\, \begin{array}{l} \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda) \\ (\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}) \\ y \leftarrow \mathsf{Eval}(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk},\mathsf{cert}) = \bot \;\middle|\; \begin{array}{l} \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda) \\ (\widetilde{C},\mathsf{vk}) \leftarrow \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\widetilde{C}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Modification Correctness: *There exists a negligible function* negl *and a QPT algorithm* Modify *such that for any* $\lambda \in \mathbb{N}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk},\mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda) \\ (\widetilde{C},\mathsf{vk}) \leftarrow \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}) \\ a,b \leftarrow \{0,1\}^{q(\lambda)} \\ \mathsf{cert} \leftarrow \mathsf{Del}(Z^b X^a \widetilde{C} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Modify}(a,b,\mathsf{cert}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 6.3. Minimum requirements for correctness are evaluation correctness and verification correctness. In this paper, however, we also require modification correctness, because we need modification correctness for the construction of functional encryption in Section 7.3.

As security, we consider two definitions, Definition 6.4 and Definition 6.5 given below. The former is just the ordinal selective security and the latter is the certified everlasting security that we newly define in this paper. Roughly, the everlasting security guarantees that any QPT adversary with the garbled circuit $\widetilde{C}$ and the labels $\{L_{i,x[i]}\}_{i\in[n]}$ cannot obtain any information beyond $C(x)$ even if it becomes computationally unbounded after it issues a valid certificate.

Definition 6.4 (Selective Security for Certified Everlasting Garbling Scheme). *Let* $\Sigma = (\mathsf{Samp},\mathsf{Grbl},\mathsf{Eval},\mathsf{Del},\mathsf{Vrfy})$ *be a certified everlasting garbling scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{selct}}_{\Sigma,\mathcal{A}}(1^\lambda, b)$ *against a QPT adversary* $\mathcal{A}$. *Let* Sim *be a QPT algorithm.*

1. *$\mathcal{A}$ sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ to the challenger.*
2. *The challenger computes $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda)$.*
3. *If $b = 0$, the challenger computes $(\widetilde{C},\mathsf{vk}) \leftarrow \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}})$, and returns $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}$. If $b = 1$, the challenger computes $\widetilde{C} \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i\in[n]})$, and returns $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. The experiment outputs b'.*

We say that Σ is selective secure if there exists a QPT simulator Sim *such that for any QPT adversary $\mathcal{A}$ it holds that*

$$\mathsf{Adv}^{\mathsf{selct}}_{\Sigma,\mathcal{A}}(\lambda) \coloneqq \left|\Pr\left[\mathsf{Exp}^{\mathsf{selct}}_{\Sigma,\mathcal{A}}(1^\lambda,0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{selct}}_{\Sigma,\mathcal{A}}(1^\lambda,1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 6.5 (Certified Everlasting Selective Security for Certified Everlasting Garbling Scheme). *Let* $\Sigma = (\mathsf{Samp},\mathsf{Grbl},\mathsf{Eval},\mathsf{Del},\mathsf{Vrfy})$ *be a certified everlasting garbling scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}selct}}_{\mathcal{A},\Sigma}(1^\lambda, b)$ *against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let* Sim *be a QPT algorithm.*

1. *$\mathcal{A}_1$ sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ to the challenger.*
2. *The challenger computes $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda)$.*
3. *If $b = 0$, the challenger computes $(\widetilde{C},\mathsf{vk}) \leftarrow \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}})$, and returns $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}_1$. If $b = 1$, the challenger computes $(\widetilde{C},\mathsf{vk}) \leftarrow \mathsf{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i\in[n]})$, and returns $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}_1$.*
4. *At some point, $\mathcal{A}_1$ sends* cert *to the challenger, and sends the internal state to $\mathcal{A}_2$.*

5. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$. *If the output is* $\bot$, *then the challenger outputs* $\bot$, *and sends* $\bot$ *to* $\mathcal{A}_2$. *Otherwise, the challenger outputs* $\top$, *and sends* $\top$ *to* $\mathcal{A}_2$.

6. $\mathcal{A}_2$ *outputs* $b' \in \{0,1\}$.

7. *If the challenger outputs* $\top$, *then the output of the experiment is* b'. *Otherwise, the output of the experiment is* $\bot$.

We say that Σ *is certified everlasting selective secure if there exists a QPT simulator* Sim *such that for any QPT* $\mathcal{A}_1$ *and any unbounded* $\mathcal{A}_2$ *it holds that*

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}selct}}(\lambda) \coloneqq \left|\Pr\left[\mathsf{Exp}_{\mathcal{A},\Sigma}^{\mathsf{cert\text{-}ever\text{-}selct}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A},\Sigma}^{\mathsf{cert\text{-}ever\text{-}selct}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

6.2 Construction

In this section, we construct a certified everlasting garbling scheme from a certified everlasting SKE scheme (Definition 3.1). Our construction is similar to Yao's construction of an ordinary garbling scheme [Yao86], but there are two important differences. First, we use a certified everlasting SKE scheme instead of an ordinary SKE scheme. Second, we use XOR secret sharing, although [Yao86] used double encryption. The reason why we cannot use double encryption is that our certified everlasting SKE scheme has quantum ciphertext and classical plaintext.

Before introducing our construction, let us quickly review notations for circuits. Let C be a boolean circuit. A boolean circuit C consists of gates, $\mathsf{gate}_1, \mathsf{gate}_2, \cdots, \mathsf{gate}_q$, where q is the number of gates in the circuit. Here, $\mathsf{gate}_i \coloneqq (g, w_a, w_b, w_c)$, where $g : \{0,1\}^2 \to \{0,1\}$ is a function, w_a, w_b are the incoming wires, and w_c is the outgoing wire. (The number of outgoing wires is not necessarily one. There can be many outgoing wires, but we use the same label w_c for all outgoing wires.) We say C is leveled if each gate has an associated level and any gate at level ℓ has incoming wires only from gates at level $\ell - 1$ and outgoing wires only to gates at level $\ell + 1$. Let $\mathsf{out}_1, \mathsf{out}_2, \cdots, \mathsf{out}_m$ be the m output wires. For any $x \in \{0,1\}^n$, $C(x)$ is the output of the circuit C on input x. We consider that $\mathsf{gate}_1, \mathsf{gate}_2, \cdots, \mathsf{gate}_q$ are arranged in the ascending order of the level.

Our certified everlasting garbling scheme. We construct a certified everlasting garbling scheme $\Sigma_{\mathsf{cegc}} = (\mathsf{Samp}, \mathsf{Grbl}, \mathsf{Eval}, \mathsf{Del}, \mathsf{Vrfy})$ from a certified everlasting SKE scheme $\Sigma_{\mathsf{cesk}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ (Definition 3.1). Let $\mathcal{K}$ be the key space of Σ_{cesk}. Let C be a leveled boolean circuit. Let n, m, q, and p be the input size, the output size, the number of gates, and the total number of wires of C, respectively.

$\mathsf{Samp}(1^\lambda)$**:**

- For each $i \in [n]$ and $\sigma \in \{0,1\}$, generate $\mathsf{ske.sk}_i^\sigma \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- Output $\{L_{i,\sigma}\}_{i\in[n],\sigma\in\{0,1\}} \coloneqq \{\mathsf{ske.sk}_i^\sigma\}_{i\in[n],\sigma\in\{0,1\}}$.

$\mathsf{Grbl}(1^\lambda, C, \{L_{i,\sigma}\}_{i\in[n],\sigma\in\{0,1\}})$**:**

- For each $i \in \{n+1, \cdots, p\}$ and $\sigma \in \{0,1\}$, generate $\mathsf{ske.sk}_i^\sigma \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
- For each $i \in [q]$, compute

$$(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}}),$$

  where $\mathsf{gate}_i = (g, w_a, w_b, w_c)$ and $\mathsf{GateGrbl}$ is described in Fig 1.
- For each $i \in [m]$, set $\widetilde{d}_i \coloneqq [(\mathsf{ske.sk}_{\mathsf{out}_i}^0, 0), (\mathsf{ske.sk}_{\mathsf{out}_i}^1, 1)]$.
- Output $\widetilde{C} \coloneqq (\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ and $\mathsf{vk} \coloneqq \{\mathsf{vk}_i\}_{i\in[q]}$.

$\mathsf{Eval}(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$**:**

- Parse $\widetilde{C} = (\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ and $\{L_{i,x_i}\}_{i\in[n]} = \{\mathsf{ske.sk}'_i\}_{i\in[n]}$.

- For each $i \in [q]$, compute $\mathsf{ske.sk}'_c \leftarrow \mathsf{GateEval}(\widetilde{g}_i, \mathsf{ske.sk}'_a, \mathsf{ske.sk}'_b)$ in the ascending order of the level, where $\mathsf{GateEval}$ is described in Fig 2. If $\mathsf{ske.sk}'_c = \bot$, output $\bot$ and abort.
- For each $i \in [m]$, set $y[i] = \sigma$ if $\mathsf{ske.sk}'_{\mathsf{out}_i} = \mathsf{ske.sk}^{\sigma}_{\mathsf{out}_i}$. Otherwise, set $y[i] = \bot$, and abort.
- Output $y := y[1]||y[2]||\cdots||y[m]$.

$\mathsf{Del}(\widetilde{C})$:

- Parse $\widetilde{C} = (\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$.
- For each $i \in [q]$, compute $\mathsf{cert}_i \leftarrow \mathsf{GateDel}(\widetilde{g}_i)$, where $\mathsf{GateDel}$ is described in Fig 3.
- Output $\mathsf{cert} := \{\mathsf{cert}_i\}_{i\in[q]}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$:

- Parse $\mathsf{vk} = \{\mathsf{vk}_i\}_{i\in[q]}$ and $\mathsf{cert} = \{\mathsf{cert}_i\}_{i\in[q]}$.
- For each $i \in [q]$, compute $\bot/\top \leftarrow \mathsf{GateVrfy}(\mathsf{vk}_i, \mathsf{cert}_i)$, where $\mathsf{GateVrfy}$ is described in Fig 4.
- If $\top \leftarrow \mathsf{GateVrfy}(\mathsf{vk}_i, \mathsf{cert}_i)$ for all $i \in [q]$, then output $\top$. Otherwise, output $\bot$.

Gate Garbling Circuit $\mathsf{GateGrbl}$

Input: $\mathsf{gate}_i, \{\mathsf{ske.sk}^{\sigma}_a, \mathsf{ske.sk}^{\sigma}_b, \mathsf{ske.sk}^{\sigma}_c\}_{\sigma\in\{0,1\}}$.

Output: $\widetilde{g}_i$ and vk_i.

1. Parse $\mathsf{gate}_i = (g, w_a, w_b, w_c)$.
2. Sample $\gamma_i \leftarrow \mathsf{S}_4$.[a]
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, sample $p^{\sigma_a,\sigma_b}_c \leftarrow \mathcal{K}$.
4. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $(\mathsf{ske.vk}^{\sigma_a,\sigma_b}_a, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_a) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^{\sigma_a}_a, p^{\sigma_a,\sigma_b}_c)$ and $(\mathsf{ske.vk}^{\sigma_a,\sigma_b}_b, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_b) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}^{\sigma_b}_b, p^{\sigma_a,\sigma_b}_c \oplus \mathsf{ske.sk}^{g(\sigma_a,\sigma_b)}_c)$.
5. Output $\widetilde{g}_i := \{\mathsf{ske.CT}^{\sigma_a,\sigma_b}_a, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_b\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in the permutated order of γ_i and $\mathsf{vk}_i := \{\mathsf{ske.vk}^{\sigma_a,\sigma_b}_a, \mathsf{ske.vk}^{\sigma_a,\sigma_b}_b\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in the permutated order of γ_i.

[a] S_4 is the symmetric group of order 4.

Figure 1: The description of $\mathsf{GateGrbl}$

Gate Evaluating Circuit $\mathsf{GateEval}$

Input: A garbled gate $\widetilde{g}_i$ and $(\mathsf{ske.sk}'_a, \mathsf{ske.sk}'_b)$.

Output: $\mathsf{ske.sk}_c$ or $\bot$.

1. Parse $\widetilde{g}_i = \{\mathsf{ske.CT}^{\sigma_a,\sigma_b}_a, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_b\}_{\sigma_a,\sigma_b\in\{0,1\}}$.
2. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $q^{\sigma_a,\sigma_b}_a \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}'_a, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_a)$ and $q^{\sigma_a,\sigma_b}_b \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk}'_b, \mathsf{ske.CT}^{\sigma_a,\sigma_b}_b)$.
3. If there exists a unique pair $(\sigma_a, \sigma_b) \in \{0,1\}^2$ such that $q^{\sigma_a,\sigma_b}_a \neq \bot$ and $q^{\sigma_a,\sigma_b}_b \neq \bot$, then compute $\mathsf{ske.sk}'^{\sigma_a,\sigma_b}_c := q^{\sigma_a,\sigma_b}_a \oplus q^{\sigma_a,\sigma_b}_b$ and output $\mathsf{ske.sk}'_c := \mathsf{ske.sk}'^{\sigma_a,\sigma_b}_c$. Otherwise, output $\mathsf{ske.sk}'_c := \bot$.

Figure 2: The description of $\mathsf{GateEval}$

Correctness: Correctness easily follows from that of Σ_{cesk}.

Gate Deletion Circuit $\mathsf{GateDel}$

Input: A garbled gate $\widetilde{g_i}$.

Output: cert_i

1. Parse $\widetilde{g_i} = \{\mathsf{ske.CT}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$.
2. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $\mathsf{ske.cert}_a^{\sigma_a,\sigma_b} \leftarrow \mathsf{SKE.Del}(\mathsf{ske.CT}_a^{\sigma_a,\sigma_b})$.
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $\mathsf{ske.cert}_b^{\sigma_a,\sigma_b} \leftarrow \mathsf{SKE.Del}(\mathsf{ske.CT}_b^{\sigma_a,\sigma_b})$.
4. Output $\mathsf{cert}_i := \{\mathsf{ske.cert}_a^{\sigma_a,\sigma_b}, \mathsf{ske.cert}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$.

Figure 3: The description of $\mathsf{GateDel}$

Gate Verification Circuit $\mathsf{GateVrfy}$

Input: vk_i and cert_i.

Output: $\top$ or $\bot$.

1. Parse $\mathsf{vk}_i = \{\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.vk}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$ and $\mathsf{cert}_i = \{\mathsf{ske.cert}_a^{\sigma_a,\sigma_b}, \mathsf{ske.cert}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$.
2. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $\top/\bot \leftarrow \mathsf{SKE.Vrfy}(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.cert}_a^{\sigma_a,\sigma_b})$.
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $\top/\bot \leftarrow \mathsf{SKE.Vrfy}(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske.cert}_b^{\sigma_a,\sigma_b})$.
4. If all the outputs are $\top$, then output $\top$. Otherwise, output $\bot$.

Figure 4: The description of $\mathsf{GateVrfy}$

Security: The following two theorems hold.

Theorem 6.6. *If Σ_{cesk} satisfies the IND-CPA security (Definition 3.4), Σ_{cegc} satisfies the selective security (Definition 6.4).*

Its proof is similar to that of Theorem 6.7, and therefore we omit it.

Theorem 6.7. *If Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition 3.5), Σ_{cegc} satisfies the certified everlasting selective security (Definition 6.5).*

Its proof is given in Appendix B.

7 Certified Everlasting Functional Encryption

In this section, we define and construct certified everlasting functional encryption (FE). In Section 7.1, we define certified everlasting FE. In Section 7.2, we construct a 1-bounded certified everlasting FE scheme with non-adaptive security for all $\mathbf{P}/\mathbf{poly}$ circuits from a certified everlasting garbling scheme (Definition 6.1) and a certified everlasting PKE scheme (Definition 4.1). In Section 7.3, we change it to the adaptive one by using a certified everlasting RNCE scheme (Definition 5.1). In Section 7.4, we further change it to a q-bounded certified everlasting FE for all $\mathbf{NC}^1$ circuits by using a multipary computation scheme.

7.1 Definition

Definition 7.1 **(q-Bounded Certified Everlasting FE (Syntax)).** *Let λ be a security parameter and let p, r, s, t, u and v be polynomials. Let q be a polynomial of the security parameter λ. A q-bounded certified everlasting FE scheme for a class $\mathcal{F}$ of functions is a tuple of algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ with plaintext space $\mathcal{M} := \{0,1\}^n$, ciphertext space $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)}$, master public key space $\mathcal{MPK} := \{0,1\}^{r(\lambda)}$, master secret key space $\mathcal{MSK} := \{0,1\}^{s(\lambda)}$, secret key space $\mathcal{SK} := \{0,1\}^{t(\lambda)}$, verification key space $\mathcal{VK} := \{0,1\}^{u(\lambda)}$ and deletion certificate space $\mathcal{D} := \mathcal{Q}^{\otimes v(\lambda)}$.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$**:** *The setup algorithm takes the security parameter 1^λ as input, and outputs a master public key $\mathsf{MPK} \in \mathcal{MPK}$ and a master secret key $\mathsf{MSK} \in \mathcal{MSK}$.*

$\mathsf{KeyGen}(\mathsf{MSK}, f) \to \mathsf{sk}_f$**:** *The keygeneration algorithm takes MSK and $f \in \mathcal{F}$ as input, and outputs a secret key $\mathsf{sk}_f \in \mathcal{SK}$.*

$\mathsf{Enc}(\mathsf{MPK}, m) \to (\mathsf{vk}, \mathsf{CT})$**:** *The encryption algorithm takes MPK and $m \in \mathcal{M}$ as input, and outputs a verification key $\mathsf{vk} \in \mathcal{VK}$ and a ciphertext $\mathsf{CT} \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}) \to y$ **or** $\bot$**:** *The decryption algorithm takes sk_f and CT as input, and outputs y or $\bot$.*

$\mathsf{Del}(\mathsf{CT}) \to \mathsf{cert}$**:** *The deletion algorithm takes CT as input, and outputs a certificate $\mathsf{cert} \in \mathcal{D}$.*

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \to \top$ **or** $\bot$**:** *The verification algorithm takes vk and cert as input, and outputs $\top$ or $\bot$.*

We require that a certified everlasting FE scheme satisfies correctness defined below.

Definition 7.2 (Correctness for Certified Everlasting FE). *There are three types of correctness, namely, evaluation correctness, verification correctness, and modification correctness.*

Evaluation Correctness: *There exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ and $f \in \mathcal{F}$,*

$$\Pr\left[y \neq f(m) \;\middle|\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m) \\ y \leftarrow \mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Verification Correctness: *There exists a negligible function* negl *such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) = \bot \;\middle|\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m) \\ \mathsf{cert} \leftarrow \mathsf{Del}(\mathsf{CT}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Modification Correctness: *There exists a negligible function* negl *and a QPT algorithm* Modify *such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ (\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \mathsf{cert} \leftarrow \mathsf{Del}(Z^b X^a \mathsf{CT} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Modify}(a, b, \mathsf{cert}) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Remark 7.3. Minimum requirements for correctness are evaluation correctness and verification correctness. In this paper, however, we also require modification correctness, because we need modification correctness for the construction of certified everlasting FE in Section 7.3.

Remark 7.4. In FE, we usually want to run Dec algorithm for many different functions f on the same ciphertext CT. One might think that the quantum CT is destroyed by Dec algorithm, and therefore it can be used only once. However, it is easy to see that Dec algorithm can be always modified so that it does not disturb the quantum state CT by using the gentle measurement lemma [Wil11].

In this paper, we introduce four types of definitions of security, Definitions 7.5 to 7.8. (We note that these securities are simulation based ones defined in [GVW12].) The first two definitions (Definitions 7.5 and 7.6) are ordinal security definitions of FE. (Definition 7.5 is non-adaptive one and Definition 7.6 is adaptive one.) The third and fourth definitions (Definitions 7.7 and 7.8) are security definitions with certified everlasting security that we newly introduce in this paper. (Definition 7.7 is non-adaptive one and Definition 7.8 is adaptive one.)

Definition 7.5 (**q-Bounded Non-Adaptive Security for Certified Everlasting FE (Simulation Base)[GVW12]**). *Let q be a polynomial of λ. Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ be a q-bounded certified everlasting FE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{non\text{-}adapt}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$. Let Sim be a QPT algorithm.*

1. *The challenger runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends MPK to $\mathcal{A}$.*
2. *$\mathcal{A}$ is allowed to make arbitrary key queries at most q times. For the ℓ-th key query, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$. Let q^* be the number of times that $\mathcal{A}$ makes key queries. Let $\mathcal{V} := \{y_i := f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i\in[q^*]}$.*
3. *$\mathcal{A}$ chooses $m \in \mathcal{M}$ and sends m to the challenger.*
4. *The experiment works as follows:*
   - *If $b = 0$, the challenger computes $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$, and sends CT to $\mathcal{A}$.*
   - *If $b = 1$, the challenger computes $\mathsf{CT} \leftarrow \mathsf{Sim}(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$, and sends CT to $\mathcal{A}$.*
5. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$. The output of the experiment is b'.*

We say that Σ is q-bounded non-adaptive secure if there exists a QPT simulator Sim such that for any QPT adversary $\mathcal{A}$ it holds that

$$\mathsf{Adv}^{\mathsf{non\text{-}adapt}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{non\text{-}adapt}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{non\text{-}adapt}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 7.6 (**q-Bounded Adaptive Security for Certified Everlasting FE (Simulation Base)[GVW12]**). *Let q be a polynomial of λ. Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ be a q-bounded certified everlasting FE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{adapt}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$. Let Sim_1 and Sim_2 be a QPT algorithm.*

1. *The challenger runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends MPK to $\mathcal{A}$.*
2. *$\mathcal{A}$ is allowed to make arbitrary key queries at most q times. For the ℓ-th key query, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$. Let q^* be the number of times that $\mathcal{A}$ makes key queries. Let $\mathcal{V} := \{y_i := f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i\in[q^*]}$.*
3. *$\mathcal{A}$ chooses $m \in \mathcal{M}$ and sends m to the challenger.*
4. *The experiment works as follows:*
   - *If $b = 0$, the challenger computes $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$, and sends CT to $\mathcal{A}$.*
   - *If $b = 1$, the challenger computes $(\mathsf{CT}, \mathsf{st}_{q^*}) \leftarrow \mathsf{Sim}_1(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$, and sends CT to $\mathcal{A}$, where st_{q^*} is a quantum state.*
5. *$\mathcal{A}$ is allowed to make arbitrary key queries at most $q - q^*$ times. For the ℓ-th key query, the challenger works as follows:*
   - *If $b = 0$, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$.*
   - *If $b = 1$, the challenger receives $f_\ell \in \mathcal{F}$, computes $(\mathsf{sk}_{f_\ell}, \mathsf{st}_\ell) \leftarrow \mathsf{Sim}_2(\mathsf{MSK}, f_\ell, f_\ell(m), \mathsf{st}_{\ell-1})$, and sends sk_{f_ℓ} to $\mathcal{A}$.*
6. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$. The output of the experiment is b'.*

We say that Σ is q-bounded adaptive secure if there exists a QPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ such that for any QPT adversary $\mathcal{A}$ it holds that

$$\mathsf{Adv}^{\mathsf{adapt}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{adapt}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{adapt}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 7.7 (q-Bounded Certified Everlasting Non-Adaptive Security for Certified Everlasting FE (Simulation Base)). *Let q be a polynomial of λ. Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ be a q-bounded certified everlasting FE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let Sim be a QPT algorithm.*

1. *The challenger runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends MPK to $\mathcal{A}_1$.*
2. *$\mathcal{A}_1$ is allowed to make arbitrary key queries at most q times. For the ℓ-th key query, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$ and sends sk_{f_ℓ} to $\mathcal{A}_1$. Let q^* be the number of times that $\mathcal{A}_1$ makes key queries. Let $\mathcal{V} \coloneqq \{y_i \coloneqq f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i\in[q^*]}$.*
3. *$\mathcal{A}_1$ chooses $m \in \mathcal{M}$ and sends m to the challenger.*
4. *The experiment works as follows:*
   - *If $b = 0$, the challenger computes $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$, and sends CT to $\mathcal{A}_1$.*
   - *If $b = 1$, the challenger computes $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Sim}(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$, and sends CT to $\mathcal{A}_1$.*
5. *At some point, $\mathcal{A}_1$ sends cert to the challenger and its internal state to $\mathcal{A}_2$.*
6. *The challenger computes $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$. If the output is $\top$, then the challenger outputs $\top$, and sends MSK to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$, and sends $\bot$ to $\mathcal{A}_2$.*
7. *$\mathcal{A}_2$ outputs $b' \in \{0,1\}$. If the challenger outputs $\top$, the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.*

We say that Σ is q-bounded certified everlasting non-adaptive secure if there exists a QPT simulator Sim such that for any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$ it holds that

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}(\lambda) \coloneqq \left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Definition 7.8 (q-Bounded Certified Everlasting Adaptive Security for Certified Everlasting FE (Simulation Base)). *Let q be a polynomial of λ. Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ be a q-bounded certified everlasting FE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}adapt}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let Sim_1, Sim_2, and Sim_3 be a QPT algorithm.*

1. *The challenger runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and sends MPK to $\mathcal{A}_1$.*
2. *$\mathcal{A}_1$ is allowed to make arbitrary key queries at most q times. For the ℓ-th key query, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$ and sends sk_{f_ℓ} to $\mathcal{A}_1$. Let q^* be the number of times that $\mathcal{A}_1$ makes key queries. Let $\mathcal{V} \coloneqq \{y_i \coloneqq f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i\in[q^*]}$.*
3. *$\mathcal{A}_1$ chooses $m \in \mathcal{M}$ and sends m to the challenger.*
4. *The experiment works as follows:*
   - *If $b = 0$, the challenger computes $(\mathsf{vk}, \mathsf{CT}) \leftarrow \mathsf{Enc}(\mathsf{MPK}, m)$, and sends CT to $\mathcal{A}_1$.*
   - *If $b = 1$, the challenger computes $(\mathsf{CT}, \mathsf{st}_{q^*}) \leftarrow \mathsf{Sim}_1(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$, and sends CT to $\mathcal{A}_1$, where st_{q^*} is a quantum state.*
5. *$\mathcal{A}_1$ is allowed to make arbitrary key queries at most $q - q^*$ times. For the ℓ-th key query, the challenger works as follows.*
   - *If $b = 0$, the challenger receives $f_\ell \in \mathcal{F}$, computes $\mathsf{sk}_{f_\ell} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}_1$.*
   - *If $b = 1$, the challenger receives $f_\ell \in \mathcal{F}$, computes $(\mathsf{sk}_{f_\ell}, \mathsf{st}_\ell) \leftarrow \mathsf{Sim}_2(\mathsf{MSK}, f_\ell, f_\ell(m), \mathsf{st}_{\ell-1})$, and sends sk_{f_ℓ} to $\mathcal{A}_1$, where st_ℓ is a quantum state.*

6. *If $b = 1$, the challenger runs $\mathsf{vk} \leftarrow \mathsf{Sim}_3(\mathsf{st}_{q'})$. Here, q' is the number of times that $\mathcal{A}_1$ makes key queries in total.*

7. *At some point, $\mathcal{A}_1$ sends cert to the challenger and its internal state to $\mathcal{A}_2$.*

8. *The challenger computes $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$. If the output is $\top$, then the challenger outputs $\top$, and sends MSK to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$, and sends $\bot$ to $\mathcal{A}_2$.*

9. *$\mathcal{A}_2$ outputs $b' \in \{0, 1\}$. If the challenger outputs $\top$, the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.*

We say that Σ is q-bounded certified everlasting adaptive secure if there exists a QPT simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ such that for any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$ it holds that

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}adapt}}(\lambda) := \left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}adapt}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}adapt}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

7.2 Construction of 1-Bounded Certified Everlasting Functional Encryption with Non-Adaptive Security

In this section, we construct a 1-bounded certified everlasting FE scheme with non-adaptive security from a certified everlasting garbling scheme (Definition 6.1) and a certified everlasting PKE scheme (Definition 4.1).

Our 1-bounded certified everlasting FE scheme with non-adaptive security. We use a universal circuit $U(\cdot, x)$ in which a plaintext x is hard-wired. The universal circuit takes a function f as input and outputs $f(x)$. Let $s := |f|$. We construct a 1-bounded certified everlasting FE scheme with non-adaptive security $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ from a certified everlasting garbling scheme $\Sigma_{\mathsf{cegc}} = \mathsf{GC}.(\mathsf{Samp}, \mathsf{Grbl}, \mathsf{Eval}, \mathsf{Del}, \mathsf{Vrfy})$ (Definition 6.1) and a certified everlasting PKE scheme $\Sigma_{\mathsf{cepk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ (Definition 4.1).

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

- Generate $(\mathsf{pke.pk}_{i,\alpha}, \mathsf{pke.sk}_{i,\alpha}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0, 1\}$.
- Output $\mathsf{MPK} := \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ and $\mathsf{MSK} := \{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$.

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

- Parse $\mathsf{MSK} = \{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ and $f = (f_1, \cdots, f_s)$.
- Output $\mathsf{sk}_f := (f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})$.

$\mathsf{Enc}(\mathsf{MPK}, m)$:

- Parse $\mathsf{MPK} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$.
- Compute $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$.
- Compute $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.
- For every $i \in [s]$ and $\alpha \in \{0, 1\}$, compute $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.CT}_{i,\alpha}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,\alpha}, L_{i,\alpha})$.
- Output $\mathsf{vk} := (\mathsf{gc.vk}, \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ and $\mathsf{CT} := (\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.

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

- Parse $\mathsf{sk}_f = (f, \{\mathsf{pke.sk}_i\}_{i\in[s]})$ and $\mathsf{CT} = (\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.
- For every $i \in [s]$, compute $L_i, \leftarrow \mathsf{PKE.Dec}(\mathsf{pke.sk}_i, \mathsf{pke.CT}_{i,f[i]})$.
- Compute $y \leftarrow \mathsf{GC.Eval}(\widetilde{U}, \{L_i\}_{i\in[s]})$.
- Output y.

$\mathsf{Del}(\mathsf{CT})$:

- Parse $\mathsf{CT} = (\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.
- Compute $\mathsf{gc.cert} \leftarrow \mathsf{GC.Del}(\widetilde{U})$.
- For every $i \in [s]$ and $\alpha \in \{0,1\}$, compute $\mathsf{pke.cert}_{i,\alpha} \leftarrow \mathsf{PKE.Del}(\mathsf{pke.CT}_{i,\alpha})$.
- Output $\mathsf{cert} := (\mathsf{gc.cert}, \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$:

- Parse $\mathsf{vk} = (\mathsf{gc.vk}, \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ and $\mathsf{cert} = (\mathsf{gc.cert}, \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.
- Output $\top$ if $\top \leftarrow \mathsf{GC.Vrfy}(\mathsf{gc.vk}, \mathsf{gc.cert})$ and $\top \leftarrow \mathsf{PKE.Vrfy}(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.cert}_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$. Otherwise, output $\bot$.

Correctness: Correctness easily follows from that of Σ_{cegc} and Σ_{cepk}.

Security: The following two theorems hold.

Theorem 7.9. *If Σ_{cegc} satisfies the selective security (Definition 6.4) and Σ_{cepk} satisfies the IND-CPA security (Definition 4.4), Σ_{cefe} satisfies the 1-bounded non-adaptive security (Definition 7.5).*

Its proof is similar to that of Theorem 7.10, and therefore we omit it.

Theorem 7.10. *If Σ_{cegc} satisfies the certified everlasting selective security (Definition 6.5) and Σ_{cepk} satisfies the certified everlasting IND-CPA security (Definition 4.5), Σ_{cefe} satisfies the 1-bounded certified everlasting non-adaptive security (Definition 7.7).*

Its proof is given in Appendix C.

7.3 Construction of 1-Bounded Certified Everlasting Functional Encryption with Adaptive Security

In this section, we change the non-adaptive scheme constructed in the previous subsection to the adaptive one by using a certified everlasting RNC scheme (Definition 5.1).

Our 1-bounded certified everlasting FE scheme with adaptive security. We construct a 1-bounded certified everlasting FE scheme with adaptive security $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ from a 1-bounded certified everlasting FE scheme with non-adaptive security $\Sigma_{\mathsf{nad}} = \mathsf{NAD}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$, where the ciphertext space is $\mathcal{C} := \mathcal{Q}^{\otimes n}$, and a certified everlasting RNCE scheme

$$\Sigma_{\mathsf{cence}} = \mathsf{NCE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal}, \mathsf{Del}, \mathsf{Vrfy})$$

(Definition 5.1). Let $\mathsf{NAD.Modify}$ be a QPT algorithm such that

$$\Pr\left[\mathsf{NAD.Vrfy}(\mathsf{nad.vk}, \mathsf{nad.cert}^*) \neq \top \;\middle|\; \begin{array}{l} (\mathsf{nad.MPK}, \mathsf{nad.MSK}) \leftarrow \mathsf{NAD.Setup}(1^\lambda) \\ (\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m) \\ a, c \leftarrow \{0,1\}^n \\ \mathsf{nad.cert} \leftarrow \mathsf{NAD.Del}(Z^c X^a \mathsf{nad.CT} X^a Z^c) \\ \mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(a, c, \mathsf{nad.cert}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

for any m.

Our construction is as follows.

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

- Run $(\mathsf{nad.MPK}, \mathsf{nad.MSK}) \leftarrow \mathsf{NAD.Setup}(1^\lambda)$.
- Run $(\mathsf{nce.pk}, \mathsf{nce.MSK}) \leftarrow \mathsf{NCE.Setup}(1^\lambda)$.
- Output $\mathsf{MPK} := (\mathsf{nad.MPK}, \mathsf{nce.pk})$ and $\mathsf{MSK} := (\mathsf{nad.MSK}, \mathsf{nce.MSK})$.

$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:**

- Parse $\mathsf{MSK} = (\mathsf{nad.MSK}, \mathsf{nce.MSK})$.
- Compute $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$.
- Compute $\mathsf{nce.sk} \leftarrow \mathsf{NCE.KeyGen}(\mathsf{nce.MSK})$.
- Output $\mathsf{sk}_f := (\mathsf{nad.sk}_f, \mathsf{nce.sk})$.

$\mathsf{Enc}(\mathsf{MPK}, m)$**:**

- Parse $\mathsf{MPK} = (\mathsf{nad.MPK}, \mathsf{nce.pk})$.
- Compute $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$.
- Generate $a, c \leftarrow \{0,1\}^n$. Let $\Psi := Z^c X^a \mathsf{nad.CT} X^a Z^c$.
- Compute $(\mathsf{nce.vk}, \mathsf{nce.CT}) \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}, (a, c))$.
- Output $\mathsf{vk} := (\mathsf{nad.vk}, \mathsf{nce.vk}, a, c)$ and $\mathsf{CT} := (\Psi, \mathsf{nce.CT})$.

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

- Parse $\mathsf{sk}_f = (\mathsf{nad.sk}_f, \mathsf{nce.sk})$ and $\mathsf{CT} = (\Psi, \mathsf{nce.CT})$.
- Compute $(a', c') \leftarrow \mathsf{NCE.Dec}(\mathsf{nce.sk}, \mathsf{nce.CT})$.
- Compute $\mathsf{nad.CT}' := X^{a'} Z^{c'} \Psi Z^{c'} X^{a'}$.
- Compute $y \leftarrow \mathsf{NAD.Dec}(\mathsf{nad.sk}_f, \mathsf{nad.CT}')$.
- Output y.

$\mathsf{Del}(\mathsf{CT})$**:**

- Parse $\mathsf{CT} = (\Psi, \mathsf{nce.CT})$.
- Compute $\mathsf{nad.cert} \leftarrow \mathsf{NAD.Del}(\Psi)$.
- Compute $\mathsf{nce.cert} \leftarrow \mathsf{NCE.Del}(\mathsf{nce.CT})$.
- Output $\mathsf{cert} := (\mathsf{nad.cert}, \mathsf{nce.cert})$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$**:**

- Parse $\mathsf{vk} = (\mathsf{nad.vk}, \mathsf{nce.vk}, a, c)$ and $\mathsf{cert} = (\mathsf{nad.cert}, \mathsf{nce.cert})$.
- Compute $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(a, c, \mathsf{nad.cert})$.
- Output $\top$ if $\top \leftarrow \mathsf{NCE.Vrfy}(\mathsf{nce.vk}, \mathsf{nce.cert})$ and $\top \leftarrow \mathsf{NAD.Vrfy}(\mathsf{nad.vk}, \mathsf{nad.cert}^*)$. Otherwise, output $\bot$.

Correctness: Correctness easily follows from that of Σ_{nad} and Σ_{cence}.

Security: The following two theorems hold.

Theorem 7.11. *If Σ_{nad} satisfies the 1-bounded non-adaptive security (Definition 7.5) and Σ_{cence} satisfies the RNC security (Definition 5.3), Σ_{cefe} satisfies the 1-bounded adaptive security (Definition 7.6).*

Its proof is similar to that of Theorem 7.12, and therefore we omit it.

Theorem 7.12. *If Σ_{nad} satisfies the 1-bounded certified everlasting non-adaptive security(Definition 7.7) and Σ_{cence} satisfies the certified everlasting RNC security (Definition 5.4), Σ_{cefe} satisfies the 1-bounded certified everlasting adaptive security (Definition 7.8).*

Its proof is given in Appendix D.

7.4 Construction of q-Bounded Certified Everlasting Functional Encryption for $\mathbf{NC}^1$ circuits

In this section, we construct a q-bounded certified everlasting FE for all $\mathbf{NC}^1$ circuits from 1-bounded certified everlasting FE constructed in the previous subsection and Shamir's secret sharing ([Sha79]). Our construction is similar to that of ordinary FE for all $\mathbf{NC}^1$ circuits in [GVW12] except that we use a 1-bounded certified everlasting FE instead of an ordinary 1-bounded FE.

Our q-bounded certified everlasting FE scheme for $\mathbf{NC}^1$ circuits. We consider the polynomial representation of circuits C in $\mathbf{NC}^1$. The input message space is $\mathcal{M} := \mathbb{F}^\ell$, and for each $\mathbf{NC}^1$ circuit C, $C(\cdot)$ is an ℓ-variate polynomial over $\mathbb{F}$ of total degree at most D. Let $q = q(\lambda)$ be a polynomial of λ. Our scheme is associated with additional parameters $S = S(\lambda)$, $N = N(\lambda)$, $t = t(\lambda)$ and $v = v(\lambda)$ that satisfy

$$t(\lambda) = \Theta(q^2\lambda), N(\lambda) = \Theta(D^2q^2t), v(\lambda) = \Theta(\lambda), S(\lambda) = \Theta(vq^2).$$

Let us define a family $\mathcal{G} := \{G_{C,\Delta}\}_{C\in\mathbf{NC}^1,\Delta\subseteq[S]}$, where

$$G_{C,\Delta}(x, Z_1, Z_2, \cdots, Z_S) := C(x) + \sum_{i\in\Delta} Z_i$$

is a function and $Z_1, \cdots, Z_S \in \mathbb{F}$.

We construct a q-bounded certified everlasting FE scheme for all $\mathbf{NC}^1$ circuits $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ from a 1-bounded certified everlasting FE scheme $\Sigma_{\mathsf{one}} = \mathsf{ONE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$.

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

- For $i \in [N]$, generate $(\mathsf{one.MPK}_i, \mathsf{one.MSK}_i) \leftarrow \mathsf{ONE.Setup}(1^\lambda)$.
- Output $\mathsf{MPK} := \{\mathsf{one.MPK}_i\}_{i\in[N]}$ and $\mathsf{MSK} := \{\mathsf{one.MSK}_i\}_{i\in[N]}$.

$\mathsf{KeyGen}(\mathsf{MSK}, C)$:

- Parse $\mathsf{MSK} = \{\mathsf{one.MSK}_i\}_{i\in[N]}$.
- Chooses a uniformly random set $\Gamma \subseteq [N]$ of size $tD + 1$.
- Chooses a uniformly random set $\Delta \subseteq [S]$ of size v.
- For $i \in \Gamma$, compute $\mathsf{one.sk}_{C,\Delta,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C,\Delta})$.
- Output $\mathsf{sk}_C := (\Gamma, \Delta, \{\mathsf{one.sk}_{C,\Delta,i}\}_{i\in\Gamma})$.

$\mathsf{Enc}(\mathsf{MPK}, x)$:

- Parse $\mathsf{MPK} = \{\mathsf{one.MPK}_i\}_{i\in[N]}$.
- For $i \in [\ell]$, pick a random degree t polynomial $\mu_i(\cdot)$ whose constant term is $x[i]$.
- For $i \in [S]$, pick a random degree Dt polynomial $\xi_i(\cdot)$ whose constant term is 0.
- For $i \in [N]$, compute $(\mathsf{one.vk}_i, \mathsf{one.CT}_i) \leftarrow \mathsf{ONE.Enc}(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)))$.
- Output $\mathsf{vk} = \{\mathsf{one.vk}_i\}_{i\in[N]}$ and $\mathsf{CT} := \{\mathsf{one.CT}_i\}_{i\in[N]}$.

$\mathsf{Dec}(\mathsf{sk}_C, \mathsf{CT})$:

- Parse $\mathsf{sk}_C = (\Gamma, \Delta, \{\mathsf{one.sk}_{C,\Delta,i}\}_{i\in\Gamma})$ and $\mathsf{CT} = \{\mathsf{one.CT}_i\}_{i\in[N]}$.
- For $i \in \Gamma$, compute $\eta(i) \leftarrow \mathsf{ONE.Dec}(\mathsf{one.sk}_{C,\Delta,i}, \mathsf{one.CT}_i)$.
- Output $\eta(0)$.

$\mathsf{Del}(\mathsf{CT})$:

- Parse $\mathsf{CT} = \{\mathsf{one.CT}_i\}_{i\in[N]}$.

- For $i \in [N]$, compute $\mathsf{one.cert}_i \leftarrow \mathsf{ONE.Del}(\mathsf{one.CT}_i)$.
- Output $\mathsf{cert} := \{\mathsf{one.cert}_i\}_{i\in[N]}$.

$\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert})$:

- Parse $\mathsf{vk} = \{\mathsf{one.vk}_i\}_{i\in[N]}$ and $\mathsf{cert} = \{\mathsf{one.cert}_i\}_{i\in[N]}$.
- For $i \in [N]$, compute $\top/\bot \leftarrow \mathsf{ONE.Vrfy}(\mathsf{one.vk}_i, \mathsf{one.cert}_i)$. If all results are $\top$, output $\top$. Otherwise, output $\bot$.

Correctness: Verification correctness easily follows from verification correctness of Σ_{one}. Let us show evaluation correctness. By decryption correctness of Σ_{one}, for all $i \in \Gamma$ we have

$$\begin{aligned}\eta(i) &= G_{C,\Delta}(\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)) \\ &= C(\mu_1(i), \cdots, \mu_\ell(i)) + \Sigma_{a\in\Delta}\xi_a(i).\end{aligned}$$

Since $|\Gamma| \geq Dt + 1$, this means that η is equal to the degree Dt polynomial

$$\eta(\cdot) = C(\mu_1(\cdot), \cdots, \mu_\ell(\cdot)) + \Sigma_{a\in\Delta}\xi_a(\cdot)$$

Hence $\eta(0) = C(x_1, \cdots, x_\ell) = C(x)$, which means that our construction satisfies evaluation correctness.

Security: The following two theorems hold.

Theorem 7.13. *If Σ_{one} satisfies the 1-bounded adaptive security, Σ_{cefe} satisfies the q-bounded adaptive security.*

Its proof is similar to that of Theorem 7.14, and therefore we omit it.

Theorem 7.14. *If Σ_{one} satisfies the 1-bounded certified everlasting adaptive security, Σ_{cefe} the q-bounded certified everlasting adaptive security.*

Its proof is given in Appendix E.

Acknowledgement

TM is supported by JST Moonshot JPMJMS2061-5-1-1, JST FOREST, MEXT QLEAP, the Grant-in-Aid for Scientific Research (B) No.JP19H04066, the Grant-in Aid for Transformative Research Areas (A) 21H05183, and the Grant-in-Aid for Scientific Research (A) No.22H00522

References

[Agr18] Shweta Agrawal. New methods for indistinguishability obfuscation: Bootstrapping and instantiation. Cryptology ePrint Archive, Report 2018/633, 2018. https://eprint.iacr.org/2018/633. (Cited on page 1.)

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2019. (Cited on page 3.)

[AJL+19] Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019. (Cited on page 1.)

[AJS18] Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615. (Cited on page 1.)

[AS17] Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017. (Cited on page 1.)

[BB84] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *IEEE International Conference on Computers Systems and Signal Processing*, pages 175–179. IEEE, 1984. (Cited on page 1.)

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011. (Cited on page 2, 3.)

[BG20] Anne Broadbent and Alex B. Grilo. QMA-hardness of consistency of local density matrices with applications to quantum zero-knowledge. In *61st FOCS*, pages 196–205. IEEE Computer Society Press, November 2020. (Cited on page 2.)

[BI20] Anne Broadbent and Rabib Islam. Quantum encryption with certified deletion. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 92–122. Springer, Heidelberg, November 2020. (Cited on page 1, 2, 4, 5.)

[BK22] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. *arXiv:2207.01754*, 2022. (Cited on page 2.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273. Springer, 2011. (Cited on page 1.)

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Cited on page 1.)

[GGHZ16] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 480–511. Springer, Heidelberg, January 2016. (Cited on page 1.)

[GMM17] Sanjam Garg, Mohammad Mahmoody, and Ameer Mohammed. Lower bounds on obfuscation from all-or-nothing encryption primitives. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 661–695. Springer, Heidelberg, August 2017. (Cited on page 1.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206. ACM, 2008. (Cited on page 7.)

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012. (Cited on page 1, 25, 26, 31, 53.)

[HMNY21a] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting zero-knowledge proof for QMA. *IACR Cryptol. ePrint Arch.*, 2021:1315, 2021. (Cited on page 1, 2, 10, 11, 14.)

[HMNY21b] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In *ASIACRYPT*, pages 606–636. Springer, 2021. (Cited on page 1, 2, 4, 5.)

[KNTY19] Fuyuki Kitagawa, Ryo Nishimaki, Keisuke Tanaka, and Takashi Yamakawa. Adaptively secure and succinct functional encryption: Improving security and efficiency, simultaneously. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 521–551. Springer, Heidelberg, August 2019. (Cited on page 18.)

[KT20] Srijita Kundu and Ernest Tan. Composably secure device-independent encryption with certified deletion. *arXiv*, 2011.12704, 2020. (Cited on page 1, 2.)

[LC97] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Phys. Rev. Lett.*, 78:3410–3413, 1997. (Cited on page 1.)

[LM18] Huijia Lin and Christian Matt. Pseudo flawed-smudging generators and their application to indistinguishability obfuscation. Cryptology ePrint Archive, Report 2018/646, 2018. https://eprint.iacr.org/2018/646. (Cited on page 1.)

[LT17] Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 630–660. Springer, Heidelberg, August 2017. (Cited on page 1.)

[May97] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, 78:3414–3417, 1997. (Cited on page 1.)

[MW18] Sanketh Menda and John Watrous. Oracle separations for quantum statistical zero-knowledge. *arXiv:1801.08967*, 2018. (Cited on page 1.)

[Por22] Alexander Poremba. Quantum proofs of deletion for learning with errors. *arXiv:2203.01610*, 2022. (Cited on page 1, 2.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009. (Cited on page 7.)

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. (Cited on page 1, 31.)

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. https://eprint.iacr.org/2004/332. (Cited on page 4.)

[Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. (Cited on page 1, 2, 3, 11, 12, 15, 16.)

[Wil11] Mark M Wilde. From classical to quantum shannon theory. *arXiv*, 1106.1445, 2011. (Cited on page 25.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 22.)

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2019. (Cited on page 3.)

A Proof of Theorem 5.6

Proof of Theorem 5.6. To prove the theorem, let us introduce the sequence of hybrids.

Hyb_0: This is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma_{\mathsf{cence}},\mathcal{A}}(\lambda, 0)$. For clarity, we describe the experiment against any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ is any QPT adversary and $\mathcal{A}_2$ is any unbounded adversary.

1. The challenger generates $(\mathsf{pke.pk}_{i,\alpha}, \mathsf{pke.sk}_{i,\alpha}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for all $i \in [n]$ and $\alpha \in \{0,1\}$.
2. The challenger sends $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
3. $\mathcal{A}_1$ sends $m \in \mathcal{M}$ to the challenger.
4. The challenger generates $x \leftarrow \{0,1\}^n$, computes $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.CT}_{i,\alpha}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,\alpha}, m[i])$ for all $i \in [n]$ and $\alpha \in \{0,1\}$, and sends $(\{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}))$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ sends $\{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to the challenger and its internal state to $\mathcal{A}_2$.
6. The challenger computes $\mathsf{PKE.Vrfy}(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.cert}_{i,\alpha})$ for all $i \in [n]$ and $\alpha \in \{0,1\}$. If all results are $\top$, the challenger outputs $\top$ and sends $\{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$ and sends $\bot$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.
8. If the challenger outputs $\top$, then the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.

Hyb_1: This is identical to Hyb_0 except that the challenger generates $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke.CT}_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i] \oplus 1)$ for all $i \in [n]$ instead of computing $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke.CT}_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i])$ for all $i \in [n]$.

Hyb_2: This is identical to Hyb_1 except for the following three points.

1. The challenger generates $x^* \leftarrow \{0,1\}^n$ instead of generating $x \leftarrow \{0,1\}^n$.
2. For all $i \in [n]$, the challenger generates $(\mathsf{pke.vk}_{i,x^*[i]}, \mathsf{pke.CT}_{i,x^*[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x^*[i]}, 0)$ and $(\mathsf{pke.vk}_{i,x^*[i]\oplus 1}, \mathsf{pke.CT}_{i,x^*[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x^*[i]\oplus 1}, 1)$ instead of computing $(\mathsf{pke.vk}_{i,x[i]}, \mathsf{pke.CT}_{i,x[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x[i]}, m[i])$ and $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke.CT}_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i] \oplus 1)$.
3. The challenger sends $(\{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x^* \oplus m, \{\mathsf{pke.sk}_{i,x^*[i]\oplus m[i]}\}_{i\in[n]}))$ to $\mathcal{A}_1$ instead of sending $(\{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}))$ to $\mathcal{A}_1$.

It is clear that Hyb_0 is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 0)$ and Hyb_2 is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, 1)$. Hence, Theorem 5.6 easily follows from the following Propositions A.1 and A.2 (whose proof is given later.). $\square$

Proposition A.1. *If Σ_{cepk} is certified everlasting IND-CPA secure, it holds that* $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda)$.

Proposition A.2. $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof of Proposition A.1. For the proof, we use Lemma A.3. We assume that $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$ defined in Lemma A.3. This contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma A.3. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ from the challenger of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$.
2. $\mathcal{B}$ sends $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$ and sends m to $\mathcal{B}$.
4. $\mathcal{B}$ generates $x \leftarrow \{0,1\}^n$ and sends $(x, m[1], \cdots, m[n], m[1] \oplus 1, \cdots, m[n] \oplus 1)$ to the challenger of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$.

5. $\mathcal{B}$ receives $(\{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}, \{\mathsf{pke.CT}_{i,x[i]\oplus 1}\}_{i\in[n]})$ from the challenger of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$.

6. $\mathcal{B}$ computes $(\{\mathsf{pke.vk}_{i,x[i]}\}_{i\in[n]}, \{\mathsf{pke.CT}_{i,x[i]}\}_{i\in[n]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,x[i]}, m[i])$ for $i \in [n]$.

7. $\mathcal{B}$ sends $(\{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}))$ to $\mathcal{A}_1$.

8. $\mathcal{A}_1$ sends $\{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to $\mathcal{B}$ and the internal state to $\mathcal{A}_2$.

9. $\mathcal{B}$ sends $\{\mathsf{pke.cert}_{i,x[i]\oplus 1}\}_{i\in[n]}$ to the challenger, and receives $\{\mathsf{pke.sk}_{i,x[i]\oplus 1}\}_{i\in[n]}$ or $\bot$. If $\mathcal{B}$ receives $\bot$, it outputs $\bot$ and aborts.

10. $\mathcal{B}$ sends $\{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$ to $\mathcal{A}_2$.

11. $\mathcal{A}_2$ outputs b'.

12. $\mathcal{B}$ computes $\mathsf{PKE.Vrfy}(\mathsf{pke.vk}_{i,x[i]}, \mathsf{pke.cert}_{i,x[i]})$ for all $i \in [n]$. If all results are $\top$, $\mathcal{B}$ outputs b'. Otherwise, $\mathcal{B}$ outputs $\bot$.

It is clear that $\Pr[1 \leftarrow \mathcal{B}|b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B}|b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible. Therefore, $|\Pr[1 \leftarrow \mathcal{B}|b = 0] - \Pr[1 \leftarrow \mathcal{B}|b = 1]|$ is also non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma A.3. $\square$

Proof of Proposition A.2. It is obvious that the joint probability distribution that $\mathcal{A}_1$ receives $(\{\mathsf{pke}, \mathsf{CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}))$ in Hyb_1 is identical to the joint probability distribution that $\mathcal{A}_1$ receives $(\{\mathsf{pke}, \mathsf{CT}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x^* \oplus m, \{\mathsf{pke.sk}_{i,x^*[i]\oplus m[i]}\}_{i\in[n]}))$ in Hyb_2. Hence, Proposition A.2 follows. $\square$

We use the following lemma for the proof of Theorem 5.6 and Theorem 6.7. The proof is shown with the standard hybrid argument. It is also easy to see that a similar lemma holds for IND-CPA security.

Lemma A.3. *Let s be some polynomial of the security parameter λ. Let $\Sigma := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ be a certified everlasting PKE scheme. Let us consider the following security experiment $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against $\mathcal{A}$ consisting of any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.*

1. *The challenge generates $(\mathsf{pk}_{i,\alpha}, \mathsf{sk}_{i,\alpha}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for all $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\mathsf{pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.*

2. *$\mathcal{A}_1$ chooses $f \in \{0,1\}^s$ and $(m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s])$ to the challenger.*

3. *The challenger computes $(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_b[i])$ for all $i \in [s]$, and sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{\mathsf{CT}_{i,f[i]\oplus 1}\}_{i\in[s]})$ to $\mathcal{A}_1$.*

4. *At some point, $\mathcal{A}_1$ sends $\{\mathsf{cert}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.*

5. *The challenger computes $\mathsf{Vrfy}(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{cert}_{i,f[i]\oplus 1})$ for every $i \in [s]$. If all results are $\top$, the challenger outputs $\top$, and sends $\{\mathsf{sk}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$, and sends $\bot$ to $\mathcal{A}_2$.*

6. *$\mathcal{A}_2$ outputs b'.*

7. *If the challenger outputs $\top$, then the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.*

If the Σ satisfies the certified everlasting IND-CPA security,

$$\mathsf{Adv}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\right]\right| \le \mathsf{negl}(\lambda)$$

for any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.

Proof of Lemma A.3. Let us consider the following hybrids for $j \in \{0, 1, \cdots, s\}$.

Hyb_j**:**

1. The challenger generates $(\mathsf{pk}_{i,\alpha}, \mathsf{sk}_{i,\alpha}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\mathsf{pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
2. $\mathcal{A}_1$ chooses $f \in \{0,1\}^s$ and $(m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s])$ to the challenger.
3. The challenger computes
$$(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_1[i])$$
for $i \in [j]$ and
$$(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_0[i])$$
for $i \in \{j+1, j+2, \cdots, s\}$, and sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{\mathsf{CT}_{i,f[i]\oplus 1}\}_{i\in[s]})$ to $\mathcal{A}_1$.
4. At some point, $\mathcal{A}_1$ sends $\{\mathsf{cert}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
5. The challenger computes $\mathsf{Vrfy}(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{cert}_{i,f[i]\oplus 1})$ for every $i \in [s]$. If all results are $\top$, the challenger outputs $\top$, and sends $\{\mathsf{sk}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$, and sends $\bot$ to $\mathcal{A}_2$.
6. $\mathcal{A}_2$ outputs b'.
7. If the challenger outputs $\top$, then the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.

It is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\Big[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, 0) = 1\Big]$ and $\Pr[\mathsf{Hyb}_s = 1] = \Pr\Big[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, 1) = 1\Big]$. Furthermore, we can show
$$\big|\Pr\big[\mathsf{Hyb}_j = 1\big] - \Pr\big[\mathsf{Hyb}_{j+1} = 1\big]\big| \leq \mathsf{negl}(\lambda)$$
for each $j \in \{0, 1, \cdots, s-1\}$. (Its proof is given below.) From these facts, we obtain Lemma A.3.

Let us show the remaining one. To show it, let us assume that $\big|\Pr\big[\mathsf{Hyb}_j = 1\big] - \Pr\big[\mathsf{Hyb}_{j+1} = 1\big]\big|$ is non-negligible. Then, we can construct an adversary $\mathcal{B}$ that can break the certified everlasting IND-CPA security of Σ as follows.

1. $\mathcal{B}$ receives pk from the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{B}}(\lambda, b)$.
2. $\mathcal{B}$ generates $\beta \leftarrow \{0,1\}$ and sets $\mathsf{pk}_{j+1,\beta} := \mathsf{pk}$.
3. $\mathcal{B}$ generates $(\mathsf{pk}_{i,\alpha}, \mathsf{sk}_{i,\alpha}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ for $i \in \{1, \cdots, j, j+2, \cdots, s\}$ and $\alpha \in \{0,1\}$, and $(\mathsf{pk}_{j+1,\beta\oplus 1}, \mathsf{sk}_{j+1,\beta\oplus 1}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
4. $\mathcal{B}$ sends $\{\mathsf{pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ chooses $f \in \{0,1\}^s$ and $(m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \cdots, m_0[s], m_1[1], m_1[2], \cdots, m_1[s])$ to the challenger.
6. If $f[j+1] = \beta$, $\mathcal{B}$ aborts the experiment, and outputs $\bot$.
7. $\mathcal{B}$ computes
$$(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_1[i])$$
for $i \in [j]$ and
$$(\mathsf{vk}_{i,f[i]\oplus 1}, \mathsf{CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_0[i])$$
for $i \in \{j+2, \cdots, s\}$.

8. $\mathcal{B}$ sends $(m_0[j+1], m_1[j+1])$ to the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}_{\Sigma,\mathcal{B}}(\lambda, b)$. The challenger computes $(\mathsf{vk}_{j+1,f[j+1]\oplus 1}, \mathsf{CT}_{j+1,f[j+1]\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{pk}_{j+1,f[j+1]\oplus 1}, m_b[j+1])$ and sends $\mathsf{CT}_{j+1,f[j+1]\oplus 1}$ to $\mathcal{B}$.
9. $\mathcal{B}$ sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{\mathsf{CT}_{i,f[i]\oplus 1}\}_{i\in[s]})$ to $\mathcal{A}_1$.
10. $\mathcal{A}_1$ sends $\{\mathsf{cert}_i\}_{i\in[s]}$ to $\mathcal{B}$, and sends its internal state to $\mathcal{A}_2$.
11. $\mathcal{B}$ sends cert_{j+1} to the challenger, and receives $\mathsf{sk}_{j+1,f[j+1]\oplus 1}$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.
12. $\mathcal{B}$ sends $\{\mathsf{sk}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to $\mathcal{A}_2$.
13. $\mathcal{A}_2$ outputs b'.
14. $\mathcal{B}$ computes Vrfy for all cert_i, and outputs b' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\perp$.

Since $\mathsf{pk}_{j+1,\beta}$ and $\mathsf{pk}_{j+1,\beta\oplus 1}$ are identically distributed, it holds that $\Pr[f[j+1]=\beta] = \Pr[f[j+1]=\beta\oplus 1] = \frac{1}{2}$. If $b=0$ and $f[j+1]=\beta\oplus 1$, $\mathcal{B}$ simulates Hyb_j. Therefore, we have

$$\begin{aligned}\Pr[1\leftarrow\mathcal{B}|b=0] &= \Pr[1\leftarrow\mathcal{B}\wedge f[j+1]=\beta\oplus 1|b=0]\\ &= \Pr[1\leftarrow\mathcal{B}|b=0, f[j+1]=\beta\oplus 1]\cdot\Pr[f[j+1]=\beta\oplus 1]\\ &= \frac{1}{2}\Pr\left[\mathsf{Hyb}_j=1\right].\end{aligned}$$

If $b=1$ and $f[j+1]=\beta\oplus 1$, $\mathcal{B}$ simulates Hyb_{j+1}. Similarly, we have $\Pr[1\leftarrow\mathcal{B}|b=1] = \frac{1}{2}\Pr\left[\mathsf{Hyb}_{j+1}=1\right]$. By assumption, $\left|\Pr\left[\mathsf{Hyb}_j=1\right]-\Pr\left[\mathsf{Hyb}_{j+1}=1\right]\right|$ is non-negligible, and therefore $|\Pr[1\leftarrow\mathcal{B}|b=0]-\Pr[1\leftarrow\mathcal{B}|b=1]|$ is non-negligible, which contradicts the certified everlasting IND-CPA security of Σ. ☐

B Proof of Theorem 6.7

Let $\widehat{\mathsf{gate}}_1, \widehat{\mathsf{gate}}_2, \cdots, \widehat{\mathsf{gate}}_q$ be the topology of the gates $\mathsf{gate}_1, \mathsf{gate}_2, \cdots, \mathsf{gate}_q$ which indicates how gates are connected. In other words, if $\mathsf{gate}_i = (g, w_a, w_b, w_c)$, then $\widehat{\mathsf{gate}}_i = (\perp, w_a, w_b, w_c)$.

Proof of Theorem 6.7. First, let us define a simulator Sim as follows.

The simulator $\mathsf{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i\in[n]})$**:**

1. Parse $\{L_{i,x_i}\}_{i\in[n]} := \{\mathsf{ske.sk}_i^{x_i}\}_{i\in[n]}$.
2. For $i\in[n]$, generate $\mathsf{ske.sk}_i^{x_i\oplus 1}\leftarrow\mathsf{SKE.KeyGen}(1^\lambda)$.
3. For $i\in\{n+1, n+2, \cdots, p\}$ and $\sigma\in\{0,1\}$, generate $\mathsf{ske.sk}_i^\sigma\leftarrow\mathsf{SKE.KeyGen}(1^\lambda)$.
4. For each $i\in[q]$, compute $(\mathsf{vk}_i, \widetilde{g}_i)\leftarrow\mathsf{Sim.GateGrbl}(\widehat{\mathsf{gate}}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{Sim.GateGrbl}$ is described in Fig 5 and $\widehat{\mathsf{gate}}_i = (\perp, w_a, w_b, w_c)$.
5. For each $i\in[m]$, generate $\widetilde{d}_i := \left[\left(\mathsf{ske.sk}^0_{\mathsf{out}_i}, C(x)_i\right), \left(\mathsf{ske.sk}^1_{\mathsf{out}_i}, C(x)_i\oplus 1\right)\right]$.
6. Output $\widetilde{C} := (\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ and $\mathsf{vk} := \{\mathsf{vk}_i\}_{i\in[q]}$.

For each $j\in[q]$, we define a QPT algorithm (a simulator) $\mathsf{InputDepSim}_j$ as follows.

The simulator $\mathsf{InputDepSim}_j(1^\lambda, C, x, \{L_{i,x_i}\}_{i\in[n]})$**:**

1. Parse $\{L_{i,x_i}\}_{i\in[n]} = \{\mathsf{ske.sk}_i^{x_i}\}_{i\in[n]}$.
2. For $i\in[n]$, generate $\mathsf{ske.sk}_i^{x_i\oplus 1}\leftarrow\mathsf{SKE.KeyGen}(1^\lambda)$.
3. For $i\in\{n+1, n+2, \cdots, p\}$ and $\sigma\in\{0,1\}$, generate $\mathsf{ske.sk}_i^\sigma\leftarrow\mathsf{SKE.KeyGen}(1^\lambda)$.

Simulation Gate Garbling Circuit $\mathsf{Sim.GateGrbl}$

Input: $(\widehat{\mathsf{gate}}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$.

Output: $\widetilde{g_i}$ and vk_i.

1. For each $\sigma_a, \sigma_b \in \{0,1\}$, sample $p_{a,b}^{\sigma_a,\sigma_b} \leftarrow \mathcal{K}$.
2. Sample $\gamma_i \leftarrow \mathsf{S}_4$.
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b})$ and $(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^0)$.
4. Output $\widetilde{g_i} := \{\mathsf{ske.CT}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in permutated order of γ_i and $\mathsf{vk}_i := \{\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.vk}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in permutated order of γ_i.

Figure 5: The description of $\mathsf{Sim.GateGrbl}$

4. For $i \in [j]$, compute $(\mathsf{vk}_i, \widetilde{g_i}) \leftarrow \mathsf{InputDep.GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{InputDep.GateGrbl}$ is described in Fig. 6 and $\mathsf{gate}_i = (g, w_a, w_b, w_c)$
5. For each $i \in \{j+1, j+2, \cdots, q\}$, compute $(\mathsf{vk}_i, \widetilde{g_i}) \leftarrow \mathsf{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{GateGrbl}$ is described in Fig 1 and $\mathsf{gate}_i = (g, w_a, w_b, w_c)$.
6. For each $i \in [m]$, generate $\widetilde{d_i} := \left[\left(\mathsf{ske.sk}_{\mathsf{out}_i}^0, 0\right), \left(\mathsf{ske.sk}_{\mathsf{out}_i}^1, 1\right)\right]$.
7. Output $\widetilde{C} := (\{\widetilde{g_i}\}_{i\in[q]}, \{\widetilde{d_i}\}_{i\in[m]})$ and $\mathsf{vk} := \{\mathsf{vk}_i\}_{i\in[q]}$.

Input Dependent Gate Garbling Circuit $\mathsf{InputDep.GateGrbl}$

Input: $\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}}$.

Output: $\widetilde{g_i}$ and vk_i.

1. For each $\sigma_a, \sigma_b \in \{0,1\}$, sample $p_c^{\sigma_a,\sigma_b} \leftarrow \mathcal{K}$.
2. Sample $\gamma_i \leftarrow \mathsf{S}_4$.
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b})$ and $(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^{v(c)})$. Here, $v(c)$ is the correct value of the bit going over the wire c during the computation of $C(x)$.
4. Output $\widetilde{g_i} := \{\mathsf{ske.CT}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in permutated order of γ_i and $\mathsf{vk}_i := \{\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.vk}_b^{\sigma_a,\sigma_b}\}_{\sigma_a,\sigma_b\in\{0,1\}}$ in permutated order of γ_i.

Figure 6: The description of $\mathsf{InputDep.GateGrbl}$

For each $j \in \{0, 1, \cdots, q\}$, let us define a sequence of hybrid games $\{\mathsf{Hyb}_j\}_{j\in\{0,1,\cdots,q\}}$ against any adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$, where $\mathcal{A}_1$ is any QPT adversary and $\mathcal{A}_2$ is any unbounded adversary. Note that

$$\mathsf{InputDepSim}_0(1^\lambda, C, x, \{L_{i,x_i}\}_{i\in[n]}) = \mathsf{Grbl}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}).$$

The hybrid game Hyb_j**:**

1. $\mathcal{A}_1$ sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ to the challenger.
2. The challenger computes $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Samp}(1^\lambda)$.
3. The challenger computes $(\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{GC.InputDepSim}_j(1^\lambda, C, x, \{L_{i,x_i}\}_{i\in[n]})$, and sends $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}_1$.
4. At some point, $\mathcal{A}_1$ sends cert to the challenger and the internal state to $\mathcal{A}_2$.
5. The challenger computes $\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \rightarrow \top/\bot$. If the output is $\bot$, then the challenger outputs $\bot$ and sends $\bot$ to $\mathcal{A}_2$. Else, the challenger outputs $\top$ and sends $\top$ to $\mathcal{A}_2$.

6. $\mathcal{A}_2$ outputs $b' \in \{0,1\}$.
7. If the challenger outputs $\top$, then the output of the experiment is b'. Otherwise, the output of the experiment is $\bot$.

Note that Hyb_0 is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}select}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}(1^\lambda, 0)$ by definition. Therefore, Theorem 6.7 easily follows from the following Propositions B.1 and B.2 (whose proofs are given later). □

Proposition B.1. *If Σ_{cesk} satisfies the certified everlasting IND-CPA security, it holds that* $\left|\Pr\left[\mathsf{Hyb}_{j-1} = 1\right] - \Pr\left[\mathsf{Hyb}_j = 1\right]\right| \leq \mathsf{negl}(\lambda)$ *for all $j \in [q]$.*

Proposition B.2. $\left|\Pr\left[\mathsf{Hyb}_q = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}select}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$

Proof of Proposition B.1. For the proof, we use Lemma B.3 whose statement and proof are given in Appendix B. We construct an adversary $\mathcal{B}$ that breaks the security experiment of $\mathsf{Exp}^{\mathsf{parallel\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}(\lambda, b)$, which is described in Lemma B.3, assuming that $\left|\Pr\left[\mathsf{Hyb}_{j-1} = 1\right] - \Pr\left[\mathsf{Hyb}_j = 1\right]\right|$ is non-negligible. This contradicts the certified everlasting IND-CPA security of Σ_{cesk} from Lemma B.3. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $C \in \mathcal{C}_n$ and $x \in \{0,1\}^n$ from $\mathcal{A}_1$. Let $\mathsf{gate_j} = (g, w_\alpha, w_\beta, w_\gamma)$.
2. The challenger of $\mathsf{Exp}^{\mathsf{parallel\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}(\lambda, b)$ generates $\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and $\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$[3].
3. For each $i \in [p] \setminus \{\alpha, \beta\}$ and $\sigma \in \{0,1\}$, $\mathcal{B}$ generates $\mathsf{ske.sk}_i^\sigma \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ generates $\mathsf{ske.sk}_\alpha^{v(\alpha)} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and $\mathsf{ske.sk}_\beta^{v(\beta)} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ sets $\{L_{i,x_i}\}_{i\in[n]} \coloneqq \{\mathsf{ske.sk}_i^{x_i}\}_{i\in[n]}$.
4. For each $i \in [j-1]$, $\mathcal{B}$ computes $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{InputDep.GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{InputDep.GateGrbl}$ is described in Fig 6 and $\mathsf{gate}_i = (g, w_a, w_b, w_c)$. $\mathcal{B}$ calls the encryption query of $\mathsf{Exp}^{\mathsf{parallel\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}(\lambda, b)$ if it needs to use $\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}$ or $\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}$ to run $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{InputDep.GateGrbl}(\mathsf{gate}_i,$ $\{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$.
5. $\mathcal{B}$ samples $p_\gamma^{v(\alpha),v(\beta)} \leftarrow \mathcal{K}$. $\mathcal{B}$ computes
$$(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \mathsf{ske.CT}_\alpha^{v(\alpha),v(\beta)}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)}, p_\gamma^{v(\alpha),v(\beta)}),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)}, \mathsf{ske.CT}_\beta^{v(\alpha),v(\beta)}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_\beta^{v(\beta)}, p_\gamma^{v(\alpha),v(\beta)} \oplus \mathsf{ske.sk}_\gamma^{v(\gamma)}).$$
6. $\mathcal{B}$ sets
$$(x_0, y_0, z_0) \coloneqq (\mathsf{ske.sk}_\gamma^{g(v(\alpha),v(\beta)\oplus 1)}, \mathsf{ske.sk}_\gamma^{g(v(\alpha)\oplus 1,v(\beta))}, \mathsf{ske.sk}_\gamma^{g(v(\alpha)\oplus 1,v(\beta)\oplus 1)}),$$
$$(x_1, y_1, z_1) \coloneqq (\mathsf{ske.sk}_\gamma^{v(\gamma)}, \mathsf{ske.sk}_\gamma^{v(\gamma)}, \mathsf{ske.sk}_\gamma^{v(\gamma)}),$$
and sends $(\mathsf{ske.sk}_\alpha^{v(\alpha)}, \mathsf{ske.sk}_\beta^{v(\beta)}, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma\in\{0,1\}})$ to the challenger of $\mathsf{Exp}^{\mathsf{parallel\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}(\lambda, b)$.
7. The challenger samples $(x, y, z) \leftarrow \mathcal{K}^3$ and $(\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}, \mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. The challenger computes
$$(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.CT}_\alpha^{v(\alpha),v(\beta)\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)}, x),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.CT}_\beta^{v(\alpha),v(\beta)\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}, x \oplus x_b),$$
$$(\mathsf{ske.vk}_\alpha^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.CT}_\alpha^{v(\alpha)\oplus 1,v(\beta)}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}, y),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.CT}_\beta^{v(\alpha)\oplus 1,v(\beta)}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)}, y \oplus y_b),$$
$$(\mathsf{ske.vk}_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske.CT}_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}, z),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske.CT}_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1}) \leftarrow \mathsf{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}, z \oplus z_b),$$

[3] Recall that $v(\alpha)$ is the correct value of the bit going over the wire α during the computation of $C(x)$.

and sends

$$(\mathsf{ske.CT}_\alpha^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.CT}_\beta^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.CT}_\alpha^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.CT}_\beta^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.CT}_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske.CT}_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1})$$

to $\mathcal{B}$.

8. $\mathcal{B}$ samples $\gamma_j \leftarrow \mathsf{S}_4$. $\mathcal{B}$ sets $\widetilde{g_j} := \{\mathsf{ske.CT}_\alpha^{\sigma_\alpha,\sigma_\beta}, \mathsf{ske.CT}_\beta^{\sigma_\alpha,\sigma_\beta}\}_{\sigma_\alpha,\sigma_\beta\in\{0,1\}}$ in the permutated order of γ_j.
9. For each $i \in \{j+1, j+2, \cdots, q\}$, $\mathcal{B}$ computes $(\mathsf{vk}_i, \widetilde{g_i}) \leftarrow \mathsf{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathcal{B}$ calls the encryption query of $\mathsf{Exp}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, b)$ if $\mathcal{B}$ needs to use $\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}$ or $\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}$ to run $(\mathsf{vk}_i, \widetilde{g_i}) \leftarrow \mathsf{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$.
10. $\mathcal{B}$ computes $\widetilde{d_i} := [(\mathsf{ske.sk}_{\mathsf{out}_i}^0, 0), (\mathsf{ske.sk}_{\mathsf{out}_i}^1, 1)]$ for $i \in [m]$, sets $\widetilde{C} := (\{\widetilde{g_i}\}_{i\in[q]}, \{\widetilde{d_i}\}_{i\in[m]})$, and sends $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ to $\mathcal{A}_1$.
11. At some point, $\mathcal{A}_1$ sends $\mathsf{cert} := \{\mathsf{cert}_i\}_{i\in[q]}$ to $\mathcal{B}$ and the internal state to $\mathcal{A}_2$, respectively.
12. $\mathcal{B}$ re-sorts $\mathsf{cert}_j = \{\mathsf{ske.cert}_\alpha^{\sigma_\alpha,\sigma_\beta}, \mathsf{ske.cert}_\beta^{\sigma_\alpha,\sigma_\beta}\}_{\sigma_\alpha,\sigma_\beta\in\{0,1\}}$ according to γ_j. $\mathcal{B}$ sends

$$(\mathsf{ske.cert}_\alpha^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.cert}_\beta^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske.cert}_\alpha^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.cert}_\beta^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske.cert}_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske.cert}_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1})$$

to the challenger of $\mathsf{Exp}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, b)$ and receives $\perp$ or $(\mathsf{ske.sk}_\alpha'^{v(\alpha)\oplus 1}, \mathsf{ske.sk}_\beta'^{v(\beta)\oplus 1})$ from the challenger. $\mathcal{B}$ computes $\mathsf{SKE.Vrfy}(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \mathsf{ske.cert}_\alpha^{v(\alpha),v(\beta)})$ and $\mathsf{SKE.Vrfy}(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)}, \mathsf{ske.cert}_\beta^{v(\alpha),v(\beta)})$. $\mathcal{B}$ computes $\mathsf{GateVrfy}(\mathsf{vk}_i, \mathsf{cert}_i)$ for each $i \in \{1, 2, \cdots, j-1, j+1, j+2, \cdots, q\}$, where GateVrfy is described in Fig4. If $\mathcal{B}$ receives $(\mathsf{ske.sk}_\alpha'^{v(\alpha)\oplus 1}, \mathsf{ske.sk}_\beta'^{v(\beta)\oplus 1})$ from the challenger, $\top \leftarrow \mathsf{SKE.Vrfy}(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \mathsf{ske.cert}_\alpha^{v(\alpha),v(\beta)})$, $\top \leftarrow \mathsf{SKE.Vrfy}(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)}, \mathsf{ske.cert}_\beta^{v(\alpha),v(\beta)})$, and $\top \leftarrow \mathsf{GateVrfy}(\mathsf{cert}_i, \mathsf{vk}_i)$ for all $i \in \{1, 2, \cdots, j-1, j+1, j+2, \cdots, q\}$, then $\mathcal{B}$ sends $\top$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}_2$, and aborts.
13. $\mathcal{B}$ outputs the output of $\mathcal{A}_2$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} | b = 0] = \Pr\left[\mathsf{Hyb}_{j-1} = 1\right]$ and $\Pr[1 \leftarrow \mathcal{B} | b = 1] = \Pr\left[\mathsf{Hyb}_j = 1\right]$. Therefore, if for an adversary $\mathcal{A}$, $\left|\Pr\left[\mathsf{Hyb}_{j-1} = 1\right] - \Pr\left[\mathsf{Hyb}_j = 1\right]\right|$ is non-negligible, then $\left|\Pr\left[\mathsf{Exp}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{Exp}_{\Sigma_{\mathsf{cesk}},\mathcal{B}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, 1) = 1\right]\right|$ is non-negligible. From Lemma B.3, it contradicts the certified everlasting IND-CPA security of Σ_{cesk} , which completes the proof. ☐

Proof of Proposition B.2. To show Proposition B.2, it is sufficient to prove that the probability distribution of $\widetilde{C}$ in $\mathsf{Exp}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}select}}(1^\lambda, 1)$ is statistically identical to that of $\widetilde{C}$ in Hyb_q.

First, let us remind the difference between Hyb_q and $\mathsf{Exp}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}select}}(1^\lambda, 1)$. In both experiments , $\widetilde{C}$ consists of $\{\widetilde{g_i}\}_{i\in[q]}$ and $\{\widetilde{d_i}\}_{i\in[m]}$. On the other hand the contents of $\{\widetilde{g_i}\}_{i\in[q]}$ and $\{\widetilde{d_i}\}_{i\in[m]}$ are different in each experiments. In Hyb_q, $\widetilde{g_i}$ consists of $(\mathsf{ske.CT}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b})$ where

$$(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b}),$$
$$(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^{v(c)}),$$

and $\widetilde{d_i}$ is

$$[(\mathsf{ske.sk}_{\mathsf{out_i}}^0, 0), (\mathsf{ske.sk}_{\mathsf{out_i}}^1, 1)].$$

In $\mathsf{Exp}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}select}}(1^\lambda, 1)$, $\widetilde{g_i}$ consists of $(\mathsf{ske.CT}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b})$ where

$$(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b}),$$
$$(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske.CT}_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^{0}),$$

and $\widetilde{d}_i$ is

$$[(\mathsf{ske.sk}^0_{\mathsf{out}_i}, C(x)_i), (\mathsf{ske.sk}^1_{\mathsf{out}_i}, C(x)_i \oplus 1)].$$

The resulting distribution of $(\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ in Hyb_q is statistically identical to the resulting distribution of $(\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ in $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}select}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}(1^\lambda, 1)$. This is because, at any level that is not output, the keys $\mathsf{ske.sk}^0_c, \mathsf{ske.sk}^1_c$ are used completely identically in the subsequent level so there is no difference between always encrypting $\mathsf{ske.sk}^{v(c)}_c$ and $\mathsf{ske.sk}^0_c$. At the output level, there is no difference between encrypting $\mathsf{ske.sk}^{v(c)}_c$ and giving the real mapping $\mathsf{ske.sk}^{v(c)}_c \rightarrow v(c)$ or encrypting $\mathsf{ske.sk}^0_c$ and giving the programming mapping $\mathsf{ske.sk}^0_c \rightarrow C(x)_i$, which completes the proof. $\square$

We use the following lemma for the proof of Proposition B.1. The proof is shown with the standard hybrid argument. It is also easy to see that a similar lemma holds for IND-CPA security.

Lemma B.3. *Let* $\Sigma := (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Del}, \mathsf{Vrfy})$ *be a certified everlasting SKE scheme. Let us consider the following security experiment* $\mathsf{Exp}^{\mathsf{parallel\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against* $\mathcal{A}$ *consisting of any QPT adversary* $\mathcal{A}_1$ *and any unbounded adversary* $\mathcal{A}_2$.

1. *The challenger generates* $(\mathsf{sk}'^0, \mathsf{sk}'^1) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
2. $\mathcal{A}_1$ *can call encryption queries. More formally, it can do the followings:* $\mathcal{A}_1$ *chooses* $\beta \in \{0,1\}$, $\mathsf{sk} \in \mathcal{SK}$ *and* $m \in \mathcal{M}$. $\mathcal{A}_1$ *sends* (β, sk, m) *to the challenger.*
   - *If* $\beta = 0$, *the challenger generates* $m^* \leftarrow \mathcal{M}$, *computes* $(\mathsf{vk}^0_m, \mathsf{CT}^0_m) \leftarrow \mathsf{Enc}(\mathsf{sk}'^0, m^*)$ *and* $(\mathsf{vk}^1_m, \mathsf{CT}^1_m) \leftarrow \mathsf{Enc}(\mathsf{sk}, m \oplus m^*)$, *and sends* $\{\mathsf{vk}^\sigma_m, \mathsf{CT}^\sigma_m\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$.
   - *If* $\beta = 1$, *the challenger generates* $m^* \leftarrow \mathcal{M}$, *computes* $(\mathsf{vk}^1_m, \mathsf{CT}^1_m) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, m \oplus m^*)$ *and* $(\mathsf{vk}^0_m, \mathsf{CT}^0_m) \leftarrow \mathsf{Enc}(\mathsf{sk}, m^*)$, *and sends* $\{\mathsf{vk}^\sigma_m, \mathsf{CT}^\sigma_m\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$.

   $\mathcal{A}_1$ *can repeat this process polynomially many times.*
3. $\mathcal{A}_1$ *generates* $(\mathsf{sk}^0, \mathsf{sk}^1) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and chooses two triples of messages* $(x_0, y_0, z_0) \in \mathcal{M}^3$ *and* $(x_1, y_1, z_1) \in \mathcal{M}^3$, *and sends* $(\mathsf{sk}^0, \mathsf{sk}^1, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma\in\{0,1\}})$ *to the challenger.*
4. *The challenger generates* $(x, y, z) \leftarrow \mathcal{M}^3$. *The challenger computes*

$$\begin{aligned}
&(\mathsf{vk}^0_x, \mathsf{CT}^0_x) \leftarrow \mathsf{Enc}(\mathsf{sk}^0, x), \quad (\mathsf{vk}^1_x, \mathsf{CT}^1_x) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, x \oplus x_b)\\
&(\mathsf{vk}^0_y, \mathsf{CT}^0_y) \leftarrow \mathsf{Enc}(\mathsf{sk}'^0, y), \quad (\mathsf{vk}^1_y, \mathsf{CT}^1_y) \leftarrow \mathsf{Enc}(\mathsf{sk}^1, y \oplus y_b)\\
&(\mathsf{vk}^0_z, \mathsf{CT}^0_z) \leftarrow \mathsf{Enc}(\mathsf{sk}'^0, z), \quad (\mathsf{vk}^1_z, \mathsf{CT}^1_z) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, z \oplus z_b)
\end{aligned}$$

   and sends $\{\mathsf{CT}^\sigma_x, \mathsf{CT}^\sigma_y, \mathsf{CT}^\sigma_z\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$.
5. $\mathcal{A}_1$ *can call encryption queries. More formally, it can do the followings:* $\mathcal{A}_1$ *chooses* $\beta \in \{0,1\}$, $\mathsf{sk} \in \mathcal{SK}$ *and* $m \in \mathcal{M}$. $\mathcal{A}_1$ *sends* (β, sk, m) *to the challenger.*
   - *If* $\beta = 0$, *the challenger generates* $m^* \leftarrow \mathcal{M}$, *computes* $(\mathsf{vk}^0_m, \mathsf{CT}^0_m) \leftarrow \mathsf{Enc}(\mathsf{sk}'^0, m^*)$ *and* $(\mathsf{vk}^1_m, \mathsf{CT}^1_m) \leftarrow \mathsf{Enc}(\mathsf{sk}, m \oplus m^*)$, *and sends* $\{\mathsf{vk}^\sigma_m, \mathsf{CT}^\sigma_m\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$.
   - *If* $\beta = 1$, *the challenger generates* $m^* \leftarrow \mathcal{M}$, *computes* $(\mathsf{vk}^1_m, \mathsf{CT}^1_m) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, m \oplus m^*)$ *and* $(\mathsf{vk}^0_m, \mathsf{CT}^0_m) \leftarrow \mathsf{Enc}(\mathsf{sk}, m^*)$, *and sends* $\{\mathsf{vk}^\sigma_m, \mathsf{CT}^\sigma_m\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$.

   $\mathcal{A}_1$ *can repeat this process polynomially many times.*
6. $\mathcal{A}_1$ *sends* $\{\mathsf{cert}^\sigma_x, \mathsf{cert}^\sigma_y, \mathsf{cert}^\sigma_z\}_{\sigma\in\{0,1\}}$ *to the challenger, and sends the internal state to* $\mathcal{A}_2$.

7. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}_x^\sigma, \mathsf{cert}_x^\sigma)$, $\mathsf{Vrfy}(\mathsf{vk}_y^\sigma, \mathsf{cert}_y^\sigma)$ *and* $\mathsf{Vrfy}(\mathsf{vk}_z^\sigma, \mathsf{cert}_z^\sigma)$ *for each* $\sigma \in \{0,1\}$. *If all results are* $\top$, *then the challenger outputs* $\top$, *and sends* $\{\mathsf{sk}'^\sigma\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_2$. *Otherwise, the challenger outputs* $\bot$, *and sends* $\bot$ *to* $\mathcal{A}_2$.

8. $\mathcal{A}_2$ *outputs* $b' \in \{0,1\}$.

9. *If the challenger outputs* $\top$, *then the output of the experiment is* b'. *Otherwise, the output of the experiment is* $\bot$.

If the Σ *satisfies the certified everlasting IND-CPA security,*

$$\mathsf{Adv}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda) := \left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,0)=1\right] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,1)=1\right]\right| \leq \mathsf{negl}(\lambda)$$

for any QPT adversary $\mathcal{A}_1$ *and any unbounded adversary* $\mathcal{A}_2$.

Proof of Lemma B.3. We define the following hybrid experiment.

Hyb_1**:** This is identical to $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,0)$ except that the challenger encrypts (x_0, y_0, z_1) instead of encrypting (x_0, y_0, z_0).

Hyb_2**:** This is identical to Hyb_1 except that the challenger encrypts (x_0, y_1, z_1) instead of encrypting (x_0, y_0, z_1).

Lemma B.3 easily follows from the following Propositions B.4 to B.6 (whose proof is given later.). $\square$

Proposition B.4. *If* Σ *is certified everlasting IND-CPA secure, it holds that* $\left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,0)=1\right] - \Pr[\mathsf{Hyb}_1=1]\right| \leq \mathsf{negl}(\lambda)$.

Proposition B.5. *If* Σ *is certified everlasting IND-CPA secure, it holds that* $|\Pr[\mathsf{Hyb}_1=1] - \Pr[\mathsf{Hyb}_2=1]| \leq \mathsf{negl}(\lambda)$.

Proposition B.6. *If* Σ *is certified everlasting IND-CPA secure, it holds that* $\left|\Pr[\mathsf{Hyb}_2=1] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,1)=1\right]\right| \leq \mathsf{negl}(\lambda)$.

Proof of Proposition B.4. We assume that $\left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda,0)=1\right] - \Pr[\mathsf{Hyb}_1(1)=1]\right|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment of $\mathsf{Exp}_{\Sigma,\mathcal{B}}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}(\lambda,b)$. This contradicts the certified everlasting IND-CPA security of Σ. Let us describe how $\mathcal{B}$ works.

1. The challenger of $\mathsf{Exp}_{\Sigma,\mathcal{B}}^{\mathsf{cert\text{-}ever\text{-}ind\text{-}cpa}}(\lambda,b)$ generates $\mathsf{sk}'^0 \leftarrow \mathsf{KeyGen}(1^\lambda)$, and $\mathcal{B}$ generates $\mathsf{sk}'^1 \leftarrow \mathsf{KeyGen}(1^\lambda)$.

2. $\mathcal{A}_1$ chooses $\beta \in \{0,1\}$, $\mathsf{sk} \in \mathcal{K}$ and $m \in \mathcal{M}$. $\mathcal{A}_1$ sends (β, sk, m) to $\mathcal{B}$.

   - If $\beta = 0$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, sends m^* to the challenger, receives $(\mathsf{vk}_m^0, \mathsf{CT}_m^0)$ from the challenger, computes $(\mathsf{vk}_m^1, \mathsf{CT}_m^1) \leftarrow \mathsf{Enc}(\mathsf{sk}, m \oplus m^*)$, and sends $\{\mathsf{vk}_m^\sigma, \mathsf{CT}_m^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.
   - If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(\mathsf{vk}_m^1, \mathsf{CT}_m^1) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, m \oplus m^*)$ and $(\mathsf{vk}_m^0, \mathsf{CT}_m^0) \leftarrow \mathsf{Enc}(\mathsf{sk}, m^*)$ and sends $\{\mathsf{vk}_m^\sigma, \mathsf{CT}_m^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.

   $\mathcal{B}$ repeats this process when (β, sk, m) is sent from $\mathcal{A}_1$.

3. $\mathcal{B}$ receives $(\mathsf{sk}^0, \mathsf{sk}^1, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma\in\{0,1\}})$ from $\mathcal{A}_1$.

4. $\mathcal{B}$ generates $(x, y, z) \leftarrow \mathcal{M}^3$. $\mathcal{B}$ computes

$$(\mathsf{vk}_x^0, \mathsf{CT}_x^0) \leftarrow \mathsf{Enc}(\mathsf{sk}^0, x), (\mathsf{vk}_x^1, \mathsf{CT}_x^1) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, x \oplus x_0),$$
$$(\mathsf{vk}_y^1, \mathsf{CT}_y^1) \leftarrow \mathsf{Enc}(\mathsf{sk}^1, y \oplus y_0),$$
$$(\mathsf{vk}_z^1, \mathsf{CT}_z^1) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, z \oplus z_0).$$

5. $\mathcal{B}$ sets $m_0 := z$ and $m_1 := z \oplus z_0 \oplus z_1$. $\mathcal{B}$ sends (m_0, m_1) to the challenger.

6. The challenger computes $(\mathsf{vk}_z^0, \mathsf{CT}_z^0) \leftarrow \mathsf{Enc}(\mathsf{sk}'^0, m_b)$, and sends CT_z^0 to $\mathcal{B}$.
7. $\mathcal{B}$ sends an encryption query y to the challenger, and receives $(\mathsf{vk}_y^0, \mathsf{CT}_y^0)$.
8. $\mathcal{B}$ sends $\{\mathsf{CT}_x^\sigma, \mathsf{CT}_y^\sigma, \mathsf{CT}_z^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.
9. $\mathcal{A}_1$ chooses $\beta \in \{0,1\}$, $\mathsf{sk} \in \mathcal{K}$ and $m \in \mathcal{M}$. $\mathcal{A}_1$ sends (β, sk, m) to $\mathcal{B}$.
   - If $\beta = 0$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, sends m^* to the challenger, receives $(\mathsf{vk}_m^0, \mathsf{CT}_m^0)$ from the challenger, computes $(\mathsf{vk}_m^1, \mathsf{CT}_m^1) \leftarrow \mathsf{Enc}(\mathsf{sk}, m \oplus m^*)$, and sends $\{\mathsf{vk}_m^\sigma, \mathsf{CT}_m^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.
   - If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(\mathsf{vk}_m^1, \mathsf{CT}_m^1) \leftarrow \mathsf{Enc}(\mathsf{sk}'^1, m \oplus m^*)$ and $(\mathsf{vk}_m^0, \mathsf{CT}_m^0) \leftarrow \mathsf{Enc}(\mathsf{sk}, m^*)$ and sends $\{\mathsf{vk}_m^\sigma, \mathsf{CT}_m^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.

   $\mathcal{B}$ repeats this process when (β, sk, m) is sent from $\mathcal{A}_1$.
10. $\mathcal{A}_1$ sends $\{\mathsf{cert}_x^\sigma, \mathsf{cert}_y^\sigma, \mathsf{cert}_z^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{B}$, and sends the internal state to $\mathcal{A}_2$.
11. $\mathcal{B}$ sends cert_z^0 to the challenger, and receives sk'^0 or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$, it outputs $\perp$ and aborts.
12. $\mathcal{B}$ sends $\{\mathsf{sk}'^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_2$.
13. $\mathcal{A}_2$ outputs b'.
14. $\mathcal{B}$ computes $\mathsf{Vrfy}(\mathsf{vk}_x^\sigma, \mathsf{cert}_x^\sigma)$ and $\mathsf{Vrfy}(\mathsf{vk}_y^\sigma, \mathsf{cert}_y^\sigma)$ for each $\sigma \in \{0,1\}$, and $\mathsf{Vrfy}(\mathsf{vk}_z^1, \mathsf{cert}_z^1)$. If all results are $\top$, $\mathcal{B}$ outputs b'. Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B}|b=0] = \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, 0) = 1\right]$. Since z is uniformly distributed, $(z, z \oplus z_1)$ and $(z \oplus z_0 \oplus z_1, z \oplus z_0)$ are identically distributed. Therefore, it holds that $\Pr[1 \leftarrow \mathcal{B}|b=1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $\left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, 0) = 1\right] - \Pr[\mathsf{Hyb}_1 = 1]\right|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B}|b=0] - \Pr[1 \leftarrow \mathcal{B}|b=1]|$ is non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cesk}. □

Proof of Proposition B.5. The proof is very similar to that of Proposition B.4. Therefore we skip the proof. □

Proof of Proposition B.6. The proof is very similar to that of Proposition B.4. Therefore, we skip the proof. □

C Proof of Theorem 7.10

Proof of Theorem 7.10. Let us describe how the simulator Sim works.

$\mathsf{Sim}(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$:

1. Parse $\mathsf{MPK} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ and $\mathcal{V} = \{f(m), f, (f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})\}$ *or* $\emptyset$.
2. If $\mathcal{V} = \emptyset$, generate $f \leftarrow \{0,1\}^s$.
3. Generate $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$ and $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$.
4. Compute $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|f|}, U(f,m), \{L_{i,f[i]}\}_{i\in[s]})$.
5. Compute $(\mathsf{pke.vk}_{i,f[i]}, \mathsf{pke.CT}_{i,f[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]}, L_{i,f[i]})$ and $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke.CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L^*_{i,f[i]\oplus 1})$ for every $i \in [s]$.
6. Output $\mathsf{vk} := (\mathsf{gc.vk}, \{\mathsf{pke.vk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ and $\mathsf{CT} := (\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.

Let us define the sequence of hybrids as follows.

Hyb_0: This is identical to $\mathsf{Exp}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}(\lambda, 0)$.

1. The challenger generates $(\mathsf{pke.pk}_{i,\alpha}, \mathsf{pke.sk}_{i,\alpha}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
2. $\mathcal{A}_1$ is allowed to call a key query at most one time. If a key query is called, the challenger receives an function f from $\mathcal{A}_1$, and sends $(f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{A}_1$.
3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to the challenger.
4. The challenger computes $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$, $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$, and $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.CT}_{i,\alpha}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,\alpha}, L_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $(\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ sends $(\mathsf{gc.cert}, \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
6. If $\top \leftarrow \mathsf{GC.Vrfy}(\mathsf{gc.vk}, \mathsf{gc.cert})$, and $\top \leftarrow \mathsf{PKE.Vrfy}(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke.cert}_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, the challenger outputs $\top$, and sends $\{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b'. If the challenger outputs $\top$, the output of the experiment is b'. Otherwise, the output of the experiment is $\perp$.

Hyb_1**:** This is identical to Hyb_0 except for the following four points. First, the challenger generates $f \in \{0,1\}^s$ if a key query is not called in step 2. Second, the challenger randomly generates $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$ and $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$ in step 2 regardless of whether a key query is called or not. Third, the challenger does not compute $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$ in step 4. Fourth, the challenger computes $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke.CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L^*_{i,f[i]\oplus 1})$ for every $i \in [s]$ instead of computing $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke.CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L_{i,f[i]\oplus 1})$ for every $i \in [s]$.

Hyb_2**:** This is identical to Hyb_1 except for the following point. The challenger computes $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|f|}, U(f,m), \{L_{i,f[i]}\}_{i\in[s]})$ instead of computing $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.

From the definition of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, b)$ and Sim, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0) = 1\right]$ and $\Pr[\mathsf{Hyb}_2 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 1) = 1\right]$. Therefore, Theorem 7.10 easily follows from the following Propositions C.1 and C.2. (whose proof is given later.) $\square$

Proposition C.1. *If Σ_{cepk} satisfies the certified everlasting IND-CPA security,*

$$|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda).$$

Proposition C.2. *If Σ_{cegc} satisfies the certified everlasting selective security,*

$$|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| \leq \mathsf{negl}(\lambda).$$

Proof of Proposition C.1. For the proof, we use Lemma A.3 whose statement and proof is given in Appendix A. We assume that $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$ defined in Lemma A.3. This contradicts the certified everlasting IND-CPA of Σ_{cepk} from Lemma A.3. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ from the challenger of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$, and sends $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.
2. $\mathcal{A}_1$ is allowed to call a key query at most one time. If a key query is called, $\mathcal{B}$ receives an function f from $\mathcal{A}_1$, generates $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$ and $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$. If a key query is not called, $\mathcal{B}$ generates $f \leftarrow \{0,1\}^s$, $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$ and $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$.
3. $\mathcal{B}$ sends $(f, L_{1,f[1]\oplus 1}, L_{2,f[2]\oplus 1}, \cdots, L_{s,f[s]\oplus 1}, L^*_{1,f[1]\oplus 1}, L^*_{2,f[2]\oplus 1}, \cdots, L^*_{s,f[s]\oplus 1})$ to the challenger of $\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma_{\mathsf{cepk}},\mathcal{B}}(\lambda, b)$.

4. $\mathcal{B}$ receives $(\{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]}, \{\mathsf{pke.CT}_{i,f[i]\oplus 1}\}_{i\in[s]})$ from the challenger. If a key query is called, $\mathcal{B}$ sends $(f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{A}_1$.

5. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to $\mathcal{B}$.

6. $\mathcal{B}$ computes $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ and $(\mathsf{pke.vk}_{i,f[i]}, \mathsf{pke.CT}_{i,f[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]}, L_{i,f[i]})$ for every $i \in [s]$, and sends $(\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to $\mathcal{A}_1$.

7. $\mathcal{A}_1$ sends $(\mathsf{gc.cert}, \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to $\mathcal{B}$, and sends its internal state to $\mathcal{A}_2$.

8. $\mathcal{B}$ sends $\{\mathsf{pke.cert}_{i,f[i]\oplus 1}\}_{i\in[s]}$ to the challenger, and receives $\{\mathsf{pke.sk}_{i,f[i]\oplus 1}\}_{i\in[s]}$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.

9. $\mathcal{B}$ sends $\{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_2$.

10. $\mathcal{A}_2$ outputs b'.

11. $\mathcal{B}$ computes $\mathsf{GC.Vrfy}$ for $\mathsf{gc.cert}$ and $\mathsf{PKE.Vrfy}$ for all $\{\mathsf{pke.cert}_{i,f[i]}\}_{i\in[s]}$, and outputs b' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B}|b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B}|b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B}|b = 0] - \Pr[1 \leftarrow \mathcal{B}|b = 1]|$ is non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma A.3. ☐

Proof of Proposition C.2. We assume that $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the certified everlasting selective security of Σ_{cegc}. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ generates $(\mathsf{pke.pk}_{i,\alpha}, \mathsf{pke.sk}_{i,\alpha}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_1$.

2. $\mathcal{A}_1$ is allowed to call a key query at most one time. If a key query is called, $\mathcal{B}$ receives an function f from $\mathcal{A}_1$, generates $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$, and sends $(f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{A}_1$. If a key query is not called, $\mathcal{B}$ generates $f \leftarrow \{0,1\}^s$ and $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$.

3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to $\mathcal{B}$.

4. $\mathcal{B}$ sends a circuit $U(\cdot, m)$ and an input $f \in \{0,1\}^s$ to the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}selct}}_{\mathcal{B},\Sigma_{\mathsf{cegc}}}(1^\lambda, b)$.

5. The challenger computes $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Samp}(1^\lambda)$ and does the following:

   - If $b = 0$, the challenger computes $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Grbl}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$, and sends $(\widetilde{U}, \{L_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{B}$.
   - If $b = 1$, the challenger computes $(\widetilde{U}, \mathsf{gc.vk}) \leftarrow \mathsf{GC.Sim}(1^\lambda, 1^{|f|}, U(f, m), \{L_{i,f[i]}\}_{i\in[s]})$, and sends $(\widetilde{U}, \{L_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{B}$.

6. $\mathcal{B}$ computes $(\mathsf{pke.vk}_{i,f[i]}, \mathsf{pke.CT}_{i,f[i]}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]}, L_{i,f[i]})$ and $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke.CT}_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L^*_{i,f[i]\oplus 1})$ for every $i \in [s]$.

7. $\mathcal{B}$ sends $(\widetilde{U}, \{\mathsf{pke.CT}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to $\mathcal{A}_1$.

8. $\mathcal{A}_1$ sends $(\mathsf{gc.cert}, \{\mathsf{pke.cert}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ to the challenger, and sends its internal state to $\mathcal{A}_2$.

9. $\mathcal{B}$ sends $\mathsf{gc.cert}$ to the challenger, and receives $\top$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.

10. $\mathcal{B}$ sends $\{\mathsf{pke.sk}_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}}$ to $\mathcal{A}_2$.

11. $\mathcal{A}_2$ outputs b'.

12. $\mathcal{B}$ computes $\mathsf{PKE.Vrfy}$ for all $\mathsf{pke.cert}_{i,\alpha}$, and outputs b' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\bot$.

It is clear that $\Pr[1 \leftarrow \mathcal{B}|b=0] = \Pr[\mathsf{Hyb}_1 = 1]$ and $\Pr[1 \leftarrow \mathcal{B}|b=1] = \Pr[\mathsf{Hyb}_2 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B}|b=0] - \Pr[1 \leftarrow \mathcal{B}|b=1]|$ is non-negligible, which contradicts the certified everlasting selective security of Σ_{cegc}. $\square$

D Proof of Theorem 7.12

Proof of Theorem 7.12. For a given $2n$-qubit, let A be the n-qubit of the first half of the $2n$-qubit, and let B be the n-qubit of the second half of the $2n$-qubit. Let $\mathsf{NAD.Sim}$ be the simulating algorithm of the ciphertext $\mathsf{nad.CT}$. Let us describe how the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ works below.

$\mathsf{Sim}_1(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$:

1. Parse $\mathsf{MPK} = (\mathsf{nad.MPK}, \mathsf{nce.pk})$ and $\mathcal{V} = (f, f(m), (\mathsf{nad.sk}_f, \mathsf{nce.sk}))$ *or* $\emptyset$. [4]
2. Sim_1 does the following:
   - If $\mathcal{V} = \emptyset$, generate $\left|\widetilde{0^n 0^n}\right\rangle$ and $(\mathsf{nce.vk}, \widetilde{\mathsf{nce.CT}}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE.Fake}(\mathsf{nce.pk})$. Let $\Psi_A := \mathrm{Tr}_B(\left|\widetilde{0^n 0^n}\right\rangle\left\langle\widetilde{0^n 0^n}\right|)$ and $\Psi_B := \mathrm{Tr}_A(\left|\widetilde{0^n 0^n}\right\rangle\left\langle\widetilde{0^n 0^n}\right|)$. Output $\mathsf{CT} := (\Psi_A, \widetilde{\mathsf{nce.CT}})$ and $\mathsf{st} := (\mathsf{nce.aux}, \mathsf{nce.pk}, \mathsf{nad.MPK}, \Psi_B, 1^{|m|}, \mathsf{nce.vk}, 0)$.
   - If $\mathcal{V} = (f, f(m), (\mathsf{nad.sk}_f, \mathsf{nce.sk}))$, generate $a, c \leftarrow \{0,1\}^n$, $(\mathsf{nce.vk}, \mathsf{nce.CT}) \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}, (a, c))$, $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Sim}(\mathsf{nad.MPK}, (f, f(m), \mathsf{nad.sk}_f), 1^{|m|})$ and $\Psi := Z^c X^a \mathsf{nad.CT} X^a Z^c$. Output $\mathsf{CT} := (\Psi, \mathsf{nce.CT})$ and $\mathsf{st} := (\mathsf{nad.vk}, \mathsf{nce.vk}, a, c, 1)$.

$\mathsf{Sim}_2(\mathsf{MSK}, f, f(m), \mathsf{st})$:

1. Parse $\mathsf{MSK} := (\mathsf{nad.MSK}, \mathsf{nce.MSK})$ and $\mathsf{st} = (\mathsf{nce.aux}, \mathsf{nce.pk}, \mathsf{nad.MPK}, \Psi_B, 1^{|m|}, \mathsf{nce.vk}, 0)$.
2. Compute $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$.
3. Compute $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Sim}(\mathsf{nad.MPK}, (f, f(m), \mathsf{nad.sk}_f), 1^{|m|})$. Measure the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis and let (x_i, z_i) be the measurement outcome for all $i \in [N]$.
4. Compute $\widetilde{\mathsf{nce.sk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.pk}, \mathsf{nce.MSK}, \mathsf{nce.aux}, (x, z))$.
5. Output $\mathsf{sk}_f := (\mathsf{nad.sk}_f, \widetilde{\mathsf{nce.sk}})$ and $\mathsf{st}' := (\mathsf{nad.vk}, \mathsf{nce.vk}, x, z, 1)$.

$\mathsf{Sim}_3(\mathsf{st}^*)$:

1. Parse $\mathsf{st}^* = (\mathsf{nad.vk}, \mathsf{nce.vk}, x^*, z^*, 1)$ or $\mathsf{st}^* = (\mathsf{nce.aux}, \mathsf{nce.pk}, \mathsf{nad.MPK}, \Psi_B, 1^{|m|}, \mathsf{nce.vk}, 0)$.
2. Sim_3 does the following:
   - If the final bit of st^* is 0, compute $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Sim}(\mathsf{nad.MPK}, \emptyset, 1^{|m|})$. Measure the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis and let (x_i, z_i) be the measurement outcome for all $i \in [N]$. Output $\mathsf{vk} := (\mathsf{nad.vk}, \mathsf{nce.vk}, x, z)$.
   - If the final bit of st^* is 1, output $\mathsf{vk} := (\mathsf{nad.vk}, \mathsf{nce.vk}, x^*, z^*)$.

Let us define the sequence of hybrids as follows.

Hyb_0: This is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}}, \mathcal{A}}(0)$.

[4] If an adversary calls a key query before the adversary receives a challenge ciphertext, then $\mathcal{V} = (f, f(m), (\mathsf{nad.sk}_f, \mathsf{nce.sk}))$. Otherwise, $\mathcal{V} = \emptyset$.

1. The challenger generates $(\mathsf{nad.MPK}, \mathsf{nad.MSK}) \leftarrow \mathsf{NAD.Setup}(1^\lambda)$ and $(\mathsf{nce.pk}, \mathsf{nce.MSK}) \leftarrow \mathsf{NCE.Setup}(1^\lambda)$, and sends $(\mathsf{nad.MPK}, \mathsf{nce.pk})$ to $\mathcal{A}_1$.
2. $\mathcal{A}_1$ is allowed to make an arbitrary key query at most one time. For a key query, the challenger receives $f \in \mathcal{F}$, computes $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$ and $\mathsf{nce.sk} \leftarrow \mathsf{NCE.KeyGen}(\mathsf{nce.MSK})$, and sends $(\mathsf{nad.sk}_f, \mathsf{nce.sk})$ to $\mathcal{A}_1$.
3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to the challenger.
4. The challenger generates $a, c \leftarrow \{0,1\}^n$, computes $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$, $\Psi := Z^c X^a \mathsf{nad.CT} X^a Z^c$ and $(\mathsf{nce.vk}, \mathsf{nce.CT}) \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}, (a, c))$, and sends $(\Psi, \mathsf{nce.CT})$ to $\mathcal{A}_1$.
5. If a key query is not called in step 2, $\mathcal{A}_1$ is allowed to make an arbitrary key query at most one time. For a key query, the challenger receives $f \in \mathcal{F}$, computes $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$ and $\mathsf{nce.sk} \leftarrow \mathsf{NCE.KeyGen}(\mathsf{nce.MSK})$, and sends $(\mathsf{nad.sk}_f, \mathsf{nce.sk})$ to $\mathcal{A}_1$.
6. $\mathcal{A}_1$ sends $(\mathsf{nad.cert}, \mathsf{nce.cert})$ to the challenger and its internal state to $\mathcal{A}_2$.
7. The challenger computes $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(a, c, \mathsf{nad.cert})$. The challenger computes $\mathsf{NCE.Vrfy}(\mathsf{nce.vk}, \mathsf{nce.cert})$ and $\mathsf{NAD.Vrfy}(\mathsf{nad.vk}, \mathsf{nad.cert}^*)$. If the results are $\top$, the challenger outputs $\top$ and sends $(\mathsf{nad.MSK}, \mathsf{nce.MSK})$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$ and sends $\bot$ to $\mathcal{A}_2$.
8. $\mathcal{A}_2$ outputs b'. The output of the experiment is b' if the challenger outputs $\top$. Otherwise, the output of the experiment is $\bot$.

Hyb_1: This is different from Hyb_0 in the following second points. First, when a key query is not called in step 2, the challenger computes $(\mathsf{nce.vk}, \widetilde{\mathsf{nce.CT}}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE.Fake}(\mathsf{nce.pk})$ and sends $(\Psi, \widetilde{\mathsf{nce.CT}})$ to $\mathcal{A}_1$ instead of computing $(\mathsf{nce.vk}, \mathsf{nce.CT}) \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}, (a, c))$ and sending $(\Psi, \mathsf{nce.CT})$ to $\mathcal{A}_1$. Second, in step 5, the challenger computes $\widetilde{\mathsf{nce.sk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.pk}, \mathsf{nce.MSK}, \mathsf{nce.aux}, (a, c))$ and sends $(\mathsf{nad.sk}_f, \mathsf{nce.sk})$ to $\mathcal{A}_1$ instead of computing $\mathsf{nce.sk} \leftarrow \mathsf{NCE.KeyGen}(\mathsf{nce.MSK})$ and sending $(\mathsf{nad.sk}_f, \mathsf{nce.sk})$ to $\mathcal{A}_1$.

Hyb_2: This is different from Hyb_1 in the following three points. First, when a key query is not called in step 2, the challenger generates $\left|\widetilde{0^n 0^n}\right\rangle$ instead of generating $a, c \leftarrow \{0,1\}^n$ and $\Psi = Z^c X^a \mathsf{nad.CT} X^a Z^c$. Let $\Psi_A := \mathrm{Tr}_B(\left|\widetilde{0^n 0^n}\right\rangle \left\langle \widetilde{0^n 0^n}\right|)$ and $\Psi_B := \mathrm{Tr}_A(\left|\widetilde{0^n 0^n}\right\rangle \left\langle \widetilde{0^n 0^n}\right|)$. Second, when a key query is not called in step 2, the challenger sends $(\Psi_A, \widetilde{\mathsf{nce.CT}})$ to $\mathcal{A}_1$ instead of sending $(\Psi, \widetilde{\mathsf{nce.CT}})$ to $\mathcal{A}_1$ and then that measures the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis for all $i \in [n]$. Let (x_i, z_i) be the measurement outcome for all $i \in [n]$. Third, the challenger computes $\widetilde{\mathsf{nce.sk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.pk}, \mathsf{nce.MSK}, \mathsf{nce.aux}, (x, z))$ instead of computing $\widetilde{\mathsf{nce.sk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.pk}, \mathsf{nce.MSK}, \mathsf{nce.aux}, (a, c))$ in step 5 and computes $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(x, z, \mathsf{nad.cert})$ instead of computing $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(a, c, \mathsf{nad.cert})$ in step 7.

Hyb_3: This is different from Hyb_2 in the following three points. First, when a key query is not called in step 2, the challenger does not generate $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$ and measure the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis in step 4. Second, if a key query is called in step 5, the challenger computes $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$ and measures the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis for all $i \in [n]$ after it computes $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$. Third, if a key query is not called throughout the experiment, the challenger computes $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$, measures the i-th qubit of $\mathsf{nad.CT}$ and Ψ_B in the Bell basis after step 5.

Hyb_4: This is identical to Hyb_3 except that the challenger computes $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Sim}(\mathsf{nad.MPK}, \mathcal{V}, 1^{|m|})$ instead of computing $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$, where $\mathcal{V} = (f, f(m), \mathsf{nad.sk}_f)$ if a key query is called and $\mathcal{V} = \emptyset$ if a key query is not called.

From the definition of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, b)$ and $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0) = 1\right]$ and $\Pr[\mathsf{Hyb}_4 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 1) = 1\right]$. Therefore, Theorem 7.12 easily follows from Propositions D.1 to D.4. (Whose proof is given later.) $\square$

Proposition D.1. *If Σ_{cence} is certified everlasting RNC secure, it holds that*

$$|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda).$$

Proposition D.2.

$$\Pr[\mathsf{Hyb}_1 = 1] = \Pr[\mathsf{Hyb}_2 = 1].$$

Proposition D.3.

$$\Pr[\mathsf{Hyb}_2 = 1] = \Pr[\mathsf{Hyb}_3 = 1].$$

Proposition D.4. *If Σ_{cegc} is certified everlasting selective secure, it holds that*

$$|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| \leq \mathsf{negl}(\lambda).$$

Proof of Proposition D.1. When an adversary makes key queries in step 2, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. Hence, we consider the case where the adversary does not make a key query in step 2 below.

We assume that $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the certified everlasting RNC security of Σ_{cence}. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $\mathsf{nce.pk}$ from the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}_{\Sigma_{\mathsf{cence}},\mathcal{B}}(\lambda, b)$, generates $(\mathsf{nad.MPK}, \mathsf{nad.MSK}) \leftarrow \mathsf{NAD.KeyGen}(1^\lambda)$, and sends $(\mathsf{nad.MPK}, \mathsf{nce.pk})$ to $\mathcal{A}_1$.

2. $\mathcal{B}$ receives a message $m \in \mathcal{M}$, computes $(\mathsf{nad.vk}, \mathsf{nad.CT}) \leftarrow \mathsf{NAD.Enc}(\mathsf{nad.MPK}, m)$, generates $a, c \leftarrow \{0,1\}^n$, computes $\Psi := Z^c X^a \mathsf{nad.CT} X^a Z^c$, sends (a, c) to the challenger, receives $(\mathsf{nce.CT}^*, \mathsf{nce.sk}^*)$ from the challenger, and sends $(\Psi, \mathsf{nce.CT}^*)$ to $\mathcal{A}_1$.

3. $\mathcal{A}_1$ is allowed to send a key query at most one time. For a key query, $\mathcal{B}$ receives an function f, generates $\mathsf{nad.sk}_f \leftarrow \mathsf{NAD.KeyGen}(\mathsf{nad.MSK}, f)$, and sends $(\mathsf{nad.sk}_f, \mathsf{nce.sk}^*)$ to $\mathcal{A}_1$.

4. $\mathcal{A}_1$ sends $(\mathsf{nad.cert}, \mathsf{nce.cert})$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.

5. $\mathcal{B}$ sends $\mathsf{nce.cert}$ to the challenger, and receives $\mathsf{nce.MSK}$ or $\perp$ from the challenger. $\mathcal{B}$ computes $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(a, c, \mathsf{nad.cert})$ and $\mathsf{NAD.Vrfy}(\mathsf{nad.vk}, \mathsf{nad.cert}^*)$. If the result is $\top$ and $\mathcal{B}$ receives $\mathsf{nce.MSK}$ from the challenger, $\mathcal{B}$ sends $(\mathsf{nad.MSK}, \mathsf{nce.MSK})$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ outputs $\perp$, sends $\perp$ to $\mathcal{A}_2$, and aborts.

6. $\mathcal{A}_2$ outputs b'.

7. $\mathcal{B}$ outputs b'.

It is clear that $\Pr[1 \leftarrow \mathcal{B} | b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} | b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B} | b = 0] - \Pr[1 \leftarrow \mathcal{B} | b = 1]|$ is non-negligible, which contradicts the certified everlasting RNC security of Σ_{cence}. □

Proof of Proposition D.2. We clarify the difference between Hyb_1 and Hyb_2. First, in Hyb_2, the challenger uses (x, z) instead of using (a, c) as in Hyb_1. Second, in Hyb_2, the challenger sends Ψ_A to $\mathcal{A}_1$ instead of sending $Z^c X^a \mathsf{nad.CT} X^a Z^c$ to $\mathcal{A}_1$ as in Hyb_1. Hence, it is sufficient to prove that x and z are uniformly randomly distributed and Ψ_A is identical to $Z^z X^x \mathsf{nad.CT} X^x Z^z$. These two things are obvious from Lemma 2.2. □

Proof of Proposition D.3. The difference between Hyb_2 and Hyb_3 is only the order of operating the algorithm $\mathsf{NAD.Enc}$ and the Bell measurement on $\mathsf{nad.CT}$ and Ψ_B. Therefore, it is clear that the probability distribution of the ciphertext and the decryption key given to the adversary in Hyb_2 is identical to that the ciphertext and the decryption key given to the adversary in Hyb_3. □

Proof of Proposition D.4. We assume that $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the 1-bounded certified everlasting non-adaptive security of Σ_{nad}. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives nad.MPK from the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}_{\Sigma_{\mathsf{nad}},\mathcal{B}}(\lambda, b)$, generates $(\mathsf{nce.pk}, \mathsf{nce.MSK}) \leftarrow \mathsf{NCE.Setup}(1^\lambda)$, and sends $(\mathsf{nad.MPK}, \mathsf{nce.pk})$ to $\mathcal{A}_1$.

2. $\mathcal{A}_1$ is allowed to call a key query at most one time. For a key query, $\mathcal{B}$ receives f from $\mathcal{A}_1$, sends f to the challenger as a key query, receives $\mathsf{nad.sk}_f$ from the challenger, computes $\mathsf{nce.sk} \leftarrow \mathsf{NCE.KeyGen}(\mathsf{nce.MSK})$, and sends $(\mathsf{nad.sk}_f, \mathsf{nce.sk})$ to $\mathcal{A}_1$.

3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$ and sends m to $\mathcal{B}$.

4. $\mathcal{B}$ does the following.

   - If a key query is called in step 2, $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.CT from the challenger, generates $a, c \leftarrow \{0,1\}^n$, $\Psi := Z^c X^a \mathsf{nad.CT} X^a Z^c$ and $(\mathsf{nce.vk}, \mathsf{nce.CT}) \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.pk}, (a, c))$, and sends $(\Psi, \mathsf{nce.CT})$ to $\mathcal{A}_1$.
   - If a key query is not called in step 2, $\mathcal{B}$ generates $\left|\widetilde{0^n0^n}\right\rangle$. Let $\Psi_A := \mathrm{Tr}_B(\left|\widetilde{0^n0^n}\right\rangle\left\langle\widetilde{0^n0^n}\right|)$ and $\Psi_B := \mathrm{Tr}_A(\left|\widetilde{0^n0^n}\right\rangle\left\langle\widetilde{0^n0^n}\right|)$. $\mathcal{B}$ computes $(\mathsf{nce.vk}, \widetilde{\mathsf{nce.CT}}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE.Fake}(\mathsf{nce.pk})$ and sends $(\Psi_A, \widetilde{\mathsf{nce.CT}})$ to $\mathcal{A}_1$.

5. If a key query is not called in step 2, $\mathcal{A}_1$ is allowed to make a key query at most one time. If $\mathcal{B}$ receives an function f as key query, $\mathcal{B}$ sends f to the challenger as key query, and receives $\mathsf{nad.sk}_f$ from the challenger. $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.CT, measures the i-th qubit of nad.CT and Ψ_B in the Bell basis, and let (x_i, z_i) be the measurement outcome for all $i \in [n]$. $\mathcal{B}$ computes $\widetilde{\mathsf{nce.sk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.pk}, \mathsf{nce.MSK}, \mathsf{nce.aux}, (x, z))$ and sends $(\mathsf{nad.sk}_f, \widetilde{\mathsf{nce.sk}})$ to $\mathcal{A}_1$.

6. If $\mathcal{B}$ does not receive a key query throughout the experiment, $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.CT, and measures the i-th qubit of nad.CT and Ψ_B in the Bell basis and let (x_i, z_i) be the measurement outcome for all $i \in [n]$.

7. $\mathcal{A}_1$ sends $(\mathsf{nad.cert}, \mathsf{nce.cert})$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.

8. $\mathcal{B}$ computes $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Modify}(x^*, z^*, \mathsf{nad.cert})$, where $(x^*, z^*) = (a, c)$ if a key query is called in step 2 and $(x^*, z^*) = (x, z)$ if a key query is not called in step 2. $\mathcal{B}$ sends nad.cert to the challenger, and receives nad.MSK or $\bot$ from the challenger. $\mathcal{B}$ computes $\mathsf{NCE.Vrfy}(\mathsf{nce.vk}, \mathsf{nce.cert})$. If the result is $\top$ and $\mathcal{B}$ receives nad.MSK from the challenger, $\mathcal{B}$ sends $(\mathsf{nad.MSK}, \mathsf{nce.MSK})$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ outputs $\bot$, sends $\bot$ to $\mathcal{A}_2$, and aborts.

9. $\mathcal{A}_2$ outputs b'.

10. $\mathcal{B}$ outputs b'.

It is clear that $\Pr[1 \leftarrow \mathcal{B} | b = 0] = \Pr[\mathsf{Hyb}_3 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} | b = 1] = \Pr[\mathsf{Hyb}_4 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B} | b = 0] - \Pr[1 \leftarrow \mathcal{B} | b = 1]|$ is non-negligible, which contradicts the 1-bounded certified everlasting non-adaptive security of Σ_{nad}. $\square$

E Proof of Theorem 7.14

Proof of Theorem 7.14. Let us denote the simulating algorithm of Σ_{one} as $\mathsf{ONE.Sim} = \mathsf{ONE.}(\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$. Let us describe how the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ works below.

$\mathsf{Sim}_1(\mathsf{MPK}, \mathcal{V}, 1^{|x|})$**:** Let q^* be the number of times that $\mathcal{A}_1$ has made key queries before it sends a challenge query.

1. Parse $\mathsf{MPK} := \{\mathsf{one.MPK}_i\}_{i \in [N]}$ and $\mathcal{V} := \{C_j, C_j(x), (\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j, \Delta_j, i}\}_{i \in [\Gamma_j]})\}_{j \in [q^*]}$.

2. Generate a uniformly random set $\Gamma_i \subseteq [N]$ of size $Dt+1$ and a uniformly random set $\Delta_i \subseteq [S]$ of size v for all $i \in \{q^*+1, \cdots, q\}$. Let $\Delta_0 := \emptyset$. Let $\mathcal{L} := \bigcup_{i \neq i'} (\Gamma_i \cap \Gamma_{i'})$. Sim_1 aborts if $|\mathcal{L}| > t$ or there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j \neq i} \Delta_j) = \emptyset$.
3. Sim_1 uniformly and independently samples ℓ random degree t polynomials $\mu_1, \cdots, \mu_\ell$ whose constant terms are all 0.
4. Sim_1 samples the polynomials $\xi_1, \cdots, \xi_S$ as follows for $j \in [q]$:
   - fix $a^* \in \Delta_j \setminus (\Delta_0 \cup \cdots \cup \Delta_{j-1})$;
   - for all $a \in (\Delta_j \setminus (\Delta_0 \cup \cdots \cup \Delta_{j-1})) \setminus \{a^*\}$, set ξ_a to be a uniformly random degree Dt polynomial whose constant term is 0;
   - if $j \le q^*$, pick a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$; if $j > q^*$, pick random values for $\eta_j(i)$ for all $i \in \mathcal{L}$;
   - the evaluation of ξ_{a^*} on the points in $\mathcal{L}$ is defined by the relation:
   $$\eta_j(\cdot) = C_j(\mu_1(\cdot), \cdots, \mu_\ell(\cdot)) + \sum_{a \in \Delta_j} \xi_a(\cdot).$$
   - Finally, for all $a \notin (\Delta_1 \cup \cdots \cup \Delta_q)$, set ξ_a to be a uniformly random degree Dt polynomial whose constant term is 0.
5. For each $i \in \mathcal{L}$, Sim_1 computes
   $$(\mathsf{one.vk}_i, \mathsf{one.CT}_i) \leftarrow \mathsf{ONE.Enc}(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i))).$$
6. For each $i \notin \mathcal{L}$, Sim_1 does the following:
   - If $i \in \Gamma_j$ for some $j \in [q^*]$ [5], computes
   $$(\mathsf{one.CT}_i, \mathsf{one.st}_i) \leftarrow \mathsf{ONE.Sim}_1(\mathsf{one.MPK}_i, (G_{C_j,\Delta_j,i}, \eta_j(i), \mathsf{one.sk}_{C_j,\Delta_j,i}), 1^{|m|}).$$
   - If $i \notin \Gamma_j$ for all $j \in [q^*]$, computes
   $$(\mathsf{one.CT}_i, \mathsf{one.st}_i) \leftarrow \mathsf{ONE.Sim}_1(\mathsf{one.MPK}_i, \emptyset, 1^{|m|}).$$
7. Output $\mathsf{CT} := \{\mathsf{one.CT}_i\}_{i \in [N]}$ and $\mathsf{st} := (\{\Gamma_i\}_{i \in [q]}, \{\Delta_i\}_{i \in [q]}, \{\eta_j(i)\}_{j \in \{q^*+1, \cdots, q\}, i \in \mathcal{L}}, \{\mathsf{one.st}_i\}_{i \in [N] \setminus \mathcal{L}}, \{\mathsf{one.vk}_i\}_{i \in \mathcal{L}})$.

$\mathsf{Sim}_2(\mathsf{MSK}, C_j, C_j(x), \mathsf{st})$**:** The simulator simulates the j-th key query for $j > q^*$.

1. Parse $\mathsf{MSK} := \{\mathsf{one.MSK}_i\}_{i \in [N]}$ and $\mathsf{st}_{j-1} := (\{\Gamma_i\}_{i \in [q]}, \{\Delta_i\}_{i \in [q]}, \{\eta_s(i)\}_{s \in \{q^*+1, \cdots, q\}, i \in \mathcal{L}}, \{\mathsf{one.st}_i\}_{i \in [N] \setminus \mathcal{L}}, \{\mathsf{one.vk}_i\}_{i \in \mathcal{L}})$.
2. For each $i \in \Gamma_j \cap \mathcal{L}$, generate $\mathsf{one.sk}_{C_j,\Delta_j,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C_j,\Delta_j})$.
3. For each $i \in \Gamma_j \setminus \mathcal{L}$, generate a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$ and subject to the constraints on the values in $\mathcal{L}$ chosen earlier, and generate
   $$(\mathsf{one.sk}_{C_j,\Delta_j,i}, \mathsf{one.st}_i^*) \leftarrow \mathsf{ONE.Sim}_2(\mathsf{one.MSK}_i, \eta_j(i), G_{C_j,\Delta_j}, \mathsf{one.st}_i).$$
   For simplicity, let us denote $\mathsf{one.st}_i^*$ as $\mathsf{one.st}_i$ for $i \in \Gamma_j \setminus \mathcal{L}$.
4. Output $\mathsf{sk}_{C_j} := (\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i \in \Gamma_j})$ and $\mathsf{st}_j := (\{\Gamma_i\}_{i \in [q]}, \{\Delta_i\}_{i \in [q]}, \{\eta_j(i)\}_{j \in \{q^*+1, \cdots, q\}, i \in \mathcal{L}}, \{\mathsf{one.st}_i\}_{i \in [N] \setminus \mathcal{L}}, \{\mathsf{one.vk}_i\}_{i \in \mathcal{L}})$.

$\mathsf{Sim}_3(\mathsf{st}^*)$**:** The simulator simulates a verification key.

[5] Note that j is uniquely determined since $i \notin \mathcal{L}$.

1. Parse $\mathsf{st}^* := (\{\Gamma_i\}_{i\in[q]}, \{\Delta_i\}_{i\in[q]}, \{\eta_j(i)\}_{j\in\{q^*+1,\cdots,q\},i\in\mathcal{L}}, \{\mathsf{one.st}_i\}_{i\in[N]\setminus\mathcal{L}}, \{\mathsf{one.vk}_i\}_{i\in\mathcal{L}})$.
2. For each $i \in [N]\setminus\mathcal{L}$, compute $\mathsf{one.vk}_i \leftarrow \mathsf{ONE.Sim}_3(\mathsf{one.st}_i)$.
3. Output $\mathsf{vk} := \{\mathsf{one.vk}_i\}_{i\in[N]}$.

Let us define the sequence of hybrids as follows.

Hyb_0**:** This is identical to $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0)$.

1. The challenger generates $(\mathsf{one.MPK}_i, \mathsf{one.MSK}_i) \leftarrow \mathsf{ONE.Setup}(1^\lambda)$ for $i \in [N]$.
2. $\mathcal{A}_1$ is allowed to call key queries at most q times. For the j-th key query, the challenger receives an function C_j from $\mathcal{A}_1$, generates a uniformly random set $\Gamma_j \in [N]$ of size $Dt+1$ and $\Delta_j \in [S]$ of size v. For $i \in \Gamma_j$, the challenger generates $\mathsf{one.sk}_{C_j,\Delta_j,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C_j,\Delta_j})$, and sends $(\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j})$ to $\mathcal{A}_1$. Let q^* be the number of times that $\mathcal{A}_1$ has called key queries in this step.
3. $\mathcal{A}_1$ chooses $x \in \mathcal{M}$ and sends x to the challenger.
4. The challenger generates a random degree t polynomial $\mu_i(\cdot)$ whose constant term is $x[i]$ for $i \in [\ell]$ and a random degree Dt polynomial $\xi_i(\cdot)$ whose constant term is 0. For $i \in [N]$, the challenger computes $(\mathsf{one.vk}_i, \mathsf{one.CT}_i) \leftarrow \mathsf{ONE.Enc}(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)))$, and sends $\{\mathsf{one.CT}_i\}_{i\in[N]}$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ is allowed to call a key query at most $q-q^*$ times. For the j-th key query, the challenger receives an function C_j from $\mathcal{A}_1$, generates a uniformly random set $\Gamma_j \in [N]$ of size $Dt+1$ and $\Delta_j \in [S]$ of size v. For $i \in \Gamma_j$, the challenger generates $\mathsf{one.sk}_{C_j,\Delta_j,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C_j,\Delta_j})$, and sends $(\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j})$ to $\mathcal{A}_1$.
6. $\mathcal{A}_1$ sends $\{\mathsf{one.cert}_i\}_{i\in[N]}$ to the challenger and its internal state to $\mathcal{A}_2$.
7. If $\top \leftarrow \mathsf{ONE.Vrfy}(\mathsf{one.vk}_i, \mathsf{one.cert}_i)$ for all $i \in [N]$, the challenger outputs $\top$ and sends $\{\mathsf{one.MSK}_i\}_{i\in[N]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\bot$ and sends $\bot$ to $\mathcal{A}_2$.
8. $\mathcal{A}_2$ outputs b.
9. The experiment outputs b if the challenger outputs $\top$. Otherwise, the experiment outputs $\bot$.

Hyb_1**:** This is identical to Hyb_0 except for the following three points. First, the challenger generates uniformly random set $\Gamma_i \in [N]$ of size $Dt+1$ and $\Delta_i \in [S]$ of size v for $i \in \{q^*+1, \cdots, q\}$ in step 4 instead of generating them when a key query is called. Second, if $|\mathcal{L}| > t$, the challenger aborts and the experiment outputs $\bot$. Third, if there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j\neq i} \Delta_j) = \emptyset$, the challenger aborts and the experiment outputs $\bot$.

Hyb_2**:** This is identical to Hyb_1 except that the challenger samples $\xi_1, \cdots, \xi_S, \eta_1, \cdots, \eta_q$ as in the simulator Sim_1 described above.

Hyb_3**:** This is identical to Hyb_2 except that the challenger generates $\{\mathsf{one.CT}_i\}_{i\in[N]\setminus\{\mathcal{L}\}}$, $\{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j}$ for $j \in \{q^*+1, \cdots, q'\}$, and $\mathsf{vk} := \{\mathsf{one.vk}_i\}_{i\in[N]\setminus\{\mathcal{L}\}}$ as in the simulator $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$ described above, where q' is the number of key queries that the adversary makes in total.

Hyb_4**:** This is identical to Hyb_3 except that the challenger generates $\mu_1, \cdots, \mu_\ell$ as in the simulator Sim_1 described above.

From the definition of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, b)$ and $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2, \mathsf{Sim}_3)$, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0) = 1\right]$ and $\Pr[\mathsf{Hyb}_4 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 1) = 1\right]$. Therefore, Theorem 7.14 easily follows from Propositions E.1 to E.4 (whose proofs are given later). $\square$

Proposition E.1.

$$|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda).$$

Proposition E.2.

$$\Pr[\mathsf{Hyb}_1 = 1] = \Pr[\mathsf{Hyb}_2 = 1].$$

Proposition E.3. *If Σ_{one} is 1-bounded certified everlasting adaptive secure,*

$$|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| \leq \mathsf{negl}(\lambda).$$

Proposition E.4.

$$\Pr[\mathsf{Hyb}_3 = 1] = \Pr[\mathsf{Hyb}_4 = 1].$$

Proof of Proposition E.1. Let $\mathsf{Hyb}_0^{'}$ be the experiment identical to Hyb_0 except that the challenger generates a set $\Gamma_i \in [N]$ and $\Delta_i \in [S]$ for $i \in \{q^*+1, \cdots, q\}$ in step 4. It is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Hyb}_0^{'} = 1\right]$.

Let Hyb_0^* be the experiment identical to $\mathsf{Hyb}_0^{'}$ except that it outputs $\perp$ if $|\mathcal{L}| > t$. It is clear that $\Pr\left[\mathsf{Hyb}_0^{'} = 1 \wedge (|\mathcal{L}| \leq t)\right] = \Pr[\mathsf{Hyb}_0^* = 1 \wedge (|\mathcal{L}| \leq t)]$. Hence, it holds that

$$\left|\Pr\left[\mathsf{Hyb}_0^{'} = 1\right] - \Pr[\mathsf{Hyb}_0^* = 1]\right| \leq \Pr[|\mathcal{L}| > t]$$

from Lemma 2.3.

Let $\mathsf{Collide}$ be the event that there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j \neq i} \Delta_j) = \emptyset$. Hyb_0^* is identical to Hyb_1 when $\mathsf{Collide}$ does not occur. Hence, it is clear that $\Pr\left[\mathsf{Hyb}_0^* = 1 \wedge \overline{\mathsf{Collide}}\right] = \Pr\left[\mathsf{Hyb}_1 = 1 \wedge \overline{\mathsf{Collide}}\right]$. Therefore, it holds that

$$|\Pr[\mathsf{Hyb}_0^* = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \Pr[\mathsf{Collide}]$$

from Lemma 2.3.

From the discussion above, we have

$$|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \Pr[|\mathcal{L}| > t] + \Pr[\mathsf{Collide}].$$

The following Lemmata E.5 and E.6 shows that $\Pr[|\mathcal{L}| > t] \leq 2^{-\Omega(\lambda)}$ and $\Pr[\mathsf{Collide}] \leq q2^{-\Omega(\lambda)}$, which completes the proof. ☐

Lemma E.5 ([GVW12]). *Let $\Gamma_1, \cdots, \Gamma_q \subseteq [N]$ be randomly chosen subsets of size $tD+1$. Let $t = \Theta(q^2\lambda)$ and $N = \Theta(D^2q^2t)$. Then,*

$$\Pr\left[\left|\bigcup_{i \neq i'} (\Gamma_i \cap \Gamma_i)\right| > t\right] \leq 2^{-\Omega(\lambda)}$$

where the probability is over the random choice of the subsets $\Gamma_1, \cdots, \Gamma_q$.

Lemma E.6 ([GVW12]). *Let $\Delta_1, \cdots, \Delta_q \subseteq [S]$ be randomly chosen subsets of size v. Let $v(\lambda) = \Theta(\lambda)$ and $S(\lambda) = \Theta(vq^2)$. Let $\mathsf{Collide}$ be the event that there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j \neq i} \Delta_j) = \emptyset$. Then, we have*

$$\Pr[\mathsf{Collide}] \leq q2^{-\Omega(\lambda)}$$

where the probability is over the random choice of subsets $\Delta_1, \cdots, \Delta_q$.

Proof of Proposition E.2. In the encryption in Hyb_1, ξ_{a^*} is chosen at random and $\eta_j(\cdot)$ is defined by the relation. Sim essentially chooses $\eta_j(\cdot)$ at random which defines ξ_{a^*}. It is easy to see that reversing the order of how the polynomials are chosen produces the same distribution. ☐

Proof of Proposition E.3. To prove the proposition, let us define a hybrid experiment Hyb_2^s for each $s \in [N]$ as follows.

Hyb_2^s**:** This is identical to Hyb_2 except for the following three points. First, the challenger generates $\{\mathsf{one.CT}_i\}_{i\in[s]\setminus\mathcal{L}}$ as in the simulator Sim_1. Second, the challenger generates $\{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j\cap[s]}$ for $j \in \{q^*+1,\cdots,q'\}$ as in the simulator Sim_2, where q' is the number of key queries that the adversary makes in total. Third, the challenger generates $\{\mathsf{one.vk}_i\}_{i\in[s]\setminus\mathcal{L}}$ as in the simulator Sim_3.

Let us denote Hyb_2 as Hyb_2^0. It is clear that $\Pr\left[\mathsf{Hyb}_2^N = 1\right] = \Pr[\mathsf{Hyb}_3 = 1]$. Furthermore, we can show that

$$\left|\Pr\left[\mathsf{Hyb}_2^{s-1} = 1\right] - \Pr[\mathsf{Hyb}_2^s = 1]\right| \leq \mathsf{negl}(\lambda)$$

for $s \in [N]$. (Its proof is given later.) From these facts, we obtain Proposition E.3.

Let us show the remaining one. In the case $s \in \mathcal{L}$, it is clear that Hyb_2^{s-1} is identical to Hyb_2^s. Hence, we consider the case $s \notin \mathcal{L}$. To show the inequality above, let us assume that $\left|\Pr\left[\mathsf{Hyb}_2^{s-1} = 1\right] - \Pr[\mathsf{Hyb}_2^s = 1]\right|$ is non-negligible. Then, we can construct an adversary $\mathcal{B}$ that can break the 1-bounded certified everlasting adaptive security of Σ_{one} as follows.

1. $\mathcal{B}$ receives $\mathsf{one.MPK}$ from the challenger of $\mathsf{Exp}_{\Sigma_{\mathsf{one}},\mathcal{A}}^{\mathsf{cert\text{-}ever\text{-}adapt}}(\lambda, b)$. $\mathcal{B}$ sets $\mathsf{one.MPK}_s := \mathsf{one.MPK}$.

2. $\mathcal{B}$ generates $(\mathsf{one.MPK}_i, \mathsf{one.MSK}_i) \leftarrow \mathsf{ONE.Setup}(1^\lambda)$ for all $i \in [N] \setminus s$, and sends $\{\mathsf{one.MPK}_i\}_{i\in[N]}$ to $\mathcal{A}_1$.

3. $\mathcal{A}_1$ is allowed to call key queries at most q times. For the j-th key query, $\mathcal{B}$ receives an function C_j from $\mathcal{A}_1$, generates a uniformly random set $\Gamma_j \in [N]$ of size $Dt+1$ and $\Delta_j \in [S]$ of size v. For $i \in \Gamma_j \setminus s$, $\mathcal{B}$ generates $\mathsf{one.sk}_{C_j,\Delta_j,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C_j,\Delta_j})$. If $s \in \Gamma_j$, $\mathcal{B}$ sends G_{C_j,Δ_j} to the challenger, receives $\mathsf{one.sk}_{C_j,\Delta_j,s}$ from the challenger, and sends $(\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j})$ to $\mathcal{A}_1$. Let q^* be the number of times that $\mathcal{A}_1$ has called key queries in this step.

4. $\mathcal{A}_1$ chooses $x \in \mathcal{M}$, and sends x to $\mathcal{B}$.

5. $\mathcal{B}$ generates uniformly random set $\Gamma_i \in [N]$ of size $Dt+1$ and $\Delta_i \in [S]$ of size v for $i \in \{q^*+1,\cdots,q\}$. $\mathcal{B}$ generates a random degree t polynomial $\mu_i(\cdot)$ whose constant term is $x[i]$ for $i \in [\ell]$, and $\xi_1,\cdots,\xi_S,\eta_1,\cdots,\eta_q$ as in the simulator Sim_1. For $i \in [s-1] \setminus \mathcal{L}$, $\mathcal{B}$ generates $\mathsf{one.CT}_i$ as in the simulator Sim_1. For $i \in \{s+1,\cdots N\} \cup \mathcal{L}$, $\mathcal{B}$ generates $(\mathsf{one.vk}_i, \mathsf{one.CT}_i) \leftarrow \mathsf{ONE.Enc}(\mathsf{one.MPK}_i, (\mu_1(i),\cdots,\mu_\ell(i),\xi_1(i),\cdots,\xi_S(i)))$. $\mathcal{B}$ sends $\mu_1(s),\cdots,\mu_\ell(s),\xi_1(s),\cdots,\xi_S(s)$ to the challenger, and receives $\mathsf{one.CT}_s$ from the challenger. $\mathcal{B}$ sends $\{\mathsf{one.CT}_i\}_{i\in[N]}$ to $\mathcal{A}_1$.

6. $\mathcal{A}_1$ is allowed to call key queries at most $q - q^*$ times. For the j-th key query, $\mathcal{B}$ receives an function C_j from $\mathcal{A}_1$. For $i \in \Gamma_j \setminus [s]$, $\mathcal{B}$ generates $\mathsf{one.sk}_{C_j,\Delta_j,i} \leftarrow \mathsf{ONE.KeyGen}(\mathsf{one.MSK}_i, G_{C_j,\Delta_j})$. For $i \in \Gamma_j \wedge [s-1]$, $\mathcal{B}$ generates $\mathsf{one.sk}_{C_j,\Delta_j,i}$ as in the simulator Sim_2. If $s \in \Gamma_j$, $\mathcal{B}$ sends G_{C_j,Δ_j} to the challenger, and receives $\mathsf{one.sk}_{C_j,\Delta_j,s}$ from the challenger. $\mathcal{B}$ sends $(\Gamma_j, \Delta_j, \{\mathsf{one.sk}_{C_j,\Delta_j,i}\}_{i\in\Gamma_j})$ to $\mathcal{A}_1$.

7. For $i \in [s-1] \setminus \mathcal{L}$, $\mathcal{B}$ generates $\mathsf{one.vk}_i$ as in the simulator Sim_3 [6].

8. $\mathcal{A}_1$ sends $\{\mathsf{one.cert}_i\}_{i\in[N]}$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.

9. $\mathcal{B}$ sends $\mathsf{one.cert}_s$ to the challenger, and receives $\mathsf{one.MSK}_s$ or $\perp$ from the challenger. $\mathcal{B}$ computes $\mathsf{ONE.Vrfy}(\mathsf{one.vk}_i, \mathsf{one.cert}_i)$ for all $i \in [N] \setminus s$. If the results are $\top$ and $\mathcal{B}$ receives $\mathsf{one.MSK}_s$ from the challenger, $\mathcal{B}$ sends $\{\mathsf{one.MSK}_i\}_{i\in[N]}$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ aborts.

10. $\mathcal{A}_2$ outputs b'.

11. $\mathcal{B}$ outputs b'.

[6]For $i \in \{s+1,\cdots N\} \cup \mathcal{L}$, $\mathcal{B}$ generated $\mathsf{one.vk}_i$ in step 5.

It is clear that $\Pr[1 \leftarrow \mathcal{B}|b=0] = \Pr\left[\mathsf{Hyb}_2^{s-1}=1\right]$ and $\Pr[1 \leftarrow \mathcal{B}|b=1] = \Pr[\mathsf{Hyb}_2^{s}=1]$. By assumption, $\left|\Pr\left[\mathsf{Hyb}_2^{s-1}=1\right] - \Pr[\mathsf{Hyb}_2^{s}=1]\right|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B}|b=0] - \Pr[1 \leftarrow \mathcal{B}|b=1]|$ is non-negligible, which contradicts the 1-bounded certified everlasting adaptive security of Σ_{one}. $\square$

Proof of Proposition E.4. In Hyb_3, the polynomials $\mu_1, \cdots, \mu_\ell$ are chosen with constant terms $x_1, \cdots, x_\ell$, respectively. In Hyb_4, these polynomials are now chosen with 0 constant terms. This only affects the distribution of $\mu_1, \cdots, \mu_\ell$ themselves and polynomials $\xi_1, \cdots, \xi_S$. Moreover, only the evaluations of these polynomials on the points in $\mathcal{L}$ affect the outputs of the experiments. Now observe that:

- The distribution of the values $\{\mu_1(i), \cdots, \mu_\ell(i)\}_{i\in\mathcal{L}}$ are identical to both Hyb_3 and Hyb_4. This is because in both experiments, we choose these polynomials to be random degree t polynomials (with different constraints in the constant term), so their evaluation on the points in $\mathcal{L}$ are identically distributed, since $|\mathcal{L}| \leq t$.
- The values $\{\xi_1(i), \cdots, \xi_S(i)\}_{i\in\mathcal{L}}$ depend only on the values $\{\mu_1(i), \cdots, \mu_\ell(i)\}_{i\in\mathcal{L}}$.

Proposition E.4 follows from these observations. $\square$