Certified Everlasting Secure Collusion-Resistant Functional Encryption, and More

Taiga Hiroka[⋆], Fuyuki Kitagawa[†♢], Tomoyuki Morimae[⋆],
Ryo Nishimaki[†♢], Tapas Pal[♭], Takashi Yamakawa[†♢⋆]

[⋆]Yukawa Institute for Theoretical Physics, Kyoto University, Japan
{taiga.hiroka,tomoyuki.morimae}@yukawa.kyoto-u.ac.jp
[†]NTT Social Informatics Laboratories, Tokyo, Japan
{fuyuki.kitagawa,ryo.nishimaki,takashi.yamakawa}@ntt.com
[♢]NTT Research Center for Theoretical Quantum Information
[♭*]Karlsruhe Institute of Technology, KASTEL Security Research Labs, Germany
tapas.real@gmail.com

March 29, 2024

Abstract

We study certified everlasting secure functional encryption (FE) and many other cryptographic primitives in this work. Certified everlasting security roughly means the following. A receiver possessing a quantum cryptographic object (such as ciphertext) can issue a certificate showing that the receiver has deleted the cryptographic object and information included in the object (such as plaintext) was lost. If the certificate is valid, the security is guaranteed even if the receiver becomes computationally unbounded after the deletion. Many cryptographic primitives are known to be impossible (or unlikely) to have information-theoretical security even in the quantum world. Hence, certified everlasting security is a nice compromise (intrinsic to quantum).

In this work, we define certified everlasting secure versions of FE, compute-and-compare obfuscation, predicate encryption (PE), secret-key encryption (SKE), public-key encryption (PKE), receiver non-committing encryption (RNCE), and garbled circuits. We also present the following constructions:

- Adaptively certified everlasting secure collusion-resistant public-key FE for all polynomial-size circuits from indistinguishability obfuscation and one-way functions.
- Adaptively certified everlasting secure bounded collusion-resistant public-key FE for NC^1 circuits from standard PKE.
- Certified everlasting secure compute-and-compare obfuscation from standard fully homomorphic encryption and standard compute-and-compare obfuscation.
- Adaptively (resp., selectively) certified everlasting secure PE from standard adaptively (resp., selectively) secure attribute-based encryption and certified everlasting secure compute-and-compare obfuscation.
- Certified everlasting secure SKE and PKE from standard SKE and PKE, respectively.
- Cetified everlasting secure RNCE from standard PKE.
- Cetified everlasting secure garbled circuits from standard SKE.

[*]The research was conducted while the author was a postdoc at NTT Social Informatics Laboratories

Contents

1 Introduction

1.1 Background

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

Good compromises (intrinsic to quantum!) have been studied recently [Unr15, BI20, KT20, HMNY21, HMNY22b, Por23]. In particular, certified everlasting security, which was introduced in [HMNY22b] based on [Unr15, BI20], achieves the following security: After receiving quantum-encrypted data, a receiver can issue a certificate to prove that (s)he deleted its quantum-encrypted data. If the certificate is valid, its security is guaranteed even if the receiver becomes computationally unbounded later. A (private or public) verification key for certificates is also generated along with quantum-encrypted data. This security notion is weaker than information-theoretical security since a malicious receiver could refuse to issue a valid certificate. However, it is still a useful security notion because, for example, a sender can penalize receivers who do not issue valid certificates. In addition, certified everlasting security is an 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 [HMNY22b]. 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 and achieves considerable 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 ct_m of a message m is decrypted by sk_f, we can obtain the value $f(m)$, and no information beyond $f(m)$ is leaked. Information-theoretically secure FE is impossible, and all known constructions are computationally secure [GVW12, GGH$^+$16, AP20, AV19, JLS21, JLS22]. A motivating application of FE is analyzing sensitive data and computing new data from personal data without sacrificing data privacy. In this example, users must store their encrypted data on a remote server since users delegate the computation. At some point, users might request the server to "forget" their data (even if they are encrypted). European Union [GDP16] and California [CCP18] adopted data deletion clauses in legal regulations for such users. Encryption with certified deletion could be useful for implementing the right to be forgotten. However, suppose that FE does not have *certified everlasting security*. In that case, the rapid growth of computational power potentially breaks the privacy of sensitive personal data (such as DNA) in the future. This risk ("recalling" in the future) is great because descendants inherit DNA information. Certified everlasting security is desirable for such practical applications of FE.

Hence, we have the following open problem:

Is it possible to construct certified everlasting secure FE?

We note that certified everlasting secure FE is particularly useful compared to certified everlasting secure public key encryption (PKE) (or more generally "all-or-nothing encryption"[2] [GMM17]) 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 for a function f cannot learn more than $f(m)$ even if the receiver can run an 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 keep a copy of a plaintext after honestly decrypting a ciphertext.

Another useful advanced cryptographic primitive is obfuscation for compute-and-compare programs [WZ17] (a.k.a. lockable obfuscation [GKW17]). A compute-and-compare obfuscation scheme can obfuscate a compute-and-compare circuit parameterized by a polynomial-time computable circuit P along with a lock value lock and

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

[2] Such as identity-based encryption (IBE), attribute-based encryption (ABE), fully homomorphic encryption (FHE), or witness encryption (WE).

a message m. The circuit takes an input x and outputs m if $P(x) = \mathsf{lock}$ and $\perp$ otherwise. Point functions, conjunction with wild cards, plaintext checkers, and affine testers are examples of such circuits [GKW17, WZ17]. Hence, certified everlasting secure compute-and-compare obfuscation achieves certified deletion for obfuscated programs in the restricted class of functionalities. In addition, compute-and-compare obfuscation has many cryptographic applications [GKW17, WZ17, CVW+18, FFMV23, AYY22, AKYY23]. We can generically convert all-or-nothing encryption into anonymous one via compute-and-compare obfuscation. In particular, we can obtain predicate encryption (PE) [KSW08, GVW15b] from ABE and compute-and-compare obfuscation. PE is an attribute-hiding variant of ABE and an intermediate primitive between ABE and FE. If we can achieve certified everlasting secure compute-and-compare obfuscation, it is possible to achieve certified everlasting secure PE (and anonymous IBE and PKE).

Hence, we have the following second open problem:

Is it possible to construct certified everlasting secure compute-and-compare obfuscation?

1.2 Our Results

We solve the above questions in this work. Our contributions are as follows.

1. We formally define certified everlasting versions of many cryptographic primitives: FE (Section 3.1), compute-and-compare obfuscation (Section 5.1), PE (Section 6.1), secret-key encryption (SKE) (Appendix C.1), PKE (Appendix C.1), receiver non-committing encryption (RNCE) (Appendix D.1), and a garbling scheme (Appendix E.1).

2. We construct adaptively certified everlasting secure collusion-resistant public-key FE for P/poly from indistinguishability obfuscation (IO) and one-way functions (OWFs) (Section 3.3). We also construct adaptively certified everlasting secure bounded collusion-resistant public-key FE for NC^1 from standard PKE (Section 4.4).

3. We construct certified everlasting secure compute-and-compare obfuscation from standard FHE and standard compute-and-compare obfuscation (Section 5.2). Both building blocks can be instantiated with the learning with errors (LWE) assumption. We also construct adaptively (resp., selectively) certified everlasting secure PE from standard adaptively (resp., selectively) secure ABE and certified everlasting secure compute-and-compare obfuscation (Section 6.2).

4. To achieve adaptively certified everlasting secure bounded collusion-resistant FE, we construct many certified everlasting secure cryptographic primitives:
   - Two constructions of certified everlasting secure SKE from standard SKE (Appendices C.2 and C.3). 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 model (QROM) [BDF+11]. The security of the second construction holds without relying on the QROM, but the certificate is quantum.
   - Two constructions of certified everlasting secure PKE with the same properties of the SKE constructions above from standard PKE (Appendices C.4 and C.5).
   - A construction of certified everlasting secure RNCE from certified everlasting PKE (Appendix D.2).
   - A construction of certified everlasting secure garbling scheme for P/poly from certified everlasting SKE (Appendix E.2).

All our constructions are privately verifiable, so we must keep verification keys (for deletion certificate) secret. It is open to achieving certified everlasting secure bounded collusion-resistant FE for P/poly from standard PKE.

We introduce fascinating techniques to achieve certified everlasting secure collusion-resistant FE and certified everlasting secure compute-and-compare obfuscation. We developed an authentication technique for BB84 state to satisfy both the functionality of FE and certified everlasting security. (See Section 1.5 for the detail.) This authentication technique for BB84 states is of independent interest and we believe that it has further applications.[3] We also developed a deferred evaluation technique using dummy lock values to satisfy both the functionality of compute-and-compare obfuscation and certified everlasting security. (See Section 1.7 for the detail.)

[3] Indeed, an application was found by Kitagawa, Nishimaki, and Yamakawa [KNY23]. See Section 1.4.

1.3 Concurrent and Independent Work

Certified everlasting secure SKE and PKE. Recently, Bartusek and Khurana concurrently and independently obtained similar results [BK23]. They introduce a generic compiler that can convert several cryptographic primitives to certified everlasting secure ones, such as PKE, ABE, FHE, WE, and timed-release encryption. Their constructions via the generic compiler have the advantage that the certificates are classical *and* no QROM is required. Our constructions of certified everlasting SKE and PKE cannot achieve both: if the certificates are classical, QROM is required, and if QROM is not used, the certificates have to be quantum. We note that their certified everlasting SKE and PKE can be used as building blocks of our RNCE, garbling, and bounded collusion-resistant FE constructions instead of our SKE and PKE schemes.

While their work focuses on all-or-nothing encryption, our work presents certified everlasting secure garbling and FE, which are not given in their work. It is unclear how to apply their generic compiler to garbling and FE.

One might think that certified everlasting garbling can be constructed from certified everlasting SKE, which is constructed from their generic compiler. However, it is non-trivial whether certified everlasting garbling can be immediately constructed from certified everlasting SKE because garbling needs double-encryption. (For details, see Section 1.6.)

Moreover, a direct application of their generic compiler to FE does not work because of the following reason. If we directly apply their generic compiler to FE, we have a ciphertext consisting of classical and quantum parts. The classical part is the original FE ciphertext whose plaintext is $m \oplus r$ with random r, and the quantum part is random BB84 states whose computational basis states encode r. The decryption key of the function f consists of 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 $f(m)$, because what the receiver obtains is only $f(m \oplus r)$ and r, which cannot recover $f(m)$.

Bartusek-Khurana's results and our collusion-resistant FE, PE, and compute-and-compare obfuscation. While our certified everlasting secure bounded collusion-resistant FE (and its building block SKE, PKE, garbling, and RNCE) schemes are concurrent and independent work, our certified everlasting secure collusion-resistant FE, PE, and compute-and-compare obfuscation schemes use the certified everlasting lemma by Bartusek and Khurana(Lemma 3.5).[4] Those three schemes were added after the paper by Bartusek and Khurana was made public. *Their work does not consider FE, PE, and compute-and-compare obfuscation.*

If we directly apply their generic compiler to PE, we cannot hide the attribute part though we can hide the plaintext part. Even if we apply the same technique to the attribute part, say, we also set the attribute to $a \oplus r'$ with random r', and put random BB84 states whose computational basis states encode r' in a ciphertext, the idea does not work. A receiver cannot obtain the plaintext even if $P(a) = 1$ because the predicate computes $P(a \oplus r')$ instead of $P(a)$, and the correctness does not hold.

It is non-trivial whether we can obtain certified everlasting compute-and-compare obfuscation by their framework for encryption with certified deletion because we need to hide information about circuits while preserving the functionality. Savvy readers might think it may be possible by applying the framework to the compute-and-compare obfuscation from circular *insecure* FHE by Kluczniak [Klu22]. However, we need compute-and-compare obfuscation to instantiate circular insecure FHE. This is a circular argument.

Certified everlasting secure FE. Bartusek, Garg, Goyal, Khurana, Malavolta, Raizes, and Roberts [BGG+23] concurrently and independently obtained adaptively certified everlasting secure collusion-resistant FE for $\mathsf{P/poly}$ from IO and OWFs. They use subspace coset states [CLLZ21], while we use BB84 states (with one-time signatures). Hence, the techniques are different. Their scheme is publicly verifiable thanks to the subspace coset state approach. Another technical difference is that they directly rely on adaptively secure multi-input FE (MIFE) [GGG+14, GJO16] while we do not. Hence, their scheme incurs an additional sub-exponential loss (from IO to adaptively secure MIFE [GJO16]). Our scheme uses selectively secure MIFE and does not incur sub-exponential loss. We note that selectively secure MIFE and IO are equivalent without any security loss [GGG+14]. They also present several certified everlasting secure

[4] This is because this paper is a major update version of the paper by Hiroka et al. [HMNY22a] with new additional results (i.e., collusion-resistant FE, PE, and compute-and-compare obfuscation). The content in the work by Hiroka et al. [HMNY22a] is a concurrent and independent work of the work by Bartusek and Khurana [BK23].

primitives that are not considered in our work. However, the results on RNCE, garbled circuits, compute-and-compare obfuscation, and PE are unique to our work.

1.4 Subsequent Work

A subsequent work by Kitagawa, Nishimaki, and Yamakawa [KNY23] shows another application of our authentication technique for BB84 states which we develop for the construction of certified everlasting secure collusion-resistant FE. Specifically, they use the technique to construct a generic compiler to add the publicly verifiable deletion property for various kinds of cryptographic primitives solely based on OWFs.

1.5 Technical Overview: Collusion-Resistant FE

Certified everlasting lemma of Bartusek and Khurana. Our construction is based on a lemma which we call *certified everlasting lemma* proven by Bartusek and Khurana [BK23], which is described as follows.

Suppose that $\{\mathcal{Z}(m)\}_{m\in\{0,1\}^{\lambda+1}}$ is a family of distributions over classical strings such that $\mathcal{Z}(m)$ is computatioally indistinguishable from $\mathcal{Z}(0^{\lambda+1})$ for any $m \in \{0,1\}^{\lambda+1}$. Intuitively, $\mathcal{Z}(m)$ can be regarded as an "encryption" of m. For $b \in \{0,1\}$ and a QPT adversary, let $\widetilde{\mathcal{Z}}(b)$ be the following experiment:

- The experiment samples $z, \theta \leftarrow \{0,1\}^{\lambda}$.
- The adversary takes $|z\rangle_{\theta}$, and $\mathcal{Z}(\theta, b \oplus \bigoplus_{j:\theta_j=0} z_j)$ as input where z_j is the j-th bit of z and outputs a classical string $z' \in \{0,1\}^{\lambda}$ and a quantum state ρ.
- The experiment outputs ρ if $z'_j = z_j$ for all j such that $\theta_j = 1$ and otherwise outputs a special symbol $\perp$.

Then for any QPT adversary, the trace distance between $\widetilde{\mathcal{Z}}(0)$ and $\widetilde{\mathcal{Z}}(1)$ is $\mathsf{negl}(\lambda)$.[5]

The above lemma can be regarded as a generic compiler that adds certified everlasting security. For example, we can construct a certified everlasting PKE scheme from any plain PKE scheme as follows. For encrypting a message $b \in \{0,1\}$, a ciphertext is set to be $|z\rangle_{\theta}, \mathsf{Enc}(\theta, b \oplus \bigoplus_{j:\theta_j=0} z_j)$ where $z, \theta \leftarrow \{0,1\}^{\lambda}$ and Enc is the encryption algorithm of the underlying PKE scheme. Here, we omit an encryption key for simplicity and keep using a similar convention throughout this subsection. The deletion algorithm simply measures $|z\rangle_{\theta}$ in the Hadamard basis to output a certificate z' and the verification algorithm checks if $z'_j = z_j$ for all j such that $\theta_j = 1$. Then the above lemma implies that an adversary's internal state has no information about b conditioned on the acceptance, which means certified everlasting security.

Public-slot FE. Unfortunately, their compiler does not directly work for FE in general. The problem is that for a function f, there may not exist a function f' such that $f(m)$ can be recovered from $f'(m \oplus \bigoplus_{j:\theta_j=0} z_j, \theta)$ and z. To overcome this issue, we introduce an extension of FE which we call public-slot FE. In public-slot FE, a decryption key is associated with a *two-input* function where the first and second inputs are referred to as the secret and public inputs, respectively. Given a ciphertext of a message m and a decryption key for a function f, one can compute $f(m, \mathsf{pub})$ for all public inputs pub. Its security is defined similarly to that of plain FE except that the challenge message pair $(m^{(0)}, m^{(1)})$ must satisfy $f(m^{(0)}, \mathsf{pub}) = f(m^{(1)}, \mathsf{pub})$ for all key queries f and public inputs pub.

We observe that many existing constructions of FE based on IO (e.g., [GGH+16]) can be naturally extended to public-slot FE. In particular, we show that a simple modification of the FE scheme of Ananth and Sahai [AS16] yields an adaptively secure public-slot FE based on IO. See Appendix B for details.

[5] In fact, we need an "interactive version" of the lemma. We believe that such an interactive version is implicitly proven and used in [BK23]. See Lemma 3.7 for the formal statement of the lemma and Remark 3.8 for a comparison with [BK23].

First attempt. Our first attempt to construct a collusion-resistant FE scheme with certified everlasting security is as follows. Let Enc be an encryption algorithm of a public-slot FE scheme. A ciphertext for a message $m = m_1 \ldots m_n \in \{0,1\}^n$ consists of $\{|z_i\rangle_{\theta_i}\}_{i\in[n]}$ and $\mathsf{Enc}(\theta_1, \ldots, \theta_n, \beta_1, \ldots, \beta_n)$ where $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for $i \in [n]$, and $\beta_i := m_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ where $z_{i,j}$ is the j-th bit of z_i. A decryption key for a function f is a decryption key of the underlying public-slot FE for a two-input function $g[f]$ defined as follows. The function $g[f]$ takes a secret input $(\theta_1, \ldots, \theta_n, \beta_1, \ldots, \beta_n)$ and a public input $(b_1, \ldots, b_n) \in \{0,1\}^{\lambda\times n}$, computes $m_i := \beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j}$ for $i \in [n]$, and outputs $f(m_1, \ldots, m_n)$. To see decryption correctness, we first observe that if we first measure $\{|z_i\rangle_{\theta_i}\}_{i\in[n]}$ in the computational basis to get $(b_1, \ldots, b_n)$, then we have $b_{i,j} = z_{i,j}$ for all i, j such that $\theta_{i,j} = 0$. Thus, if we run the decryption algorithm of the public-slot FE scheme with the public input $(b_1, \ldots, b_n)$, then this yields the correct output $f(m_1, \ldots, m_n)$. We remark that the decryption can actually be done without measuring $\{|z_i\rangle_{\theta_i}\}_{i\in[n]}$ by running the above procedure coherently. The deletion and verification algorithms can be defined similarly to those for the certified everlasting PKE scheme as explained above: The deletion algorithm simply measures $\{|z_i\rangle_{\theta_i}\}_{i\in[n]}$ in the Hadamard basis to get $\{z'_i\}_{i\in[n]}$ and the verification algorithm checks if $z'_{i,j} = z_{i,j}$ for all i, j such that $\theta_{i,j} = 1$.

However, the above scheme is insecure. The problem is that public-slot FE does not force an adversary to use a legitimate public input. By running the decryption algorithm with different public inputs many times, an adversary can learn more than $f(m_1, ..., m_n)$, which would even break security as a plain FE scheme. For example, if the adversary uses a public input $(b_1, ..., b'_i, ..., b_n)$ such that b'_i is the same as b_i except that $b'_{i,j} \neq b_{i,j}$ for some j such that $\theta_{i,j} = 0$, then it can obtain $f(m_1, ..., 1 - m_i, ..., m_n)$.

Certify the public input by one-time signatures. Our idea to resolve the above issue is to certify $\{z_i\}_{i\in[n]}$ in the quantum part of the ciphertext by using one-time signatures. Specifically, the encryption algorithm first generates a pair of a verification key $\mathsf{vk}_{i,j}$ and a signing key $\mathsf{sk}_{i,j}$ of a deterministic one-time signature for $i \in [n]$ and $j \in [\lambda]$. A ciphertext for a message $m = m_1 \ldots m_n \in \{0,1\}^n$ consists of $\{|\psi_{i,j}\rangle\}_{i\in[n],j\in[\lambda]}$ and $\mathsf{Enc}(\{\mathsf{vk}_{i,j}\}_{i\in[n],j\in[\lambda]}, \theta_1, \ldots, \theta_n, \beta_1, \ldots, \beta_n)$ where $z_i, \theta_i \leftarrow \{0,1\}^n$ for $i \in [n]$, $\beta_i := m_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_j$, and

$$|\psi_{i,j}\rangle := \begin{cases} |z_{i,j}\rangle \left|\sigma_{i,j,z_{i,j}}\right\rangle & \text{if } \theta_{i,j} = 0 \\ |0\rangle |\sigma_{i,j,0}\rangle + (-1)^{z_{i,j}} |1\rangle |\sigma_{i,j,1}\rangle & \text{if } \theta_{i,j} = 1 \end{cases}$$

where $\sigma_{i,j,b}$ is a signature generated by using the signing key $\mathsf{sk}_{i,j}$ on the message $b \in \{0,1\}$. Note that $|\psi_{i,j}\rangle$ is the state obtained by coherently running the signing algorithm with the signing key $\mathsf{sk}_{i,j}$ on j-th qubit of $|z_i\rangle_{\theta_i}$. We modify the function $g[f]$ associated with the decryption key of the public-slot FE to additionally check the validity of the signatures for $b_{i,j}$ for i, j such that $\theta_{i,j} = 0$. That is, $g[f]$ takes a secret input $(\{\mathsf{vk}_{i,j}\}_{i\in[n],j\in[\lambda]}, \theta_1, \ldots, \theta_n, \beta_1, \ldots, \beta_n)$ and a public input $(b_1, \ldots, b_n, \sigma_1, \ldots, \sigma_n)$, parses $\sigma_i = (\sigma_{i,1}, \ldots, \sigma_{i,\lambda})$ for each $i \in [n]$, and checks if $\sigma_{i,j}$ is a valid signature for $b_{i,j}$ (i.e., if $\sigma_{i,j} = \sigma_{i,j,b_{i,j}}$) for all i, j such that $\theta_{i,j} = 0$. If it is not the case, it just outputs $\perp$. Otherwise, it computes $m_i := \beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j}$ for $i \in [n]$ and outputs $f(m_1, \ldots, m_n)$. Note that $|\psi_{i,j}\rangle$ contains the valid signature $\sigma_{i,j,z_{i,j}}$ on the message $z_{i,j}$ whenever $\theta_{i,j} = 0$. Thus, the decryption correctness is unaffected. In addition, if we measure $|\psi_{i,j}\rangle$ in the Hadamard basis for i, j such that $\theta_{i,j} = 1$, then the outcome $(c_{i,j}, d_{i,j})$ satisfies $z_{i,j} = c_{i,j} \oplus d_{i,j}(\sigma_{i,j,0} \oplus \sigma_{i,j,1})$. By modifying the verification algorithm to check the above equality, the verification correctness also holds. By the security of one-time signatures, an adversary cannot arbitrarily modify the public input when running the decryption algorithm of the underlying public-slot FE.

While this authentication technique seems to prevent obvious attacks, we still do not know how to prove certified everlasting security of this scheme. In particular, we want to rely on the certified everlasting lemma of [BK23]. However, the lemma only enables us to perform bit-wise game hops. For example, if $n = 3$ and the challenge messages are 000 and 111, we would need to consider hybrid experiments where the challenge message evolves as $000 \to 100 \to 110 \to 111$.[6] However, the restriction on the adversary only ensures $f(000) = f(111)$ for decryption key queries f and does not ensure, say, $f(000) = f(100)$. Without this condition, we cannot rely on the security of the

[6]Note that an FE scheme with 3-bit messages itself is trivial to construct from any PKE scheme. We are considering this toy example just to explain a technical difficulty.

underlying public-slot FE. Hence, it seems impossible to prove indistinguishability between neighboring intermediate hybrids.

Redundant encoding. Our idea for resolving the above issue is to encode the message in a redundant way so that there is a space for a "spare message". Specifically, we first encode a message $m = m_1 \dots m_n \in \{0,1\}^n$ into a $(2n+1)$-bit string $m_1 \dots m_n \| 0^{n+1}$. The rest of the scheme is identical to that in the previous paragraph, except that i's range is $[2n+1]$ instead of $[n]$ and $g[f]$ chooses which part to use for deriving the output depending on the value of the $(2n+1)$-th bit. Specifically, $g[f]$ takes a secret input $(\{\mathsf{vk}_{i,j}\}_{i\in[2n+1],j\in[\lambda]}, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1})$ and a public input $(b_1, \dots, b_{2n+1}, \sigma_1, \dots, \sigma_{2n+1})$ and first checks the validity of the signatures on positions corresponding to i, j such that $\theta_{i,j} = 0$ as before. Then it computes $m_i := \beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j}$ for $i \in [2n+1]$, and outputs $F(m_1, \dots, m_{2n+1})$ where F is defined as

$$F(m_1, \dots, m_{2n+1}) := \begin{cases} f(m_1, \dots, m_n) & \text{if } m_{2n+1} = 0 \\ f(m_{n+1}, \dots, m_{2n}) & \text{if } m_{2n+1} = 1 \end{cases}.$$

The decryption correctness is unaffected because we always have $m_{2n+1} = 0$ when decrypting an honestly generated message. The verification correctness is also unaffected since the way of encoding messages is irrelevant. We explain why this enables us to avoid the issue mentioned in the previous paragraph. Intuitively, the advantage of such a redundant encoding is that we can ensure that the encoded challenge message contains either of two challenge messages in all intermediate hybrids. Let $m^{(0)}$ and $m^{(1)}$ be a pair of challenge messages. Note that they correspond to $m^{(0)} \| 0^{n+1}$ and $m^{(1)} \| 0^{n+1}$ after encoding. Then we consider intermediate hybrids where the corresponding challenge messages after the encoding evolves as follows:

1. Starting from $m^{(0)} \| 0^{n+1}$, we change the $(n+1)$-th to $2n$-th bits one-by-one toward $m^{(0)} \| m^{(1)} \| 0$.
2. Flip the $(2n+1)$-th bit, which results in $m^{(0)} \| m^{(1)} \| 1$.
3. Change the first to n bits one-by-one toward $m^{(1)} \| m^{(1)} \| 1$.
4. Flip the $(2n+1)$-th bit, which results in $m^{(1)} \| m^{(1)} \| 0$.
5. Change the $(n+1)$-th to $2n$-th bits one-by-one toward $m^{(1)} \| 0^{2n+1}$.

Importantly, the value of F on the encoded challenge message is equal to $f(m^{(0)}) = f(m^{(1)})$ at any point of the hybrids. This enables us to rely on the security of the underlying public-slot FE along with certified everlasting lemma in every hybrid.

What one-time signatures to use? Finally, we remark that we have to choose an instantiation of one-time signatures carefully. Roughly speaking, the reason why we are using one-time signatures is to prevent an adversary from using "unauthorized" $b_{i,j}$, i.e., those for which the valid signature $\sigma_{i,j,b_{i,j}}$ is not given to the adversary. However, by the correctness of one-time signatures, a valid signature must exist on every message. This means that a valid signature on an "unauthorized" $b_{i,j}$ must exist even if it is difficult for an adversary to find. This situation is not compatible with the security definition of public-slot FE. Recall that its security requires that the challenge message pair $(m^{(0)}, m^{(1)})$ must satisfy $f(m^{(0)}, \mathsf{pub}) = f(m^{(1)}, \mathsf{pub})$ for all key queries f and *all* public inputs pub. That is, the security is not applicable if there is at least one pub such that $f(m^{(0)}, \mathsf{pub}) \neq f(m^{(1)}, \mathsf{pub})$ even if such pub is difficult to find. To overcome this issue, we use Lamport signatures instantiated with a PRG. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG. When the message length is 1, a signing key is set to be $(u_0, u_1) \in \{0,1\}^{\lambda\times 2}$ and a verification key is set to be $(v_0 = \mathsf{PRG}(u_0), v_1 = \mathsf{PRG}(u_1)) \in \{0,1\}^{2\lambda\times 2}$. A signature for a bit b is defined to be u_b. This scheme has a special property in that we can program a verification key so that it does not have a valid signature for a particular message. For example, if we want to ensure that a message 0 does not have a valid signature, then we can set v_0 to be a uniformly random 2λ-bit string. Then, with probability $1 - 2^{-\lambda}$, there is no preimage of v_0, which means that there is no valid signature on the massage 0. By using this property, whenever $b_{i,j}$ is unauthorized, we can switch to a hybrid where there is no valid signature for $b_{i,j}$. This effectively resolves the above issue.

1.6 Technical Overview: Bounded Collusion-Resistant FE

In this subsection, we give a high-level overview of our certified everlasting secure bounded collusion-resistant FE schemes. It is known that the (bounded collusion-resistant) plain FE is constructed from (standard) PKE, RNCE, and garbling [GVW12]. A natural strategy is constructing PKE, RNCE, and garbling with certified everlasting security and using them as building blocks. We show that PKE with certified everlasting security can be constructed using the techniques of [Unr15, HMNY22b]. RNCE with certified everlasting security for *classical messages* can be constructed from certified everlasting PKE in the same way as standard RNCE [KNTY19]. However, such an RNCE scheme is insufficient for our purpose (constructing adaptively-secure FE) because it is not for *quantum messages*. We also need a new idea to construct garbling with certified everlasting security. The following explains these ideas and how to construct FE with certified everlasting security.

Certified everlasting garbling for $\mathsf{P/poly}$ circuits. In classical cryptography, it is known that we can construct plain garbling from plain SKE using double-encryption [Yao86, LP09]. Double-encryption means we generate a nested ciphertext $\mathsf{ct}_2 \leftarrow \mathsf{Enc}(\mathsf{sk}', \mathsf{ct}_1)$, where $\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, m)$, m is the message, Enc is the encryption algorithm of SKE, and $\mathsf{sk}, \mathsf{sk}'$ are secret keys of SKE. This double-encryption is an essential technique for garbling. However, it is an obstacle to our purpose. First, we do not know SKE with certified everlasting security for *quantum* messages. Second, even if the first problem is solved, we have another problem: We can obtain a valid certificate showing that ct_1 has been deleted by running the deletion algorithm on ct_2. However, such a certificate does not necessarily mean the deletion of m. We bypass the problem using XOR secret sharing instead of double-encryption.[7] More precisely, we uniformly randomly sample p and compute $(\mathsf{vk}', \mathsf{ct}') \leftarrow \mathsf{Enc}(\mathsf{sk}', p)$ and $(\mathsf{vk}, \mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{sk}, p \oplus m)$ to encrypt message m. Here, Enc is the encryption algorithm of certified everlasting SKE, and $\mathsf{vk}', \mathsf{vk}$ are the verification keys that are used to verify the correctness of deletion certificates. The receiver with $(\mathsf{ct}', \mathsf{ct})$ can obtain m only if it has both sk' and sk, and nothing else otherwise, as in the case of double-encryption. Furthermore, once the receiver issues the deletion certificate of $(\mathsf{ct}', \mathsf{ct})$, it can no longer obtain the information of m even if it becomes computationally unbounded.

It is easy to see that we can implement the well-known gate garbling [Yao86, LP09] by using the double encryption in the parallel way above instead of the sequential double encryption. We can prove its computational security via a similar discussion as that in [LP09]. (Although [LP09] uses double-encryption, we can show the security for the XOR secret sharing case similarly.) Furthermore, we can prove its certified everlasting security by using the certified everlasting security of the SKE. Hence, we can obtain certified everlasting garbling. The formal construction of our certified everlasting garbling is given in Appendix E.2. For details, see that section.

FE with non-adaptive security. Our next task is achieving certified everlasting FE using certified everlasting garbling. It is known that plain FE with *non-adaptive security* can be constructed by running the encryption algorithm of (plain) PKE on labels of a plain garbling scheme [SS10].[8] In our certified everlasting garbling scheme (explained in the previous paragraph), the labels are classical bit strings and the deletion algorithm does not take the labels as input. Therefore, this classical construction for plain FE by Sahai and Seyalioglu [SS10] can be directly applied to the construction of our 1-bounded certified everlasting FE for $\mathsf{P/poly}$ circuits with *non-adaptive security*. (The formal construction is given in Section 4.2.)

FE with adaptive security. Now, we want to convert non-adaptive security to adaptive one.[9] However, the conversion is non-trivial. Let us first review the conversion for plain FE. In classical cryptography, we can convert non-adaptively secure FE into adaptively secure FE by using RNCE. Roughly speaking, RNCE is the same as PKE except that we can generate a fake ciphertext $\widetilde{\mathsf{ct}} \leftarrow \mathsf{Fake}(\mathsf{pk})$ without plaintext and we can generate a fake secret key $\widetilde{\mathsf{sk}} \leftarrow \mathsf{Reveal}(\mathsf{pk}, m)$ that decrypts $\widetilde{\mathsf{ct}}$ to m. The security of RNCE guarantees that $(\mathsf{Enc}(\mathsf{pk}, m), \mathsf{sk})$ and $(\mathsf{Fake}(\mathsf{pk}), \mathsf{Reveal}(\mathsf{pk}, m))$ are computationally indistinguishable, where Enc is the real encryption algorithm, and sk is the real secret key. Adaptively secure FE can be constructed by running the real encryption algorithm Enc of the RNCE on the ciphertext $\mathsf{nad.ct}$ of the FE. We can prove its adaptive security as follows. The adversary of adaptive security can send key queries after

[7] A similar technique was used by Gentry, Halevi, and Vaikuntanathan [GHV10].

[8] The non-adaptive security means that the adversary can call the key queries only before the challenge encryption query.

[9] The adaptive security means that the adversary can call key queries before and after the challenge encryption query.

the challenge encryption query. However, the sender can simulate the challenge encryption query without generating $\mathsf{nad.ct}$. This is because, from the security of RNCE, we can switch $(\mathsf{Enc}(\mathsf{pk},\mathsf{nad.ct}),(\mathsf{sk},\mathsf{nad.sk}_f))$ to the fake one $(\mathsf{Fake}(\mathsf{pk}),(\mathsf{Reveal}(\mathsf{pk},\mathsf{nad.ct}),\mathsf{nad.sk}_f))$, where $\mathsf{nad.sk}_f$ is the functional secret key of the non-adaptively secure FE. Therefore, the sender needs not generate $\mathsf{nad.ct}$ before generating $\mathsf{nad.sk}_f$ for the simulation of the adversary's queries, which means that we can reduce the adaptive security to the non-adaptive security.

How can we adopt the above classical idea of the conversion to the certified everlasting case? From the discussion above, a straightforward way is to encrypt the ciphertext $\mathsf{nad.ct}$ of certified everlasting FE with non-adaptive security using certified everlasting RNCE as follows: $(\mathsf{vk},\mathsf{ct}) \leftarrow \mathsf{Enc}(\mathsf{pk},\mathsf{nad.ct})$, where vk is the verification key, pk is the public key, and Enc is the real encryption algorithm of the certified everlasting RNCE. However, this idea fails for the following two reasons. First, $\mathsf{nad.ct}$ is a quantum state. Our certified everlasting RNCE scheme does not support quantum messages. Second, even if we can construct RNCE for quantum messages, we have another problem: A valid certificate of ct is issued by running the deletion algorithm on ct. However, such a certificate does not necessarily mean the deletion of the plaintext of $\mathsf{nad.ct}$. The first problem is about security, and the second problem is about correctness.

Our idea to resolve the first problem is to use quantum teleportation. We construct RNCE for quantum messages from RNCE for classical messages by using quantum teleportation.[10] (We believe that the idea of using quantum teleportation in the following way will be useful in many other applications beyond RNCE.) As the ciphertext and the secret key of adaptively secure FE, we take

$$\frac{1}{2^{2N}}\sum_{a,b\in\{0,1\}^N}(Z^bX^a(\mathsf{nad.ct})X^aZ^b)_{C1}\otimes\mathsf{Enc}(\mathsf{pk},(a,b))_{C2}\otimes(\mathsf{nad.sk}_f,\mathsf{sk})_S,$$

where $\mathsf{nad.ct}$ is an N-qubit state, the registers C_1 and C_2 are the ciphertext, and the register S is the secret key. Here, $X^a := \bigotimes_{j=1}^N X_j^{a_j}$, $Z^b := \bigotimes_{j=1}^N Z_j^{b_j}$, a_j is the jth bit of a, and b_j is the jth bit of b. Moreover, Enc is the real encryption algorithm of RNCE for classical messages, $\mathsf{nad.sk}_f$ is the secret key of non-adaptively secure FE, and sk is the real secret key of RNCE for classical messages. We want to show the adaptive security of the construction by reducing it to the non-adaptive security of the building block FE. In the first step of hybrids, we switch the state to

$$\frac{1}{2^{2N}}\sum_{a,b}(Z^bX^a(\mathsf{nad.ct})X^aZ^b)_{C1}\otimes\mathsf{Fake}(\mathsf{pk})_{C2}\otimes(\mathsf{nad.sk}_f,\mathsf{Reveal}(\mathsf{pk},(a,b)))_S$$

by using the property of RNCE for classical messages. In the second step of hybrids, we switch the state to

$$\frac{1}{2^{2N}}\sum_{x,z\in\{0,1\}^N}\mathcal{T}^{x,z}_{A',A}[\mathsf{nad.ct}_{A'}\otimes|\Phi_N\rangle\langle\Phi_N|_{A,C1}]\otimes\mathsf{Fake}(\mathsf{pk})_{C2}\otimes(\mathsf{nad.sk}_f,\mathsf{Reveal}(\mathsf{pk},(x,z)))_S,$$

where $|\Phi_N\rangle$ is the N Bell pairs between the registers A and C_1. $\mathcal{T}^{x,z}_{A',A}[\mathsf{nad.ct}_{A'}\otimes|\Phi_N\rangle\langle\Phi_N|_{A,C1}]$ is the state on the register C_1 obtained in the following way: the state $\mathsf{nad.ct}_{A'}$ on the register A' is coupled with the halves of N Bell pairs on the register A, and the teleportation measurement $\mathcal{T}^{x,z}_{A',A}$ with the result (x,z) is applied on the registers A and A'. Now, we can generate the states on the registers C_1 and C_2 without knowing $\mathsf{nad.ct}$, which means that the sender can simulate the challenge encryption query without $\mathsf{nad.ct}$. In other words, the sender does not need to generate $\mathsf{nad.ct}$ before generating $\mathsf{nad.sk}_f$ for the simulation of adversary queries.

This idea solves the first problem. However, the second problem remains. The receiver with $(Z^bX^a(\mathsf{nad.ct})X^aZ^b,$ $\mathsf{Enc}(\mathsf{pk},(a,b)))$ can issue a deletion certificate of $Z^bX^a(\mathsf{nad.ct})X^aZ^b$. The deletion certificate does not necessarily pass the verification algorithm for the deletion of $\mathsf{nad.ct}$. This is an obstacle to achieving correctness. We solve this problem by introducing an efficient algorithm that we call the modification algorithm. Let $\mathsf{nad.cert}^*$ be the deletion certificate of $Z^bX^a(\mathsf{nad.ct})X^aZ^b$. The modification algorithm takes (a,b) and $\mathsf{nad.cert}^*$ as input, and outputs $\mathsf{nad.cert}$ that is the deletion certificate of $\mathsf{nad.ct}$. Therefore, by using the modification algorithm, we can convert the deletion certificate $\mathsf{nad.cert}^*$ of $Z^bX^a(\mathsf{nad.ct})X^aZ^b$ to the deletion certificate $\mathsf{nad.cert}$ of $\mathsf{nad.ct}$. We observe that the modification algorithm exists for many natural constructions, including our construction.[11]

The formal explanation of the conversion from non-adaptive to adaptive FE is given in Section 4.3.

[10] A similar technique was used in the context of multi-party quantum computation [BCKM21].

[11] If the deletion algorithm is the computational-basis measurements followed by Clifford gates, the modification algorithm is just modifying the Pauli one-time pad, X^aZ^b. In fact, all known constructions use only Hadamard basis measurements.

q-bounded FE for NC^1 circuits. Finally, we explain how to convert 1-bounded one to the q-bounded one.[12] Unfortunately, we do not know how to obtain q-bounded certified everlasting FE for P/poly circuits. What we can construct in this paper is that only for NC^1 circuits. (It is an open problem to obtain q-bounded certified everlasting FE for P/poly circuits. For more details, see Section 4.4.)

Let us explain how to convert 1-bounded certified everlasting FE for P/poly circuits to q-bounded certified everlasting FE for NC^1 circuits. In classical cryptography, it is known that [GVW12] multi-party computation (MPC) can convert plain 1-bounded FE for P/poly circuits to plain q-bounded FE for NC^1 circuits. The idea is, roughly speaking, the view of each party in the MPC protocol is encrypted using 1-bounded FE scheme. In this classical construction, no encryption is done on the ciphertexts of plain FE, and therefore this classical construction can be directly applied to our certified everlasting case. (It is an open problem to obtain q-bounded certified everlasting FE for P/poly circuits. The formal construction is given in Section 4.4.

1.7 Technical Overview: Compute-and-Compare Obfuscation

This section provides a high-level overview of our certified everlasting compute-and-compare obfuscation. Recall that a compute-and-compare obfuscation scheme obfuscates a circuit P along with a lock value lock and a message m and outputs an obfuscated circuit $\widetilde{P}$. In the evaluation phase, one can recover m from $\widetilde{P}$ using an input x to the circuit such that $P(x) = \mathsf{lock}$. A certified everlasting compute-and-compare obfuscation scheme additionally generates a verification key vk while obfuscating circuit P. A user can generate a deletion certificate cert from $\widetilde{P}$. If we have vk, we can verify whether the certificate is valid or not. The certified everlasting security ensures that no information about P, lock and m is available to the user after producing a valid certificate. This means that the user actually deleted the obfuscated circuit.

Compute-and-compare obfuscation without a message. We first explain our idea to construct a certified everlasting compute-and-compare obfuscation without any message. That is, the evaluation returns 1 if $P(x) = \mathsf{lock}$ holds. Let $\mathsf{CC.Obf}$ be the obfuscation algorithm of a standard compute-and-compare obfuscation scheme and $\mathsf{Enc}, \mathsf{Dec}$ be the encryption, and decryption algorithms of FHE. The main idea is to compute an FHE ciphertext ct_P encrypting the circuit P and use $\mathsf{CC.Obf}$ to produce an obfuscated circuit $\widetilde{\mathsf{Dec}}$ of the decryption circuit of FHE with lock value lock and message 1. The obfuscated circuit $\widetilde{P}$ consists of ct_P and $\widetilde{\mathsf{Dec}}$. Given an input x, we first apply the evaluation procedure of FHE to get a ciphertext $\mathsf{ct}_{P(x)} = \mathsf{Enc}(P(x))$ (we omit the encryption key) and then run the evaluation algorithm of the compute-and-compare obfuscation with input $\mathsf{ct}_{P(x)}$ to check whether $P(x) = \mathsf{lock}$. Note that we cannot use certified everlasting FHE [BK23] in a black-box manner since $\mathsf{CC.Obf}$ is a classical algorithm that cannot obfuscate a quantum circuit, in particular, the decryption algorithm of the FHE. Instead, we use BB84 states along with classical FHE as follows. The obfuscated circuit $\widetilde{\mathcal{P}}$ consists of $\widetilde{\mathsf{Dec}} := \mathsf{CC.Obf}(\mathsf{Dec}(\mathsf{sk}, \cdot), \mathsf{lock}, 1)$ and $\{|z_i\rangle_{\theta_i}, \mathsf{ct}_i\}_{i \in [\ell_P]}$ where $\mathsf{ct}_i := \mathsf{Enc}(\theta_i \| \widetilde{b}_i)$, $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for $i \in [\ell_P]$, $\widetilde{b}_i := b_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ and b_i is the i-th bit of the binary string of length ℓ_P representing the circuit P. The verification key is $\mathsf{vk} = (\{z_i, \theta_i\}_{i \in [\ell_P]})$. To evaluate the obfuscated circuit with an input x, we first coherently compute an evaluated FHE ciphertext $\left|\mathsf{ct}_{U_x(P)}\right\rangle$ where U_x is a circuit that on input $(\{z_i, \theta_i, \widetilde{b}_i\}_{i \in [\ell_P]})$ first recovers b_i, the bits representing P, and then outputs $P(x)$. Then, we coherently evaluate the obfuscated circuit $\widetilde{\mathsf{Dec}}$ with input $\left|\mathsf{ct}_{U_x(P)}\right\rangle$ and check that the measured outcome is 1 to decide $P(x) = \mathsf{lock}$. The deletion and verification algorithm works similarly as in the certified everlasting PKE scheme described in Section 1.5. That is, we use the concrete certified everlasting secure FHE scheme by Bartusek and Khurana in a non-black-box way.

However, the above scheme cannot guarantee certified everlasting security. The reason is that the classical compute-and-compare obfuscation cannot hide the lock value from an unbounded adversary. More precisely, the unbounded adversary is given a target circuit and an auxiliary input and can easily distinguish between the obfuscated circuit $\widetilde{\mathsf{Dec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{Dec}, \mathsf{lock}, 1)$ and the corresponding simulated circuit $\widetilde{\mathsf{Dec}} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_{\mathsf{Dec}}, 1^1)$ if the auxiliary input and lock are correlated, where $\mathsf{pp}_{\mathsf{Dec}}$ consists of parameters of Dec (input and output length and circuit size).

[12] q-bounded means that the adversary can call key queries q times with an a priori bounded polynomial q.

We solve this problem by masking the obfuscated circuit that encodes lock using the XOR function in combination with the BB84 states. In particular, we sample "dummy" lock value $R \leftarrow \{0,1\}^\lambda$ and set the obfuscated circuit $\mathcal{L}_C$ as $(\widetilde{\mathsf{Dec}} := \mathsf{CC.Obf}(\mathsf{Dec}(\mathsf{sk},\cdot), R, 1), \{|z_i\rangle_{\theta_i}, \mathsf{ct}_i\}_{i\in[\ell]})$ where $\ell = \ell_P + \ell_{\widetilde{I}}$ and $\{\mathsf{ct}_i\}_{i\in[\ell]}$ encrypts the binary string representing the circuits $(P\|\widetilde{I})$ where $\widetilde{I} := \mathsf{CC.Obf}(I, \mathsf{lock}, R)$. We denote I by the identity circuit that is $I(x) = x$ for all x. The evaluation algorithm works as before except that the circuit U_x on input $(\{z_i, \theta_i, \widetilde{b}_i\}_{i\in[\ell]})$ first reconstructs $(P\|\widetilde{I})$ and then outputs the result obtained in the evaluation of $\widetilde{I}$ with input $P(x)$. Hence, checking $P(x) = \mathsf{lock}$ is deferred until evaluating $\widetilde{I}$, which is hidden due to the certified everlasting security of FHE. The correctness follows from the fact that U_x returns R if $P(x) = \mathsf{lock}$ and evaluation of $\widetilde{\mathsf{Dec}}$ outputs 1 if $U_x(P\|\widetilde{I}) = R$.

The simulated circuit $\widetilde{\mathcal{P}}$ consists of $\widetilde{\mathsf{Dec}} = \mathsf{CC.Obf}(\mathsf{Dec}(\mathsf{sk},\cdot), R, 1)$ and $\{|z_i\rangle_{\theta_i}, \mathsf{ct}_i\}_{i\in[\ell]}$ where $\mathsf{ct}_i := \mathsf{Enc}(\theta_i\|\widetilde{b}_i)$ and $\widetilde{b}_i := 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for $i \in [\ell]$. Note that, $\widetilde{\mathcal{P}}$ does not contain any information about P and lock. We rely on the certified everlasting lemma of [BK23] to show that the real obfuscated circuit is indistinguishable from the simulated circuit for any unbounded adversary who produces a valid certificate of deletion. Although an unbounded adversary can recover sk from $\widetilde{\mathsf{Dec}}$, sk is useless for distinguishing after the deletion. Since the lemma only allows us to flip one bit at a time, we use a sequence of ℓ hybrid experiments. In the i-th hybrid, we change the bit b_i from 1 to 0. If we can show that $\mathsf{Enc}(\theta_i\|\widetilde{b}_i)$ is computationally indistinguishable from $\mathsf{Enc}(\mathbf{0}\|\widetilde{b}_i)$ and then it is possible to apply the certified everlasting lemma to flip the bit b_i without noticing the unbounded adversary. To establish the computational indistinguishability, we first replace the circuit $\widetilde{I} \leftarrow \mathsf{CC.Obf}(I, \mathsf{lock}, R)$ with the simulated one $\widetilde{I} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_I, 1^{|R|})$ and then change the circuit $\widetilde{\mathsf{Dec}} \leftarrow \mathsf{CC.Obf}(\mathsf{Dec}, \mathsf{lock}, 1)$ to the corresponding simulated circuit $\widetilde{\mathsf{Dec}} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_{\mathsf{Dec}}, 1^1)$ depending on the security of the underlying compute-and-compare obfuscation scheme. Since the FHE secret key sk is no longer required to simulate the adversary's view, we can change $\mathsf{Enc}(\theta_i\|\widetilde{b}_i)$ to $\mathsf{Enc}(\mathbf{0}\|\widetilde{b}_i)$ using the IND-CPA security of FHE. Hence, b_i can be set to 0 by employing the certified everlasting lemma.

Compute-and-compare obfuscation with a message. Next, we discuss extending the above construction into a certified everlasting compute-and-compare obfuscation scheme that obfuscates a circuit P along with lock and a message $m = m_1 \ldots m_n \in \{0,1\}^n$. Our idea is to encrypt the message using FHE in combination with the BB84 states and recover the message bits during evaluation depending on the outcome of the obfuscated circuit $\widetilde{\mathsf{Dec}}$. The obfuscated circuit $\widetilde{\mathcal{P}}$ now additionally includes $\{|z_{\ell+k}\rangle_{\theta_{\ell+k}}, \mathsf{ct}_{\ell+k}\}_{k\in[n]}$ where $z_{\ell+k}, \theta_{\ell+k} \leftarrow \{0,1\}^\lambda$, $\mathsf{ct}_{\ell+k} := \mathsf{Enc}(\theta_{\ell+k}\|\widetilde{b}_{\ell+k})$ and $\widetilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for $k \in [n]$. The evaluation procedure works as before except the U_x on input $((\{z_i, \theta_i, \widetilde{b}_i\}_{i\in[\ell]}), (z_{\ell+k}, \theta_{\ell+k}, \widetilde{b}_{\ell+k}))$ first reconstructs $(P\|\widetilde{I})$ from $\{z_i, \theta_i, \widetilde{b}_i\}_{i\in[\ell]}$ and m_k from $(z_{\ell+k}, \theta_{\ell+k}, \widetilde{b}_{\ell+k})$, and then outputs $m_k \cdot \widetilde{I}(P(x))$. We can similarly define the deletion and verification algorithms as before. The scheme correctly recovers m in a bit-by-bit manner. Let us consider $P(x) = \mathsf{lock}$ and $m_k = 1$. Then, by the definition of U_x and the correctness of compute-and-compare obfuscation, we have $m_k \cdot \widetilde{I}(P(x)) = R$. Consequently, $\widetilde{\mathsf{Dec}}$ evaluates to 1 for an input $\mathsf{ct}_{m_k\cdot\widetilde{I}(P(x))}$. If the result of the evaluation is not 1, then we set $m_k := 0$. We prove the certified everlasting security of the scheme using the same idea as discussed for the compute-and-compare obfuscation scheme without a message. The only difference is that we additionally delete the information of m using the IND-CPA security of FHE and certified everlasting lemma of [BK23] after we erase the information about P and lock. The formal construction and its security analysis are provided in Section 5.

Certified everlasting predicate encryption. Goyal, Koppula and Waters [GKW17] and Wichs and Zirdelis [WZ17] showed a generic construction of PE[13] from compute-and-compare obfuscation and ABE. The construction works as follows. The setup and key generation algorithms are the same as the underlying ABE. Let Enc and Dec be the encryption and decryption algorithms of ABE. To encrypt a message m with attribute x, the encryption algorithm samples a random lock $R \in \{0,1\}^\ell$ and computes $\mathsf{ct} := \mathsf{Enc}(x, R)$ and $\widetilde{\mathsf{Dec}} := \mathsf{CC.Obf}(\mathsf{Dec}(\cdot, \mathsf{ct}), R, m)$. The ciphertext is the obfuscated circuit $\widetilde{\mathsf{Dec}}$. Given a secret key sk_P for a policy P, a user simply evaluates $\widetilde{\mathsf{Dec}}$ with input sk_P to recover the message m. Note that, if $P(x) = 1$ then by the correctness of ABE, $\mathsf{Dec}(\mathsf{sk}_P, \mathsf{ct}) = R$ and hence $\widetilde{\mathsf{Dec}}(\mathsf{sk}_P)$ returns m.

[13]It satisfies one-sided attribute-hiding security, meaning that the attribute and message are both hidden to a user who does not have a secret key for successful decryption.

One might hope that replacing the compute-and-compare obfuscation and ABE with their certified everlasting counterparts in the classical construction yields a certified everlasting PE. This would not work since our certified everlasting compute-and-compare obfuscation cannot obfuscate a quantum decryption circuit $\mathcal{Dec}(\cdot, ct)$ of the certified everlasting ABE. However, we need to erase the information about R from the ABE ciphertext ct in order to apply the certified everlasting security of the compute-and-compare obfuscation. In other words, after a valid certificate of deletion is produced, an unbounded adversary should not be able to distinguish between $\mathsf{Enc}(x, R)$ and $\mathsf{Enc}(x, \mathbf{0})$. A classical ABE alone can not guarantee such indistinguishability. We solve this problem by using a classical ABE coupled with BB84 states and a certified everlasting compute-and-compare obfuscation in the above construction. In particular, we first sample $z_i, \theta_i \leftarrow \{0,1\}^\ell$, set $\widetilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ and then compute $\mathsf{ct} := \mathsf{Enc}(x, (\theta_1, \ldots, \theta_\ell, \widetilde{r}_i, \ldots, \widetilde{r}_\ell))$ where r_i denotes the i-th bit of R. The ciphertext consists of $\widetilde{\mathcal{Dec}} := \mathcal{CCObf}(\mathsf{Dec}(\cdot, \mathsf{ct}), R, m)$ and $\{|z_i\rangle_{\theta_i}\}_{i\in[\ell]}$. The verification key associated with the ciphertext includes $\{z_i, \theta_i\}_{i\in[\ell]}$ and a verification key $\mathsf{vk}_{\mathsf{Dec}}$ corresponding to the circuit Dec. The deletion and verification algorithms can be defined in a natural way. That is, we use the concrete certified everlasting secure ABE scheme by Bartusek and Khurana in a non-black-box way.

Suppose an adversary queries secret keys sk_P such that $P(x) = 0$ and becomes unbounded after delivering a valid certificate of deletion of the ciphertext. Our idea is to use the security of ABE and the certified everlasting lemma of [BK23] to delete the information of R. Then, we utilize the certified everlasting security of compute-and-compare obfuscation for replacing $\widetilde{\mathcal{Dec}}$ with a simulated circuit that does not contain any information about m, x. The formal security analysis can be found in Section 6.2.

1.8 More on Related Works

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

Broadbent and Islam [BI20] constructed one-time SKE with certified deletion. It is standard one-time SKE except that once the receiver issues a valid classical certificate, the receiver cannot obtain the plaintext information even if the receiver later becomes a computationally *unbounded* adversary. (See also [KT20].)

Hiroka, Morimae, Nishimaki, and Yamakawa [HMNY21] constructed reusable SKE, PKE, and ABE with certified deletion. These reusable SKE, PKE, and ABE with certified deletion are standard reusable SKE, PKE, and ABE with additional properties, respectively. Once the receiver issues a valid classical certificate, the receiver cannot obtain the plaintext information even if it obtains some secret information (e.g., the master secret key of ABE). In these primitives, the security holds against computationally bounded adversaries, unlike in this work. Poremba [Por23] achieved FHE with certified deletion. In addition, certificates for deletion are publicly verifiable in his construction. The security holds against computationally bounded adversaries, unlike in this work. However, the security of the construction relies on a strong conjecture that a particular hash function is "strong Gaussian-collapsing".

Hiroka, Morimae, Nishimaki, and Yamakawa [HMNY22b] constructed commitments with statistical binding and certified everlasting hiding. From it, they also constructed a certified everlasting zero-knowledge proof system for QMA based on the zero-knowledge protocol of [BG20].

Key certified deletion. Kitagawa and Nishimaki [KN22] introduced the notion of FE with secure key leasing, where functional decryption keys are quantum states and we can generate certificates for deleting the keys. This can be seen as certified deletion of keys and the dual of certified deletion of ciphertexts. They achieved bounded collusion-resistant secret-key FE with secure key leasing for P/poly from standard SKE.

Secure software leasing. Ananth and La Place introduced the notion of secure software leasing and achieved it for a sub-class of evasive functions from public-key quantum money (need IO and OWFs) and the LWE assumption [AL21]. Secure software leasing encode classical program into quantum program and has an explicit returning process. After a lessor verifies that a returned quantum program is valid, a lessee cannot run the leased program anymore. Later, several secure software leasing schemes for a sub-class of evasive functions or cryptographic functionalities (or its variant) with various properties (such as classical communication, without assumptions) were presented [CMP20, BJL+21, KNY21, ALL+21]. None of them are certified everlasting secure.

Compute-and-compare obfuscation, PE, and FE. There are tremendous amount of previous works on standard FE and PE for general circuits and standard compute-and-compare obfuscation. We focus on strongly related works. No previous work consider certified everlasting secure FE, PE, and compute-and-compare obfuscation.

Gorbunov, Vaikuntanathan, and Wee [GVW12] constructed bounded collusion-resistant adaptively secure PKFE for $\mathsf{P/poly}$ from standard PKE (and either the DDH or LWE assumption). Later, Ananth and Vaikuntanathan improved ciphertext size and assumptions. They constructed adaptively secure bounded collusion-resistant PKFE for $\mathsf{P/poly}$ with optimal ciphertext size from standard PKE. Garg, Gentry, Halevi, Raykova, and Sahai [GGH+16] constructed selectively secure collusion-resistant PKFE for $\mathsf{P/poly}$ from IO and OWFs. Waters [Wat15] constructed adaptively secure PKFE collusion-resistant for $\mathsf{P/poly}$ from IO and OWFs. Ananth, Brakerski, Segev, and Vaikuntanathan [ABSV15] presented a transformation from selectively secure collusion-resistant FE for $\mathsf{P/poly}$ to adaptively secure collusion-resistant FE for $\mathsf{P/poly}$. Jain, Lin, and Sahai constructed IO for $\mathsf{P/poly}$ from well-founded assumptions [JLS21, JLS22]. However, their constructions are not post-quantum secure.[14]

Gorbunov, Vaikuntanathan, and Wee [GVW15b] constructed PE for $\mathsf{P/poly}$ from the LWE assumption. Goyal, Koppula, and Waters [GKW17] and Wichs and Zirdelis [WZ17] presented the notion of compute-and-compare obfuscation (or lockable obfuscation) and achieved it from the LWE assumption. These two works also presented a general transformation from ABE to PE using compute-and-compare obfuscation. Kluczniak [Klu22] constructed compute-and-compare obfuscation from circular *insecure* FHE. However, all known instantiations of circular insecure FHE rely on compute-and-compare obfuscation.

Organization. In Section 2, we define the notation and preliminaries that we require in this work. In Section 3, we define the notion of certified everlasting secure collusion-resistant FE and provide a construction. In Section 4, we define the notion of certified everlasting secure bounded collusion-resistant FE and provide constructions. In Section 5, we define the notion of certified everlasting secure compute-and-compare obfuscation and provide a construction. In Section 6, we define the notion of certified everlasting secure PE and provide a construction.

In Appendix B, we provide a construction of adaptively secure public-slot FE, which is a building block of the construction in Section 3. In Appendix C, we define the notion of certified everlasting secure SKE and PKE and provide constructions, which are building blocks of the constructions in Section 4 and Appendices D and E. In Appendix D, we define the notion of certified everlasting secure RNCE and provide a construction, which is a building block of the construction in Section 4. In Appendix E, we define the notion of certified everlasting secure garbling schemes and provide a construction, which is a building block of the construction in Section 4.

2 Preliminaries

2.1 Notations

Here we introduce basic notations we will use in this paper.

In this paper, standard math or sans serif font stands for classical algorithms (e.g., C or Gen) and classical variables (e.g., x or pk). Calligraphic font stands for quantum algorithms (e.g., $\mathcal{G}en$) and calligraphic font and/or the bracket notation for (mixed) quantum states (e.g., q or $|\psi\rangle$).

Let $x \leftarrow X$ denote selecting an element x from a finite set X uniformly at random, and $y \leftarrow A(x)$ denote 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 $[n] := \{1,\dots,n\}$. 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 $f : \mathbb{N} \rightarrow \mathbb{R}$ is a negligible function if for any

[14] There are a few candidate constructions of post-quantum secure IO [BGMZ18, CHVW19, AP20].

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 $\epsilon \leq \mathsf{negl}(\lambda)$, 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 SKE and PKE with certified everlasting deletion in Appendices C.2 and C.4, 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 4.3 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 $X^xZ^z\rho Z^zX^x$.*

CSS code. We explain basics of CSS codes. CSS codes are used only in the constructions of SKE and PKE with certified everlasting deletion (Appendix C.3 and Appendix C.5), 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 [15] 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 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, $(x \bmod C) \bmod C = x \bmod C$ 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 $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

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

Pseudorandom generators.

Definition 2.4 (Pseudorandom Generator). *A pseudorandom generator (PRG)* $\mathsf{PRG} : \{0,1\}^{\lambda} \to \{0,1\}^{\lambda+\ell(\lambda)}$ *with stretch* $\ell(\lambda)$ *(*ℓ *is some polynomial function) is a polynomial-time computable function that satisfies the following. For any QPT adversary* $\mathcal{A}$*, it holds that*

$$\left|\Pr[\mathcal{A}(\mathsf{PRG}(s)) = 1 \mid s \leftarrow \mathcal{U}_\lambda] - \Pr\Big[\mathcal{A}(r) \mid r \leftarrow \mathcal{U}_{\lambda+\ell(\lambda)}\Big]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{U}_m$ *denotes the uniform distribution over* $\{0,1\}^m$*.*

Theorem 2.5 ([HILL99]). *If there exists a OWF, there exists a PRG.*

Pseudorandom Functions.

Definition 2.6 (Pseudorandom Function). *Let* $\{\mathsf{F}_K : \{0,1\}^{\ell_1} \to \{0,1\}^{\ell_2} \mid K \in \{0,1\}^{\lambda}\}$ *be a family of polynomially computable functions, where* ℓ_1 *and* ℓ_2 *are some polynomials of* λ*. We say that* F *is a pseudorandom function (PRF) family if, for any QPT distinguisher* $\mathcal{A}$*, there exists* $\mathsf{negl}(\cdot)$ *such that it holds that*

$$\left|\Pr\Big[\mathcal{A}^{\mathsf{F}_K(\cdot)}(1^\lambda) = 1 \mid K \leftarrow \{0,1\}^{\lambda}\Big] - \Pr\Big[\mathcal{A}^{\mathsf{R}(\cdot)}(1^\lambda) = 1 \mid \mathsf{R} \leftarrow \mathcal{U}\Big]\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{U}$ *is the set of all functions from* $\{0,1\}^{\ell_1}$ *to* $\{0,1\}^{\ell_2}$*.*

Theorem 2.7 ([GGM86]). *If one-way functions exist, then for all efficiently computable functions* $n(\lambda)$ *and* $m(\lambda)$*, there exists a PRF that maps* $n(\lambda)$ *bits to* $m(\lambda)$ *bits.*

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 as Lindell and Pinkas [LP09]. This is because we need special correctness for the construction of garbling in Appendix E.2. In fact, special correctness can be easily satisfied as well as shown by Lindell and Pinkas [LP09].

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 $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ow\text{-}cpa}}(\lambda)$ 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

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

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

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

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

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

Definition 2.13 (Ciphertext Pseudorandomness for SKE). *Let $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a SKE scheme whose ciphertext space is $\{0, 1\}^{\ell}$. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ct\text{-}pr}}(\lambda, b)$ 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_i to the challenger. If $b = 0$, the challenger computes $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sk}, m_i)$ and returns ct_i to $\mathcal{A}$. If $b = 1$, the challenger chooses $\mathsf{ct}_i \leftarrow \{0, 1\}^{\ell}$ and returns ct_i to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.*

3. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$. *This is the output of the experiment.*

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

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

Theorem 2.14. *If OWFs exist, there exists an SKE scheme that is ciphertext pseudorandom.*

Public Key Encryption (PKE).

Definition 2.15 (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** $\bot$**:** *The decryption algorithm takes as input* sk *and* ct*, and outputs a plaintext* m' *or* $\bot$.

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

Definition 2.16 (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 2.17 (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* $\mathsf{Exp}^{\mathsf{ow\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda)$ *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*

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

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

Definition 2.18 (IND-CPA Security for PKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a PKE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ind\text{-}cpa}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *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* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$*, 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*

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

It is well known that IND-CPA security implies OW-CPA security. 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.

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

Definition 2.19 (One-Time SKE with Certified Deletion (Syntax) [BI20, HMNY21]). *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* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *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* $\mathcal{D} := \{0,1\}^{r(\lambda)}$*.*

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

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

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

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

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

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

Definition 2.20 (Correctness for One-Time SKE with Certified Deletion). *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* $m \in \mathcal{M}$*,*

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

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

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

We additionally require verification correctness with QOTP in this work. This is because we need it for the construction of FE in Section 4.3. This notion means that even if we encrypt a ciphertext with quantum one-time pad (QOTP), we can run the original deletion algorithm $\mathcal{Del}$ and recover a valid certificate by using the QOTP key. In fact, the construction of [BI20] satisfies verification correctness with QOTP as well.

Definition 2.21 (Verification Correctness with QOTP). *There exists a negligible function* negl *and a PPT algorithm* $\mathsf{Recover}$ *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{sk}, \mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ ct \leftarrow \mathcal{E}nc(\mathsf{sk}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \widetilde{\mathsf{cert}} \leftarrow \mathcal{D}el\left(Z^b X^a ct X^a Z^b\right) \\ \mathsf{cert}^* \leftarrow \mathsf{Recover}(a, b, \widetilde{\mathsf{cert}}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

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

Definition 2.22 (Certified Deletion Security for One-Time SKE with Certified Deletion). *Let* $\Sigma = (\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *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* $ct \leftarrow \mathcal{E}nc(\mathsf{sk}, m_b)$ *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*

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

Theorem 2.23 ([BI20]). *There exists one-time SKE with certified deletion that satisfies Definitions 2.19 to 2.22 exists unconditionally.*

Attribute-Based Encryption.

Definition 2.24 (KP-ABE (Syntax)). *A key-policy ABE (KP-ABE) scheme is a tuple of PPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *with a class of predicates* $\mathcal{P}_n$ *(represented as class of boolean circuits with* n *input bits), and a message space* $\mathcal{M}$.

$\mathsf{Setup}(1^\lambda)$**:** *The setup algorithm takes as input a security parameter* 1^λ *and outputs a public key* pk *and a master secret key* msk.

$\mathsf{KeyGen}(\mathsf{msk}, P)$**:** *The key generation algorithm takes as input the master secret key* msk *and a predicate* $P \in \mathcal{P}_n$, *and outputs a secret key* sk_P *corresponding to the predicate* P.

$\mathsf{Enc}(\mathsf{pk}, a, m)$**:** *The encryption algorithm takes as input a public key* pk, *an attribute* $a \in \{0,1\}^n$ *and a message* $m \in \mathcal{M}$, *and outputs a ciphertext* ct.

$\mathsf{Dec}(\mathsf{sk}_P, \mathsf{ct})$**:** *The decryption algorithm takes as input a secret key* sk_P *and a ciphertext* ct, *and outputs a classical message* m' *or* $\bot$.

Definition 2.25 (Correctness of KP-ABE). *The correctness of KP-ABE for a class of predicates* $\mathcal{P}_n$ *and a message space* $\mathcal{M}$ *is defined as follows.*

Decryption correctness: *For any* $\lambda \in \mathbb{N}, P \in \mathcal{P}_n, a \in \{0,1\}^n, m \in \mathcal{M}$ *such that* $P(a) = 1$,

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

Definition 2.26 (IND-CPA Security of KP-ABE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a KP-ABE scheme for a class predicates* $\mathcal{P}_n$ *and message space* $\mathcal{M}$. *We consider the following experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$.

1. *The challenger computes* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *sends a predicate* $P_i \in \mathcal{P}_n$, *called key query, to the challenger and it returns* $\mathsf{sk}_{P_i} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, P)$. $\mathcal{A}$ *can send unbounded polynomially many key queries. Let* q *be the total number of key queries.*
3. $\mathcal{A}$ *sends a challenge message pair* (m_0, m_1) *and an attribute* $a \in \{0,1\}^n$ *such that* $|m_0| = |m_1|$.
4. *The challenger computes* $\mathsf{ct}_b \leftarrow \mathsf{Enc}(\mathsf{pk}, a, m_b)$. *It sends* ct_b *to* $\mathcal{A}$.
5. $\mathcal{A}$ *can send key queries again.*
6. $\mathcal{A}$ *outputs its guess* $b' \in \{0,1\}$. *If* $P_i(a) = 0$ *for all* $i \in [q]$, *the experiments outputs* b'.

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

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

We can define similar experiment $\mathsf{Exp}^{\mathsf{sel\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *where* $\mathcal{A}$ *is restricted to submit the challenge attribute* $a \in \{0,1\}^n$ *before it receives* pk *from the challenger. We say that the* Σ *is selectively secure if for any QPT adversary* $\mathcal{A}$, *it holds that*

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

Theorem 2.27 ([GVW15a, BGG⁺14]). *If the LWE assumption holds, there exists selectively secure KP-ABE for all boolean circuits. In addition, if the LWE assumption holds against sub-exponential time algorithms, there exists adaptively secure KP-ABE for all boolean circuits.*

Classical Fully Homomorphic Encryption.

Definition 2.28 (Leveled Fully Homomorphic Encryption). *A leveled FHE is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *with a class of circuits* $\mathcal{C} = \{\mathcal{C}_d\}_{d \in \mathbb{N}}$, *where* $\mathcal{C}_d$ *contains all Boolean circuits of depth up to* d.

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

$\mathsf{Enc}(\mathsf{pk}, x)$**:** *The encryption algorithm takes as input a public key* pk *and a message* $x \in \{0,1\}$, *and outputs a ciphertext* ct.

$\mathsf{Eval}(\mathsf{pk}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)$**:** *The evaluation algorithm takes as input a public key* pk, *a circuit* $\mathsf{C} \in \mathcal{C}$, *ciphertexts* $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ *where* n *denotes the input length of the circuit* C, *and outputs a ciphertext* ct_{C}.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$**:** *The decryption algorithm takes as input a secret key* sk *and a ciphertext* ct, *and outputs a message* x' *or* $\bot$.

Definition 2.29 (Compactness). *A classical FHE is compact if there exists a fixed polynomial bound* $B(\cdot)$ *so that, for all* $\lambda \in \mathbb{N}$, *any* $\mathsf{C} \in \mathcal{C}$, *and plaintext* $x \in \{0,1\}^n$, *it holds that*

$$\Pr\left[|\mathsf{ct}_{\mathsf{C}}| \leq B(\lambda) \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i) \\ \mathsf{ct}_{\mathsf{C}} \leftarrow \mathsf{Eval}(\mathsf{pk}, \mathsf{C}, \mathsf{ct}_1, \ldots, \mathsf{ct}_n)\end{array}\right] = 1.$$

Definition 2.30 (Correctness). *An FHE scheme is said to be correct for $\mathcal{C}$ if for any $\lambda, n \in \mathbb{N}, C \in \mathcal{C}, \boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$,*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_C) \neq C(\boldsymbol{x}) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{pk}, x_i) \\ \mathsf{ct}_C \leftarrow \mathsf{Eval}(\mathsf{pk}, C, \mathsf{ct}_1, \ldots, \mathsf{ct}_n) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.31 (Security of FHE). *An FHE scheme $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with a class of circuits $\mathcal{C}$ is said to be IND-CPA secure if for any QPT adversary $\mathcal{A}$, any $\lambda, n \in \mathbb{N}$, the following holds:*

$$\Pr\left[\mathcal{A}(1^\lambda, \mathsf{pk}, \mathsf{ct}) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, 0) \end{array}\right] - \Pr\left[\mathcal{A}(1^\lambda, \mathsf{pk}, \mathsf{ct}) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pk}, 1) \end{array}\right] = \mathsf{negl}(\lambda).$$

Theorem 2.32 ([BV14, GSW13]). *If the LWE assumption holds, there exists leveled FHE.*

Indistinguishability Obfuscation

Definition 2.33 (Indistinguishability Obfuscator [BGI⁺12]). *A PPT algorithm $i\mathcal{O}$ is a secure IO for a classical circuit class $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following two conditions.*

Functionality: *For any security parameter $\lambda \in \mathbb{N}$, circuit $C \in \mathcal{C}_\lambda$, and input x, we have that*

$$\Pr[C'(x) = C(x) \mid C' \leftarrow i\mathcal{O}(C)] = 1 \ .$$

Indistinguishability: *For any PPT* Sampler *and QPT distinguisher $\mathcal{D}$, the following holds:*

If $\Pr[\forall x\ C_0(x) = C_1(x) \wedge |C_0| = |C_1| \mid (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda)] > 1 - \mathsf{negl}(\lambda)$, *then we have*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{io}}_{i\mathcal{O},\mathcal{D}}(\lambda) := \Big|&\Pr\left[\mathcal{D}(i\mathcal{O}(C_0), \mathsf{aux}) = 1 \mid (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda)\right] \\ &- \Pr\left[\mathcal{D}(i\mathcal{O}(C_1), \mathsf{aux}) = 1 \mid (C_0, C_1, \mathsf{aux}) \leftarrow \mathsf{Sampler}(1^\lambda)\right]\Big| \leq \mathsf{negl}(\lambda).\end{aligned}$$

There are a few candidates of secure IO for polynomial-size classical circuits against quantum adversaries [BGMZ18, CHVW19, AP20].

Obfuscation for compute-and-compare programs.

Definition 2.34 (Compute-and-Compare Circuits). *A compute-and-compare circuit $\mathbf{CC}[P, \mathsf{lock}, m]$ is of the form*

$$\mathbf{CC}[P, \mathsf{lock}, m](x) = \begin{cases} m & (P(x) = \mathsf{lock}) \\ 0 & (\textit{otherwise}) \end{cases},$$

where P is a circuit, lock *is a string called lock value, and m is a message.*

We assume that a program P has an associated set of parameters pp_P (input size, output size, circuit size) which we do not need to hide.

Definition 2.35 (Compute-and-Compare Obfuscation). *A PPT algorithm* CC.Obf *is a secure obfuscator for the family of distributions $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$ if the following holds:*

Functionality Preserving: *There exists a negligible function* negl *such that for all program P, all lock value* lock*, and all message m, it holds that*

$$\Pr\left[\forall x, \widetilde{P}(x) = \mathbf{CC}[P, \mathsf{lock}, m](x) \mid \widetilde{P} \leftarrow \mathsf{CC.Obf}(1^\lambda, P, \mathsf{lock}, m)\right] \geq 1 - \mathsf{negl}(\lambda).$$

Distributional Indistinguishability: *There exists an efficient simulator* Sim *such that for all* $\mathcal{D}$, param *and message* m, *we have*

$$\left|\Pr\left[\mathcal{D}(\mathsf{CC.Obf}(1^\lambda, P, \mathsf{lock}, m), \mathit{aux}) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{Sim}(1^\lambda, \mathsf{pp}_P, 1^{|m|}), \mathit{aux}) = 1\right]\right| \leq \mathsf{negl}(\lambda),$$

where $(P, \mathsf{lock}, \mathit{aux}) \leftarrow D_{\mathsf{param}}$.

Theorem 2.36 ([GKW17, WZ17]). *If the LWE assumption holds, there exists compute-and-compare obfuscation for all families of distributions* $D = \{D_{\mathsf{param}}\}$, *where each* D_{param} *outputs uniformly random lock value* lock *independent of* P *and* aux.

3 Collusion-Resistant Functional Encryption with Certified Everlasting Deletion

In this section, we present the definitions of FE with certified everlasting deletion and a collusion-resistant construction.

3.1 Definitions

First, we introduce the syntax and security definitions of FE with certified everlasting deletion.

Definition 3.1 (Functional Encryption with Certified Everlasting Deletion). *A functional encryption with certified everlasting deletion for a class* $\mathcal{F}$ *of functions is a tuple of QPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *with plaintext space* $\mathcal{M}$, *ciphertext space* $\mathcal{C}$, *master public key space* $\mathcal{MPK}$, *master secret key space* $\mathcal{MSK}$, *and secret key space* $\mathcal{SK}$, *that works as follows.*

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

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

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

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

$\mathcal{D}el(ct) \to \mathsf{cert}$**:** *The deletion algorithm takes the ciphertext* ct *as input, and outputs a classical certificate* cert.

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

Definition 3.2 (Correctness of Functional Encryption with Certified Everlasting Deletion). *The correctness of FE with certified everlasting deletion for a class of functions* $\mathcal{F}$ *and plaintext space* $\mathcal{M}$ *is defined as follows.*

Evaluation Correctness: *For any* $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, *and* $f \in \mathcal{F}$,

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

Verification Correctness: *For any* $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, *and* $f \in \mathcal{F}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \neq \top \;\middle|\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (ct, \mathsf{vk}) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m) \\ \mathsf{cert} \leftarrow \mathcal{D}el(ct) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Remark 3.3. In FE, we should be able to run $\mathcal{Dec}$ algorithm for many different functions f on the same ciphertext ct. One might think that the quantum ct is destroyed by $\mathcal{Dec}$ algorithm, and it can be used only once. However, it is easy to see that $\mathcal{Dec}$ algorithm can be always modified so that it does not disturb the quantum state ct by using the gentle measurement lemma [Win99] thanks to the evaluation correctness.

Security notions. We define an indistinguishability-based security notion in the collusion-resistant setting in this section. We extend the certified everlasting security notion of PKE by Bartusek and Khurana [BK23] to the FE setting to obtain our indistinguishability-based security notion.

Definition 3.4 (Certified Everlasting Indistinguishable-Security of FE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be a functional encryption with certified everlasting deletion for a class* $\mathcal{F}$ *of functions, plaintext space* $\mathcal{M}$*. We consider two experiments* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *and* $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *played between a challenger and a non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$*. The experiments are defined as follows:*

1. *The challenger computes* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* MPK *to* $\mathcal{A}_\lambda(|\psi\rangle_\lambda)$.
2. $\mathcal{A}_\lambda$ *is allowed to make arbitrarily many key queries. 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* $\mathcal{A}_\lambda$.
3. $\mathcal{A}_\lambda$ *sends* $(m_0, m_1) \in \mathcal{M}^2$ *to the challenger.*
4. *The challenger computes* $(ct_b, \mathsf{vk}_b) \leftarrow \mathcal{Enc}(\mathsf{MPK}, m_b)$*, and sends* ct_b *to* $\mathcal{A}_\lambda$.
5. $\mathcal{A}_\lambda$ *is allowed to make arbitrarily many key queries. 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* $\mathcal{A}_\lambda$.
6. $\mathcal{A}_\lambda$ *sends a certificate of deletion* cert *and its internal state* ρ *to the challenger.*
7. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}_b, \mathsf{cert})$*. If the outcome is* $\top$ *and* $f_\ell(m_0) = f_\ell(m_1)$ *holds for all key queries* f_ℓ*, the experiment* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *outputs* ρ*; otherwise* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *outputs* $\perp$.
8. *The challenger sends the outcome of* $\mathsf{Vrfy}(\mathsf{vk}_b, \mathsf{cert})$ *to* $\mathcal{A}_\lambda$.
9. *Again,* $\mathcal{A}_\lambda$ *is allowed to make arbitrarily many key queries. 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* $\mathcal{A}_\lambda$.
10. $\mathcal{A}_\lambda$ *outputs its guess* b'*. If* $f_\ell(m_0) = f_\ell(m_1)$ *holds for all key queries* f_ℓ*, the experiment* $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *outputs* b'*; otherwise* $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *outputs* $\perp$.

We say that the Σ *is adaptively certified everlasting indistinguishable-secure if for any non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$*, it holds that*

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, 0), \mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, 1)) \leq \mathsf{negl}(\lambda),$$

and

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

3.2 Tools

We introduce a few tools for FE with certified everlasting deletion in this section.

(Interactive) certified everlasting lemma. First, we recall the certified everlasting lemma by Bartusek and Khurana [BK23].

Lemma 3.5 (Certified Everlasting Lemma [BK23, ePrint Ver. 20221122:050839]). *Let $\{\mathcal{Z}_\lambda(\theta)\}_{\lambda\in\mathbb{N},\theta\in\{0,1\}^\lambda}$ be a family of distributions over either classical bit strings or quantum states, and let $\mathfrak{A}$ be any class of adversaries such that for any $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}} \in \mathfrak{A}$, it holds that*

$$\left|\Pr[\mathcal{A}_\lambda(\mathcal{Z}_\lambda(\theta)) = 1] - \Pr\left[\mathcal{A}_\lambda(\mathcal{Z}_\lambda(0^\lambda)) = 1\right]\right| \le \mathsf{negl}(\lambda).$$

For any $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}} \in \mathfrak{A}$, consider the following distribution $\{\widetilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(b)\}_{\lambda\in\mathbb{N},b\in\{0,1\}}$ over quantum states, obtained by running $\mathcal{A}_\lambda$ as follows:

- *Sample $z,\theta \leftarrow \{0,1\}^\lambda$ and initialize $\mathcal{A}_\lambda$ with $(|z\rangle_\theta, b \oplus \bigoplus_{i:\theta_i=0} z_i, \mathcal{Z}_\lambda(\theta))$.*
- *$\mathcal{A}_\lambda$'s output is parsed as bit string $z' \in \{0,1\}^\lambda$ and a residual quantum state ρ.*
- *If $z_i = z'_i$ for all i such that $\theta_i = 1$ then output ρ, and otherwise output a special symbol $\bot$.*

Then, it holds that

$$\mathsf{TD}(\widetilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(0), \widetilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(1)) \le \mathsf{negl}(\lambda).$$

Remark 3.6. Although the description of the lemma above is slightly different from the original version (see the first item of Remark 3.8), it is essentially the same as what Bartusek and Khurana [BK23] proved. In addition, even if we put $b \oplus \bigoplus_{i:\theta_i=0} z_i$ in $\mathcal{Z}_\lambda$ with θ, the lemma holds.

We can generalize Lemma 3.5 to a variant in the interactive game setting as follows.

Lemma 3.7 (Interactive Certified Everlasting Lemma [BK23]). *For interactive QPT algorithms $\mathcal{A}$ and $\mathcal{C}$, $\theta \in \{0,1\}^\lambda$, and $\beta \in \{0,1\}$, let $\mathsf{Expt}_{\mathcal{A},\mathcal{C}}(\lambda,\theta,\beta)$ be an experiment that works as follows:*

- *$\mathcal{A}$ takes 1^λ as input and $\mathcal{C}$ takes $(1^\lambda,\theta,\beta)$ as input.*
- *$\mathcal{A}$ and $\mathcal{C}$ interact with each other through a quantum channel.*
- *$\mathcal{A}$ outputs a bit b', which is treated as the output of the experiment.*

For interactive QPT algorithms $\mathcal{A}'$ and $\mathcal{C}$ and $b \in \{0,1\}$, let $\widetilde{\mathsf{Expt}}_{\mathcal{A}',\mathcal{C}}(\lambda,b)$ be an experiment that works as follows:

- *Sample $z,\theta \leftarrow \{0,1\}^\lambda$.*
- *$\mathcal{A}'$ takes $(1^\lambda, |z\rangle_\theta)$ as input and $\mathcal{C}$ takes $(1^\lambda,\theta,b \oplus \bigoplus_{i:\theta_i=0} z_i)$ as input.*
- *$\mathcal{A}'$ and $\mathcal{C}$ interact with each other through a quantum channel.*
- *$\mathcal{A}'$ outputs a string $z' \in \{0,1\}^\lambda$ and a quantum state ρ.*
- *If $z_i = z'_i$ for all i such that $\theta_i = 1$ then the experiment outputs ρ, and otherwise it outputs a special symbol $\bot$.*

For a QPT algorithm $\mathcal{C}$, if for any QPT algorithm $\mathcal{A}$, $\theta \in \{0,1\}^\lambda$, and $\beta \in \{0,1\}$, it holds that

$$\left|\Pr\left[\mathsf{Expt}_{\mathcal{A},\mathcal{C}}(\lambda,\theta,\beta) = 1\right] - \Pr\left[\mathsf{Expt}_{\mathcal{A},\mathcal{C}}(\lambda,0^\lambda,\beta) = 1\right]\right| \le \mathsf{negl}(\lambda),$$

then for any QPT algorithm $\mathcal{A}'$, it holds that

$$\mathsf{TD}(\widetilde{\mathsf{Expt}}_{\mathcal{A}',\mathcal{C}}(\lambda,0), \widetilde{\mathsf{Expt}}_{\mathcal{A}',\mathcal{C}}(\lambda,1)) \le \mathsf{negl}(\lambda).$$

Remark 3.8. There are the following three differences from the certified everlasting lemma of [BK23] besides notational adaptations.

1. The challenger can use θ in an arbitrary manner whereas they require the challenger to use θ in a bit-by-bit manner.[16]
2. We consider an interactive setting whereas they consider a non-interactive setting.
3. The challenger also takes $b \oplus \bigoplus_{i:\theta_i=0} z_i$ as part of its input.

Indeed, we believe that the above variant is implicitly used in the security proof of their certified everlasting secure ABE in [BK23]. We observe that the above variant can be proven in essentially the same way as their original proof.

Public-Slot functional encryption. We introduce a new primitive which we call public-slot functional encryption. In this primitive, a decryption key is associated with two-input function where the first and second inputs are referred to as secret and public inputs, respectively. Given a ciphertext of a message m and a decryption key for a two-input function f, one can compute $f(m, \mathsf{pub})$ for any public input pub. In the security experiment, we require that a pair of challenge messages (m_0, m_1) must satisfy $f(m_0, \mathsf{pub}) = f(m_1, \mathsf{pub})$ for all key queries f and public inputs pub to prevent trivial attacks. A formal definition is given below.

Definition 3.9 (Public-Slot FE (Syntax)). *A public-slot functional encryption scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$, master public key space $\mathcal{MPK}$, master secret key space $\mathcal{MSK}$, secret key space $\mathcal{SK}$, and public input space $\mathcal{P}$, that work as follows.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$**:** *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 key generation algorithm takes MSK and $f \in \mathcal{F}$ as input, and outputs a secret key $\mathsf{sk}_f \in \mathcal{SK}$.*

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

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

We require that an FE with certified everlasting deletion scheme satisfies correctness defined below.

Definition 3.10 (Correctness of Public-Slot FE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $f \in \mathcal{F}$, and $\mathsf{pub} \in \mathcal{P}$,*

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

Definition 3.11 (Security of Public-Slot FE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-slot FE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

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 arbitrarily many key queries. 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 $\mathcal{A}$.*
3. *$\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.*

[16] In their notation, the challenger corresponds to $\mathcal{Z}(\theta)$.

4. *The challenger computes* $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_b)$ *and sends* CT *to* $\mathcal{A}$.
5. *Again,* $\mathcal{A}$ *is allowed to make arbitrarily many key queries.*
6. $\mathcal{A}$ *outputs* $b' \in \{0,1\}$. *If* $f_\ell(m_0, \mathsf{pub}) = f_\ell(m_1, \mathsf{pub})$ *holds for all key queries* f_ℓ *and public inputs* $\mathsf{pub} \in \mathcal{P}$, *the experiment outputs* b'. *Otherwise, it outputs* $\perp$.

We say that Σ *is adaptively indistinguishable-secure if for any QPT adversary* $\mathcal{A}$ *it holds that*

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

It is easy to construct selectively single-ciphertext public-slot SKFE from multi-input FE by Goldwasser et al., which can be instantiated with IO [GGG+14]. We convert it to adaptively indistinguishable-secure public-slot PKFE via a few transformation. We prove the following theorem in Appendix B.4.

Theorem 3.12. *If there exist IO and OWFs, there exists an adaptively indistinguishable-secure public-slot PKFE for* $\mathsf{P/poly}$.

3.3 Collusion-Resistant Construction

Ingredients. We need the following building blocks.

- Public-slot FE $\mathsf{FE} = \mathsf{FE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for all polynomial-size circuits.
- PRG $\mathsf{PRG} : \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$.

Scheme description. Our FE with certified everlasting deletion scheme $\mathsf{CED} = \mathsf{CED}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ is described below.

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

1. Generate $(\mathsf{fe}.\mathsf{MPK}, \mathsf{fe}.\mathsf{MSK}) \leftarrow \mathsf{FE}.\mathsf{Setup}(1^\lambda)$.
2. Output $\mathsf{MPK} := \mathsf{fe}.\mathsf{MPK}$ and $\mathsf{MSK} := \mathsf{fe}.\mathsf{MSK}$.

$\mathsf{CED}.\mathsf{KeyGen}(\mathsf{msk}, f)$:

1. Parse $\mathsf{MSK} = \mathsf{fe}.\mathsf{MSK}$.
2. Generate $\mathsf{fe}.\mathsf{sk}_{g[f]} \leftarrow \mathsf{FE}.\mathsf{KeyGen}(\mathsf{fe}.\mathsf{MSK}, g[f])$ where $g[f]$ is a function described in Figure 1.
3. Output $\mathsf{sk}_f = \mathsf{fe}.\mathsf{sk}_{g[f]}$.

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

1. Parse $\mathsf{MPK} = \mathsf{fe}.\mathsf{MPK}$.
2. Generate $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for every $i \in [2n+1]$.
3. Generate $u_{i,j,b} \leftarrow \{0,1\}^\lambda$ and compute $v_{i,j,b} \leftarrow \mathsf{PRG}(u_{i,j,b})$ for every $i \in [2n+1]$, $j \in [\lambda]$, and $b \in \{0,1\}$. Set $U := (u_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$ and $V := (v_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$.
4. Generate a state

$$|\psi_{i,j}\rangle := \begin{cases} |z_{i,j}\rangle \left|u_{i,j,z_{i,j}}\right\rangle & \text{if } \theta_{i,j} = 0 \\ |0\rangle \left|u_{i,j,0}\right\rangle + (-1)^{z_{i,j}} |1\rangle \left|u_{i,j,1}\right\rangle & \text{if } \theta_{i,j} = 1 \end{cases}$$

where $\theta_{i,j}$ (resp. $z_{i,j}$) is the j-th bit of θ_i (resp. z_i) for every $i \in [2n+1]$ and $j \in [\lambda]$.

$\underline{g[f]}$

Secret Input: $V, \theta_1, \ldots, \theta_{2n+1}, \beta_1, \ldots, \beta_{2n+1}$
Public Input: $(b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}$

1. Parse $V = (v_{i,j,b})_{i \in [2n+1], j \in [\lambda], b \in \{0,1\}}$.
2. Check if $\mathsf{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$. If so, go to the next step and otherwise output $\bot$.
3. Compute $m_i := \beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j}$ for every $i \in [2n+1]$.
4. Output $f(m_1 \| \cdots \| m_n)$ if $m_{2n+1} = 0$ and output $f(m_{n+1} \| \cdots \| m_{2n})$ otherwise.

Figure 1: The description of the function $g[f]$

5. Generate

$$\beta_i := \begin{cases} m_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n+1] \end{cases}.$$

6. Generate $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.MPK}, V \| \theta_1 \| \ldots \| \theta_{2n+1} \| \beta_1 \| \ldots \| \beta_{2n+1})$.
7. Output $ct = (\mathsf{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ and $\mathsf{vk} = (U, (z_i, \theta_i)_{i \in [2n+1]})$.

$\mathsf{CED}.\mathcal{Dec}(\mathsf{sk}_f, ct)$**:**

1. Parse $\mathsf{sk}_f \leftarrow \mathsf{fe.sk}_{g[f]}$ and $ct = (\mathsf{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$.
2. Coherently apply $\mathsf{FE.Dec}(\mathsf{fe.sk}_{g[f]}, \mathsf{fe.ct}, \cdot)$ on $\bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle$ and measure the outcome y.
3. Output y.

$\mathsf{CED}.\mathcal{Del}(ct)$**:**

1. Parse $ct = (\mathsf{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$.
2. Measure $|\psi_{i,j}\rangle$ in the Hadamard basis to get $c_{i,j} \| d_{i,j} \in \{0,1\}^{\lambda+1}$ for every $i \in [2n+1]$ and $j \in [\lambda]$.
3. Output $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i \in [2n+1], j \in [\lambda]}$.

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

1. Parse $\mathsf{vk} = (U, (z_i, \theta_i)_{i \in [2n+1]})$ and $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i \in [2n+1], j \in [\lambda]}$.
2. Check if $z_{i,j} = c_{i,j} \oplus d_{i,j} \cdot (u_{i,j,0} \oplus u_{i,j,1})$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 1$, where $z_{i,j}$ is the j-th bit of z_i. If so, output $\top$ and otherwise output $\bot$.

Theorem 3.13. *If* FE *is adaptively indistinguishable-secure public-slot FE for* $\mathsf{P/poly}$ *and* PRG *is a secure PRG,* CED *is adaptively certified everlasting indistinguishable-secure FE for* $\mathsf{P/poly}$.

Decryption Correctness. Let $ct = (\mathsf{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ be an honestly generated ciphertext for a message m and $\mathsf{sk}_f = \mathsf{fe.sk}_{g[f]}$ be an honestly generated decryption key for a function f. Then we have $\mathsf{fe.ct} \in \mathsf{FE.Enc}(\mathsf{fe.MPK}, V \| \theta_1 \| \ldots \| \theta_{2n+1} \| \beta_1 \| \ldots \| \beta_{2n+1})$ and $|\psi_{i,j}\rangle = |z_{i,j}\rangle \left|u_{i,j,z_{i,j}}\right\rangle$ for all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. Since we have $\mathsf{PRG}(u_{i,j,b}) = v_{i,j,b}$ for all $i \in [2n+1]$, $j \in [\lambda]$, and $b \in \{0,1\}$, and $m_{2n+1} =$

$\beta_{2n+1} \oplus \bigoplus_{j:\theta_{2n+1,j}=0} z_{i,j} = 0$, if we coherently run $g[f](V\|\theta_1\|\ldots\|\theta_{2n+1}\|\beta_1\|\ldots\|\beta_{2n+1},\cdot)$ on $\bigotimes_{i\in[2n+1],j\in[\lambda]} |\psi_{i,j}\rangle$ and measure the output, then the resulting outcome is $f(\beta_1 \oplus \bigoplus_{j:\theta_{1,j}=0} z_{1,j}\|\ldots\|\beta_n \oplus \bigoplus_{j:\theta_{n,j}=0} z_{n,j}) = f(m)$. Then the decryption correctness follows from that of FE.

Verification Correctness. Let $ct = (\mathsf{fe.ct}, \bigotimes_{i\in[2n+1],j\in[\lambda]} |\psi_{i,j}\rangle)$ be an honestly generated ciphertext and $\mathsf{vk} = (U,(z_i,\theta_i)_{i\in[2n+1]})$ be the corresponding verification key. For all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 1$, since we have $|\psi_{i,j}\rangle = |0\rangle |u_{i,j,0}\rangle + (-1)^{z_{i,j}} |1\rangle |u_{i,j,1}\rangle$, if we measure it in the Hadamard basis, then the outcome $c_i\|d_i$ satisfies $z_{i,j} = c_{i,j} \oplus d_{i,j} \cdot (u_{i,j,0} \oplus u_{i,j,1})$. This immediately implies the verification correctness.

Security for $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{CED},\mathcal{A}}(\lambda, b)$. We omit the proof in the main body. See Appendix A.

Security for $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\mathsf{CED},\mathcal{A}}(\lambda, b)$. Let $\mathcal{A}$ be an adversary against the adaptive certified everlasting indistinguishable-security. We consider the following sequence of hybrids.

Hyb_0: This is the original everlasting adaptive security experiment where the challenge bit is set to be 0. Specifically, it works as follows:

1. The challenger generates $(\mathsf{fe.MPK}, \mathsf{fe.MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{MPK} := \mathsf{fe.MPK}$ and $\mathsf{MSK} := \mathsf{fe.MSK}$, and sends MPK to $\mathcal{A}$.
2. $\mathcal{A}$ can make arbitrarily many key queries at any point of the experiment. When it makes a key query f, the challenger generates $\mathsf{fe.sk}_{g[f]} \leftarrow \mathsf{FE.KeyGen}(\mathsf{fe.MSK}, g[f])$ and returns $\mathsf{sk}_f = \mathsf{fe.sk}_{g[f]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $(m^{(0)}, m^{(1)})$ to the challenger.[17] It must satisfy $f(m^{(0)}) = f(m^{(1)})$ for all key queries f that are made before or after sending $(m^{(0)}, m^{(1)})$.
4. The challenger generates $(ct, \mathsf{vk}) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m^{(0)})$. Specifically,
   (a) Generate $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for every $i \in [2n+1]$.
   (b) Generate $u_{i,j,b} \leftarrow \{0,1\}^\lambda$ and compute $v_{i,j,b} \leftarrow \mathsf{PRG}(u_{i,j,b})$ for every $i \in [2n+1]$, $j \in [\lambda]$ and $b \in \{0,1\}$ and set $U = (u_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$ and $V := (v_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$.
   (c) Generate a state
   $$|\psi_{i,j}\rangle := \begin{cases} |z_{i,j}\rangle \left|u_{i,j,z_{i,j}}\right\rangle & \text{if } \theta_{i,j} = 0 \\ |0\rangle |u_{i,j,0}\rangle + (-1)^{z_{i,j}} |1\rangle |u_{i,j,1}\rangle & \text{if } \theta_{i,j} = 1 \end{cases}$$
   where $\theta_{i,j}$ (resp. $z_{i,j}$) is the j-th bit of θ_i (resp. z_i) for every $i \in [2n+1]$ and $j \in [\lambda]$.
   (d) Generate
   $$\beta_i := \begin{cases} m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n+1] \end{cases}.$$
   (e) Generate $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.MPK}, V\|\theta_1\|\ldots\|\theta_{2n+1}\|\beta_1\|\ldots\|\beta_{2n+1})$.
   (f) Set $ct = (\mathsf{fe.ct}, \bigotimes_{i\in[2n+1],j\in[\lambda]} |\psi_{i,j}\rangle)$ and $\mathsf{vk} = (U, (z_i, \theta_i)_{i\in[2n+1]})$.

   The challenger sends ct to $\mathcal{A}$.
5. $\mathcal{A}$ sends $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i\in[2n+1],j\in[\lambda]}$ and its internal state ρ to the challenger.
6. The challenger checks if $z_{i,j} = c_{i,j} \oplus d_{i,j} \cdot (u_{i,j,0} \oplus u_{i,j,1})$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 1$. If it does not hold, the challenger outputs $\perp$ as a final output of the experiment. Otherwise, go to the next step.

[17] We use $(m^{(0)}, m^{(1)})$ instead of (m_0, m_1) to denote a pair of challenge messages to avoid a notational collision.

7. The experiment outputs ρ as a final output.

$\mathsf{Hyb}_{1,k}$**:** For $k = 0, 1, \ldots, n$, this hybrid is identical to Hyb_0 except that the way of generating β_i is modified as follows:

$$\beta_i := \begin{cases} m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ m_{i-n}^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, n+k] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+k+1, 2n+1] \end{cases}.$$

Remark that $\mathsf{Hyb}_{1,0}$ is identical to Hyb_0.

Hyb_2**:** This hybrid is identical to $\mathsf{Hyb}_{1,n}$ except that β_{2n+1} is flipped. That is, β_i is generated as follows.

$$\beta_i := \begin{cases} m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ m_{i-n}^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n] \\ 1 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i = 2n+1 \end{cases}.$$

$\mathsf{Hyb}_{3,k}$**:** For $k = 0, 1, \ldots, n$, this hybrid is identical to Hyb_2 except that the way of generating β_i is modified as follows:

$$\beta_i := \begin{cases} m_i^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [k] \\ m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [k+1, n] \\ m_{i-n}^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n] \\ 1 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+k+1, 2n+1] \end{cases}.$$

Remark that $\mathsf{Hyb}_{3,0}$ is identical to Hyb_2.

Hyb_4**:** This hybrid is identical to $\mathsf{Hyb}_{3,n}$ except that β_{2n+1} is flipped. That is, β_i is generated as follows.

$$\beta_i := \begin{cases} m_i^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ m_{i-n}^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i = 2n+1 \end{cases}.$$

$\mathsf{Hyb}_{5,k}$**:** For $k = 0, 1, \ldots, n$, this hybrid is identical to Hyb_4 except that the way of generating β_i is modified as follows:

$$\beta_i := \begin{cases} m_i^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, n+k] \\ m_{i-n}^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+k+1, 2n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i = 2n+1 \end{cases}.$$

Remark that $\mathsf{Hyb}_{5,0}$ is identical to Hyb_4.

Remark that $\mathsf{Hyb}_{5,n}$ is exactly the everlasting adaptive experiment where the challenge bit is set to be 1. Thus, we have to prove

$$\mathsf{TD}(\mathsf{Hyb}_0, \mathsf{Hyb}_{5,n}) \leq \mathsf{negl}(\lambda). \tag{1}$$

We prove this by the following lemmata.

Lemma 3.14. *If* FE *is adaptively secure and* PRG *is a secure PRG, for any* $k \in [n]$,

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \leq \mathsf{negl}(\lambda).$$

Lemma 3.15. *If* FE *is adaptively secure and* PRG *is a secure PRG,*

$$\mathsf{TD}(\mathsf{Hyb}_{1,n}, \mathsf{Hyb}_2) \le \mathsf{negl}(\lambda).$$

Lemma 3.16. *If* FE *is adaptively secure and* PRG *is a secure PRG, for any* $k \in [n]$*,*

$$\mathsf{TD}(\mathsf{Hyb}_{3,k-1}, \mathsf{Hyb}_{3,k}) \le \mathsf{negl}(\lambda).$$

Lemma 3.17. *If* FE *is adaptively secure and* PRG *is a secure PRG,*

$$\mathsf{TD}(\mathsf{Hyb}_{3,n}, \mathsf{Hyb}_4) \le \mathsf{negl}(\lambda).$$

Lemma 3.18. *If* FE *is adaptively secure and* PRG *is a secure PRG, for any* $k \in [n]$*,*

$$\mathsf{TD}(\mathsf{Hyb}_{5,k-1}, \mathsf{Hyb}_{5,k}) \le \mathsf{negl}(\lambda).$$

Noting that $\mathsf{Hyb}_{1,0}$, $\mathsf{Hyb}_{3,0}$, and $\mathsf{Hyb}_{5,0}$ are identical to Hyb_0, Hyb_2, and Hyb_4, respectively, Lemmata 3.14 to 3.18 imply Equation (1).

What is left is to prove these lemmata. Actually, the proofs of these lemmata are very similar. We give a full proof of Lemma 3.14 below. After that, we also explain how to modify it to prove Lemma 3.15. The proofs of Lemmata 3.16 and 3.18 are almost identical to that of Lemma 3.14 and the proof of Lemma 3.17 is almost identical to that of Lemma 3.15, and thus we omit them.

Proof of Lemma 3.14. First, we prove Lemma 3.14 below.

Proof. For applying Lemma 3.7, we consider the following experiment $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$ between a QPT adversary $\mathcal{B}$ and a challenger $\mathcal{C}$ for $\theta \in \{0,1\}^\lambda$ and $\beta \in \{0,1\}$ as follows:

$\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$**:** In this experiment, $\mathcal{B}$ and $\mathcal{C}$ play the roles of $\mathcal{A}$ and the challenger of $\mathsf{Hyb}_{1,k}$ with the differences that $\mathcal{C}$ sets $\beta_{n+k} := \beta$ and $\theta_{n+k} := \theta$, $\mathcal{C}$ does not generate $\left|\psi_{n+k,j}\right\rangle$ for $j \in [\lambda]$ and thus not send it to $\mathcal{B}$, $\mathcal{C}$ additionally sends $\{u_{n+k,j,b}\}_{j\in[\lambda],b\in\{0,1\}}$ to $\mathcal{B}$, and $\mathcal{B}$ finally outputs a bit b' instead of a certificate. Specifically, it works as follows:

1. $\mathcal{C}$ generates $(\mathsf{fe.MPK}, \mathsf{fe.MSK}) \leftarrow \mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{MPK} := \mathsf{fe.MPK}$ and $\mathsf{MSK} := \mathsf{fe.MSK}$, and sends MPK to $\mathcal{B}$.
2. $\mathcal{B}$ can make arbitrarily many key queries at any point of the experiment. When it makes a key query f, $\mathcal{C}$ generates $\mathsf{fe.sk}_{g[f]} \leftarrow \mathsf{FE.KeyGen}(\mathsf{fe.MSK}, g[f])$ and returns $\mathsf{sk}_f = \mathsf{fe.sk}_{g[f]}$ to $\mathcal{B}$.
3. $\mathcal{B}$ sends $(m^{(0)}, m^{(1)})$ to $\mathcal{C}$. It must satisfy $f(m^{(0)}) = f(m^{(1)})$ for all key queries f that are made before or after sending $(m^{(0)}, m^{(1)})$.
4. $\mathcal{C}$ does the following:
   (a) Generate $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for every $i \in [2n+1] \setminus \{n+k\}$ and set $\theta_{n+k} := \theta$.
   (b) Generate $u_{i,j,b} \leftarrow \{0,1\}^\lambda$ and compute $v_{i,j,b} \leftarrow \mathsf{PRG}(u_{i,j,b})$ for every $i \in [2n+1]$, $j \in [\lambda]$ and $b \in \{0,1\}$ and set $U = (u_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$ and $V := (v_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$.
   (c) Generate a state
   $$\left|\psi_{i,j}\right\rangle := \begin{cases} \left|z_{i,j}\right\rangle \left|u_{i,j,z_{i,j}}\right\rangle & \text{if } \theta_{i,j} = 0 \\ \left|0\right\rangle \left|u_{i,j,0}\right\rangle + (-1)^{z_{i,j}} \left|1\right\rangle \left|u_{i,j,1}\right\rangle & \text{if } \theta_{i,j} = 1 \end{cases}$$
   where $\theta_{i,j}$ (resp. $z_{i,j}$) is the j-th bit of θ_i (resp. z_i) for every $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$.

(d) Generate

$$\beta_i := \begin{cases} m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ m_{i-n}^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, n+k-1] \\ \beta & \text{if } i = n+k \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+k+1, 2n+1] \end{cases}.$$

(e) Generate $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.MPK}, V\|\theta_1\|\ldots\|\theta_{2n+1}\|\beta_1\|\ldots\|\beta_{2n+1})$.

$\mathcal{C}$ sends $(\mathsf{fe.ct}, \bigotimes_{i\in[2n+1]\setminus\{n+k\}, j\in[\lambda]} |\psi_{i,j}\rangle, \{u_{n+k,j,b}\}_{j\in[\lambda], b\in\{0,1\}})$ to $\mathcal{B}$.

5. $\mathcal{B}$ outputs a bit b' as a final output of the experiment.

We prove the following lemma.

Lemma 3.19. *For any QPT $\mathcal{B}$,*

$$\left|\Pr[\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)=1] - \Pr[\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,0^n,\beta)=1]\right| \leq \mathsf{negl}(\lambda).$$

Before proving Lemma 3.19, we complete the proof of Lemma 3.14 assuming that Lemma 3.19 is true. By Lemmata 3.7 and 3.19, for any QPT adversary $\mathcal{B}'$, we have

$$\mathsf{TD}(\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,0), \widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,1)) \leq \mathsf{negl}(\lambda). \tag{2}$$

where $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,b)$ is an experiment that works as follows:

$\widetilde{\mathsf{Expt}}^{1.k}_{\mathcal{B}',\mathcal{C}}(\lambda,b)$:

1. Sample $z,\theta \leftarrow \{0,1\}^\lambda$.
2. $\mathcal{B}'$ takes $(1^\lambda, |z\rangle_\theta)$ as input.
3. $\mathcal{B}'$ interacts with $\mathcal{C}$ as in $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,b\oplus\bigoplus_{j:\theta_{i,j}=0} z_{i,j})$ where $\mathcal{B}'$ plays the role of $\mathcal{B}$.
4. $\mathcal{B}'$ outputs a string $z' \in \{0,1\}^\lambda$ and a quantum state ρ.
5. If $z_j = z'_j$ for all $j \in [\lambda]$ such that $\theta_j = 1$ then the experiment outputs ρ, and otherwise it outputs a special symbol $\perp$.

Note that the only difference between $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ is that β_{n+k} is set to be $0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k-1}$ and $m_k^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k}$. If $m_k^{(1)} = 0$, then there is no difference. Thus, we assume that $m_k^{(1)} = 1$. Then we construct $\mathcal{B}'$ that distinguishes $\widetilde{\mathsf{Expt}}^{1.k}_{\mathcal{B}',\mathcal{C}}(\lambda,0)$ and $\widetilde{\mathsf{Expt}}^{1.k}_{\mathcal{B}',\mathcal{C}}(\lambda,1)$ using $\mathcal{A}$ that distinguishes $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ as follows.

$\mathcal{B}'(1^\lambda, |z\rangle_\theta)$:

1. $\mathcal{B}'$ plays the role of $\mathcal{A}$ in $\mathsf{Hyb}_{1,k}$ where the external challenger $\mathcal{C}$ of $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,b)$ is used to simulate the challenger of $\mathsf{Hyb}_{1,k}$. $\mathcal{C}$ provides everything that should be sent to $\mathcal{A}$ except for $|\psi_{n+k,j}\rangle$ for $j\in[\lambda]$. $\mathcal{B}'$ generates $|\psi_{n+k,j}\rangle$ by applying the map $|b\rangle \to |b\rangle |u_{n+k,j,b}\rangle$ on the j-th qubit of $|z\rangle_\theta$ and uses it as part of ct sent to $\mathcal{A}$. Note that this is possible since $(u_{n+k,j,b})_{j\in[\lambda],b\in\{0,1\}}$ is provided from $\mathcal{C}$.
2. Suppose that $\mathcal{A}$ returns a certificate $(c_{i,j}, d_{i,j})_{i\in[2n+1],j\in[\lambda]}$. $\mathcal{B}'$ sets $z'_{n+k,j} = c_{n+k,j} \oplus d_{n+k,j}\cdot(u_{n+k,j,0} \oplus u_{n+k,j,1})$ for $j\in[\lambda]$. Again, note that this is possible since $(u_{n+k,j,b})_{j\in[\lambda],b\in\{0,1\}}$ is provided from $\mathcal{C}$.

3. Output $z' := z'_{n+k,1} \| \dots \| z'_{n+k,\lambda}$ and $\mathcal{A}$'s internal state ρ.

We can see that $\mathcal{B}'$ perfectly simulates $\mathsf{Hyb}_{1,k-1}$ if $b = 0$ and $\mathsf{Hyb}_{1,k}$ if $b = 1$. (Recall that we are assuming $m_k^{(1)} = 1$.) Moreover, we have $z_j = z'_j$ for all $j \in [\lambda]$ (which is the condition to not output $\perp$ in $\widetilde{\mathsf{Expt}}^{1.k}_{\mathcal{B}',\mathcal{C}}(\lambda, b)$) whenever $z_{i,j} = c_{i,j} \oplus d_{i,j} \cdot (u_{i,j,0} \oplus u_{i,j,1})$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 1$ (which is the condition to not output $\perp$ in $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$). Therefore, we must have

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \leq \mathsf{TD}(\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, 0), \widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, 1)).$$

Combined with Equation (2), this completes the proof of Lemma 3.14. ☐

Now, we are left to prove Lemma 3.19.

Proof of Lemma 3.19. We further consider the following sequence of hybrids:

$\mathsf{Expt}^{1,k,a}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$**:** This is identical to $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$ except that $v_{i,j,1\oplus z_{i,j}}$ is uniformly chosen from $\{0,1\}^{2\lambda}$ instead of being set to be $\mathsf{PRG}(u_{i,j,1\oplus z_{i,j}})$ for all $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$.

$\mathsf{Expt}^{1,k,b}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$**:** This is identical to $\mathsf{Expt}^{1,k,a}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$ except that $\theta_{n+k} = \theta$ is replaced with 0^n. Note that θ_{n+k} only appears in the encrypted message for fe.ct in $\mathsf{Expt}^{1,k,a}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$.

Proposition 3.20. *If* PRG *is a secure PRG,*

$$\left| \Pr[\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta) = 1] - \Pr[\mathsf{Expt}^{1,k,a}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta) = 1] \right| \leq \mathsf{negl}(\lambda).$$

Proof. Noting that $u_{i,j,1\oplus z_{i,j}}$ for $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$ is used only for generating $v_{i,j,1\oplus z_{i,j}}$ in $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta)$, Proposition 3.20 directly follows from the security of PRG. ☐

Proposition 3.21. *If* FE *is adaptively secure,*

$$\left| \Pr[\mathsf{Expt}^{1,k,a}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta) = 1] - \Pr[\mathsf{Expt}^{1,k,b}_{\mathcal{B},\mathcal{C}}(\lambda, \theta, \beta) = 1] \right| \leq \mathsf{negl}(\lambda).$$

Proof. For each $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$, there is no u such that $\mathsf{PRG}(u) = v_{i,j,1\oplus z_{i,j}}$ except for probability $2^{-\lambda}$. Let Good be the event that the above holds for all $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. We have $\Pr[\mathsf{Good}] \geq 1 - 2n\lambda 2^{-\lambda} = 1 - \mathsf{negl}(\lambda)$. We prove that whenever Good occurs, we have

$$\begin{aligned} &g[f]((V, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}) \\ =&g[f]((V, \theta_1, \dots, \theta_{n+k-1}, 0^n, \theta_{n+k+1,\dots}\theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}) \end{aligned} \tag{3}$$

for all key queries f and $(b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}$. If this is proven, Proposition 3.21 directly follows from the adaptive security of FE.

Below, we prove Equation (3). We consider the following two cases.

- If $\mathsf{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$, then by the assumption that Good occurs, we have $b_{i,j} = z_{i,j}$ for all $i \in [2n+1] \setminus \{n+k\}$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. Then we have $\beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j} = m_i^{(0)}$ for $i \in [n]$ and $\beta_{2n+1} \oplus \bigoplus_{j:\theta_{2n+1,j}=0} b_{2n+1,j} = 0$. Then both sides of Equation (3) are equal to $f(m^{(0)})$.
- Otherwise, both sides of Equation (3) are equal to $\perp$.

In either case, Equation (3) holds. This completes the proof of Proposition 3.21. ☐

Proposition 3.22. *If* PRG *is a secure PRG,*

$$\left|\Pr[\mathsf{Expt}^{1,\mathsf{k},\mathsf{b}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)=1]-\Pr[\mathsf{Expt}^{1,\mathsf{k}}_{\mathcal{B},\mathcal{C}}(\lambda,0^n,\beta)=1]\right|\leq \mathsf{negl}(\lambda).$$

Proof. Noting that $u_{i,j,1\oplus z_{i,j}}$ for $i\in[2n+1]\setminus\{n+k\}$ and $j\in[\lambda]$ such that $\theta_{i,j}=0$ is used only for generating $v_{i,j,1\oplus z_{i,j}}$ in $\mathsf{Expt}^{1,\mathsf{k}}_{\mathcal{B},\mathcal{C}}(\lambda,0^n,\beta)$, Proposition 3.22 directly follows from the security of PRG. □

Lemma 3.19 follows from the above propositions. □

Proof of Lemma 3.15. Next, we prove Lemma 3.15.

Proof. Since the proof of Lemma 3.15 is very similar to that of Lemma 3.14, we only explain the difference. First, we define an experiment $\mathsf{Expt}^{2}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ that is similar to $\mathsf{Expt}^{1,\mathsf{k}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ except that $n+k$ is replaced with $2n+1$. Then by almost the same argument as that in the proof of Lemma 3.14 using Lemma 3.7, we only have to prove

$$\left|\Pr[\mathsf{Expt}^{2}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)=1]-\Pr[\mathsf{Expt}^{2}_{\mathcal{B},\mathcal{C}}(\lambda,0^n,\beta)=1]\right|\leq \mathsf{negl}(\lambda)$$

for all QPT $\mathcal{B}$. Its proof is also similar to that of Lemma 3.19. We define $\mathsf{Expt}^{2,\mathsf{a}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and $\mathsf{Expt}^{2,\mathsf{b}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ similarly to $\mathsf{Expt}^{1,\mathsf{k},\mathsf{a}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and $\mathsf{Expt}^{1,\mathsf{k},\mathsf{b}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ except that $n+k$ is replaced with $2n+1$. Then the computational indistinguishability between $\mathsf{Expt}^{2}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and $\mathsf{Expt}^{2,\mathsf{a}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and between $\mathsf{Expt}^{2,\mathsf{b}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and $\mathsf{Expt}^{2}_{\mathcal{B},\mathcal{C}}(\lambda,0^n,\beta)$ immediately follow from the security of PRG. We argue the computational indistinguishability between $\mathsf{Expt}^{2,\mathsf{a}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ and $\mathsf{Expt}^{2,\mathsf{b}}_{\mathcal{B},\mathcal{C}}(\lambda,\theta,\beta)$ based on the security of FE as follows.

Let Good be the event that there is no u such that $\mathsf{PRG}(u)=v_{i,j,1\oplus z_{i,j}}$ for all $i\in[2n]$ and $j\in[\lambda]$ such that $\theta_{i,j}=0$. We have $\Pr[\mathsf{Good}]\geq 1-\mathsf{negl}(\lambda)$. Similarly to the proof of Proposition 3.21, it suffices to prove that whenever Good occurs, we have

$$\begin{aligned}&g[f]((V,\theta_1,\ldots,\theta_{2n+1},\beta_1,\ldots,\beta_{2n+1}),(b_{i,j},u_{i,j})_{i\in[2n+1],j\in[\lambda]})\\=&g[f]((V,\theta_1,\ldots,\theta_{2n},0^\lambda,\beta_1,\ldots,\beta_{2n+1}),(b_{i,j},u_{i,j})_{i\in[2n+1],j\in[\lambda]})\end{aligned}\tag{4}$$

for all key queries f and $(b_{i,j},u_{i,j})_{i\in[2n+1],j\in[\lambda]}$. Below, we prove Equation (4). We consider the following two cases.

- If $\mathsf{PRG}(u_{i,j})=v_{i,j,b_{i,j}}$ holds for every $i\in[2n+1]$ and $j\in[\lambda]$, then by the assumption that Good occurs, we have $b_{i,j}=z_{i,j}$ for all $i\in[2n]$ and $j\in[\lambda]$ such that $\theta_{i,j}=0$. Then we have

$$\beta_i\oplus\bigoplus_{j:\theta_{i,j}=0}b_{i,j}=\begin{cases}m_i^{(0)} & \text{if } i\in[n]\\ m_{i-n}^{(1)} & \text{if } i\in[n+1,2n]\end{cases}.$$

  Then the LHS of Equation (4) is equal to $f(m^{(\gamma)})$ where $\gamma=\beta_{2n+1}\oplus\bigoplus_{j:\theta_{i,j}=0}b_{2n+1,j}$ and the RHS of Equation (4) is equal to $f(m^{(\gamma')})$ where $\gamma'=\beta_{2n+1}\oplus\bigoplus_{j\in[\lambda]}b_{2n+1,j}$. By the restriction on $\mathcal{B}$, we have $f(m^{(0)})=f(m^{(1)})$. Therefore, both sides of Equation (4) are equal to $f(m^{(0)})=f(m^{(1)})$.

- Otherwise, both sides of Equation (4) are equal to $\perp$.

In either case, Equation (4) holds. This completes the proof of Lemma 3.15. □

4 Bounded Collusion-Resistant Functional Encryption with Certified Everlasting Deletion

4.1 Definitions

We also require verification correctness with QOTP for q-bounded certified everlasting simulation-secure FE because we need it for the construction of certified everlasting secure FE in Section 4.3.

Definition 4.1 (Verification Correctness with QOTP). *There exists a negligible function* negl *and a PPT algorithm* $\mathsf{Recover}$ *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$*,*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}^*) = \bot \;\middle|\; \begin{array}{l} (\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^q) \\ (\mathsf{vk}, ct) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m) \\ a, b \leftarrow \{0,1\}^{p(\lambda)} \\ \widetilde{\mathsf{cert}} \leftarrow \mathcal{D}el(Z^b X^a \mathsf{ct} X^a Z^b) \\ \mathsf{cert}^* \leftarrow \mathsf{Recover}(a, b, \widetilde{\mathsf{cert}}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Another is an adaptively simulation-based security notion in the bounded collusion-resistant setting. The other is a non-adaptively simulation-based security notion in the bounded collusion-resistant setting. We consider only bounded collusion-resistance in the simulation-based definitions because achieving simulation-based security is impossible in the collusion-resistant setting [AGVW13].

Our simulation-based security notion is a natural extension of that in the classical FE setting [GVW12]. Note that the setup algorithm additionally takes 1^q as input in the bounded collusion-resistant setting where q is the total number of key queries.

Definition 4.2 (q-Bounded Certified Everlasting Simulation-Security). *Let* q *be a polynomial of* λ*. Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *be a* q*-bounded FE with certified everlasting deletion scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}_1$ *and an unbounded adversary* $\mathcal{A}_2$*. Let* $\mathcal{S}im_1$*,* $\mathcal{S}im_2$*, and* $\mathcal{S}im_3$ *be a QPT algorithm.*

1. *The challenger runs* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^q)$ *and sends* MPK *to* $\mathcal{A}_1$*.*
2. $\mathcal{A}_1$ *is allowed to make arbitrary key queries. 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_{pre} *be the number of times that* $\mathcal{A}_1$ *makes key queries in this phase. Let* $\mathcal{V} := \{y_i := f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i \in [q_{\mathsf{pre}}]}$*.*
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* $(\mathsf{vk}, ct) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m)$*, and sends* ct *to* $\mathcal{A}_1$*.*
   - *If* $b = 1$*, the challenger computes* $(ct, \mathsf{st}_{q_{\mathsf{pre}}}) \leftarrow \mathcal{S}im_1(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$*, and sends* ct *to* $\mathcal{A}_1$*, where* $\mathsf{st}_{q_{\mathsf{pre}}}$ *is a quantum state.*
5. $\mathcal{A}_1$ *is allowed to make arbitrary key queries at most* $q - q_{\mathsf{pre}}$ *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* $(\mathsf{sk}_{f_\ell}, \mathsf{st}_\ell) \leftarrow \mathcal{S}im_2(\mathsf{MSK}, f_\ell, f_\ell(m), \mathsf{st}_{\ell-1})$*, and sends* sk_{f_ℓ} *to* $\mathcal{A}_1$*, where* st_ℓ *is a quantum state.*
6. *If* $b = 1$*, the challenger runs* $\mathsf{vk} \leftarrow \mathcal{S}im_3(\mathsf{st}_q)$*.*
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 adaptively certified everlasting simulation-secure if there exists a QPT simulator* $\mathcal{S}im = (\mathcal{S}im_1, \mathcal{S}im_2, \mathcal{S}im_3)$ *such that for any QPT adversary* $\mathcal{A}_1$ *and any unbounded adversary* $\mathcal{A}_2$ *it holds that*

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

Remark 4.3. Note that Definitions 4.2 and 4.4 were presented before the work by Bartusek and Khurana [BK23] appeared. Although we can define simulation-based definitions based on the definitions by Bartusek and Khurana, we leave our original simulation-based definitions as a concurrent and independent work. We also note that the challenger can omit sending MSK to $\mathcal{A}_2$ in Definitions 4.2 and 4.4 in the public-key setting based on the results by Bartusek and Khurana [BK23, Claim A.3 and A.4].

We can consider a non-adaptive variant of the definition above.

Definition 4.4 (q-Bounded Non-Adaptive Certified Everlasting Simulation-Security). *Let q be a polynomial of* λ. *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *be a q-bounded FE with certified everlasting deletion scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}noada\text{-}sim}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}_1$ *and an unbounded adversary* $\mathcal{A}_2$. *Let* $\mathcal{S}im$ *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. 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 total number of times that* $\mathcal{A}_1$ *makes key queries. Let* $\mathcal{V} := \{y_i := f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i\in[q]}$.

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* $(\mathsf{vk}, ct) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m)$, *and sends* ct *to* $\mathcal{A}_1$.
   - *If* $b = 1$, *the challenger computes* $(\mathsf{vk}, ct) \leftarrow \mathcal{S}im(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$, *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 non-adaptive certified everlasting simulation-secure if there exists a QPT simulator* $\mathcal{S}im$ *such that for any QPT adversary* $\mathcal{A}_1$ *and any unbounded adversary* $\mathcal{A}_2$ *it holds that*

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

We need to consider standard simulation-security notions for FE with certified everlasting deletion. We note that the following two security definitions are simulation-based ones defined in [GVW12].

Definition 4.5 (q-Bounded Non-Adaptive Simulation-Security for FE with Certified Everlasting Deletion [GVW12]). *Let q be a polynomial of* λ. *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *be a q-bounded FE with certified everlasting deletion scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{non\text{-}ada\text{-}sim}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$. *Let* $\mathcal{S}im$ *be a QPT algorithm.*

1. *The challenger runs* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^q)$ *and sends* MPK *to* $\mathcal{A}$.

2. $\mathcal{A}$ *is allowed to make arbitrary key queries. 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 total 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* $(\mathsf{vk}, \mathcal{ct}) \leftarrow \mathcal{Enc}(\mathsf{MPK}, m)$*, and sends* $\mathcal{ct}$ *to* $\mathcal{A}$.
   - *If* $b = 1$*, the challenger computes* $\mathcal{ct} \leftarrow \mathcal{Sim}(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$*, and sends* $\mathcal{ct}$ *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* $\mathcal{Sim}$ *such that for any* QPT *adversary* $\mathcal{A}$ *it holds that*

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

Definition 4.6 (q-Bounded Adaptive Simulation-Security for FE with Certified Everlasting Deletion [GVW12]). *Let* q *be a polynomial of* λ*. Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be a* q*-bounded FE with certified everlasting deletion scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{ada\text{-}sim}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *against a QPT adversary* $\mathcal{A}$*. Let* $\mathcal{Sim}_1$ *and* $\mathcal{Sim}_2$ *be a QPT algorithm.*

1. *The challenger runs* $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^q)$ *and sends* MPK *to* $\mathcal{A}$.

2. $\mathcal{A}$ *is allowed to make arbitrary key queries. 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_{pre} *be the number of times that* $\mathcal{A}$ *makes key queries in this phase. Let* $\mathcal{V} := \{y_i := f_i(m), f_i, \mathsf{sk}_{f_i}\}_{i \in [q_{\mathsf{pre}}]}$.

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* $(\mathsf{vk}, \mathcal{ct}) \leftarrow \mathcal{Enc}(\mathsf{MPK}, m)$*, and sends* $\mathcal{ct}$ *to* $\mathcal{A}$.
   - *If* $b = 1$*, the challenger computes* $(\mathcal{ct}, \mathsf{st}_{q_{\mathsf{pre}}}) \leftarrow \mathcal{Sim}_1(\mathsf{MPK}, \mathcal{V}, 1^{|m|})$*, and sends* $\mathcal{ct}$ *to* $\mathcal{A}$*, where* $\mathsf{st}_{q_{\mathsf{pre}}}$ *is a quantum state.*

5. $\mathcal{A}$ *is allowed to make arbitrary key queries at most* $(q - q_{\mathsf{pre}})$ *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* $(\mathsf{sk}_{f_\ell}, \mathsf{st}_\ell) \leftarrow \mathcal{Sim}_2(\mathsf{MSK}, f_\ell, f_\ell(m), \mathsf{st}_{\ell-1})$*, 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 simulation-secure if there exists a* QPT *simulator* $\mathcal{Sim} = (\mathcal{Sim}_1, \mathcal{Sim}_2)$ *such that for any* QPT *adversary* $\mathcal{A}$ *it holds that*

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

4.2 1-Bounded Construction with Non-Adaptive Security

To achieve q-bounded adaptive certified everlasting simulation-secure FE in Section 4.4, we prepare building blocks in this section and Section 4.3. In this section, we construct a 1-bounded non-adaptive certified everlasting simulation-secure FE scheme from a certified everlasting secure garbling scheme (Definition E.1) and a certified everlasting secure PKE scheme (Definition C.8). See Appendices C.4, C.5 and E.2 for how to achieve these building blocks. Regarding PKE, we can also use the construction by Bartusek and Khurana [BK23].

Our 1-bounded non-adaptive certified everlasting secure FE scheme. This construction is essentially the same as the 1-bound FE by Sahai and Seyalioglu [SS10]. 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 non-adaptive certified everlasting secure FE scheme $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ from a certified everlasting secure garbling scheme $\Sigma_{\mathsf{cegc}} = \mathsf{GC}.(\mathsf{Setup}, \mathcal{Garble}, \mathcal{Eval}, \mathcal{Del}, \mathsf{Vrfy})$ (Definition E.1) and a certified everlasting secure PKE scheme $\Sigma_{\mathsf{cepk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ (Definition C.8).

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

- 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 $\mathsf{sk}_f := (f, \{\mathsf{pke.sk}_{i,f[i]}\}_{i\in[s]})$.

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

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

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

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

$\mathcal{Del}(ct)$:

- Parse $ct = (\widetilde{\mathcal{U}}, \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.
- Compute $\mathsf{gc.cert} \leftarrow \mathsf{GC}.\mathcal{Del}(\widetilde{\mathcal{U}})$.
- For every $i \in [s]$ and $\alpha \in \{0,1\}$, compute $\mathsf{pke.cert}_{i,\alpha} \leftarrow \mathsf{PKE}.\mathcal{Del}(\mathsf{pke.ct}_{i,\alpha})$.
- 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 4.7. *If Σ_{cegc} satisfies the selective security (Definition E.4) and Σ_{cepk} satisfies the IND-CPA security (Definition C.12), Σ_{cefe} satisfies the 1-bounded non-adaptive simulation-security (Definition 4.5).*

Its proof is similar to that of Theorem 4.8, and therefore we omit it.

Theorem 4.8. *If Σ_{cegc} satisfies the selective certified everlasting security (Definition E.5) and Σ_{cepk} satisfies the certified everlasting IND-CPA security (Definition C.13), Σ_{cefe} satisfies the 1-bounded non-adaptive certified everlasting simulation-security (Definition 4.4).*

Proof of Theorem 4.8. Let us describe how the simulator $\mathcal{Sim}$ works.

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

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 $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$ and $L^*_{i,f[i]\oplus 1} \leftarrow \mathcal{L}$ for every $i \in [s]$.
4. Compute $(\widetilde{\mathcal{U}}, \mathsf{gc.vk}) \leftarrow \mathsf{GC}.\mathcal{Sim}(1^\lambda, 1^{|f|}, U(f,m), \{L_{i,f[i]}\}_{i\in[s]})$.
5. Compute $(\mathsf{pke.vk}_{i,f[i]}, \mathsf{pke}.ct_{i,f[i]}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,f[i]}, L_{i,f[i]})$ and $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke}.ct_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L^*_{i,f[i]\oplus 1})$ 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 $ct := (\widetilde{\mathcal{U}}, \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$.

Let us define the sequence of hybrids as follows.

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

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 $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$, $(\widetilde{\mathcal{U}}, \mathsf{gc.vk}) \leftarrow \mathsf{GC}.\mathcal{Garble}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$, and $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke}.ct_{i,\alpha}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,\alpha}, L_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $(\widetilde{\mathcal{U}}, \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ 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 $\bot$, and sends $\bot$ 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 $\bot$.

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 $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$ in step 2 regardless of whether a key query is called or not. Third, the challenger does not compute $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$ in step 4. Fourth, the challenger computes $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke}.ct_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L^*_{i,f[i]\oplus 1})$ for every $i \in [s]$ instead of computing $(\mathsf{pke.vk}_{i,f[i]\oplus 1}, \mathsf{pke}.ct_{i,f[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,f[i]\oplus 1}, L_{i,f[i]\oplus 1})$ for every $i \in [s]$.

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

From the definition of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}non\text{-}adapt}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, b)$ and $\mathcal{S}im$, 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 4.8 easily follows from the following Propositions 4.9 and 4.10. (whose proof is given later.) ☐

Proposition 4.9. *If Σ_{cepk} satisfies the certified everlasting IND-CPA security,*

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

Proposition 4.10. *If Σ_{cegc} satisfies the certified everlasting selective security,*

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

Proof of Proposition 4.9. For the proof, we use Lemma D.9 whose statement and proof is given in Appendix D.2. 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 D.9. This contradicts the certified everlasting IND-CPA of Σ_{cepk} from Lemma D.9. 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 $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$. 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 $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$.
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 $(\widetilde{\mathcal{U}}, \mathsf{gc.vk}) \leftarrow \mathsf{GC}.\mathcal{G}arble(1^\lambda, U(\cdot,m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ and $(\mathsf{pke.vk}_{i,f[i]}, \mathsf{pke}.ct_{i,f[i]}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,f[i]}, L_{i,f[i]})$ for every $i \in [s]$, and sends $(\widetilde{\mathcal{U}}, \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ 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 $\bot$ from the challenger. If $\mathcal{B}$ receives $\bot$ from the challenger, it outputs $\bot$ 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 $\bot$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B} \mid b = 0] - \Pr[1 \leftarrow \mathcal{B} \mid b = 1]|$ is non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma D.9. ☐

Proof of Proposition 4.10. 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 selective certified everlasting 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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}sel\text{-}gbl}}_{\mathcal{B},\Sigma_{\mathsf{cegc}}}(1^\lambda, b)$.

5. The challenger computes $\{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}} \leftarrow \mathsf{GC.Setup}(1^\lambda)$ and does the following:
   - If $b = 0$, the challenger computes $(\widetilde{\mathcal{U}}, \mathsf{gc.vk}) \leftarrow \mathsf{GC}.\mathcal{G}arble(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$, and sends $(\widetilde{\mathcal{U}}, \{L_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{B}$.
   - If $b = 1$, the challenger computes $(\widetilde{\mathcal{U}}, \mathsf{gc.vk}) \leftarrow \mathsf{GC}.\mathcal{S}im(1^\lambda, 1^{|f|}, U(f, m), \{L_{i,f[i]}\}_{i\in[s]})$, and sends $(\widetilde{\mathcal{U}}, \{L_{i,f[i]}\}_{i\in[s]})$ to $\mathcal{B}$.

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

7. $\mathcal{B}$ sends $(\widetilde{\mathcal{U}}, \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[s],\alpha\in\{0,1\}})$ 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 $\bot$ from the challenger. If $\mathcal{B}$ receives $\bot$ from the challenger, it outputs $\bot$ 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\mathsf{Hyb}_1 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\mathsf{Hyb}_2 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B} \mid b = 0] - \Pr[1 \leftarrow \mathcal{B} \mid b = 1]|$ is non-negligible, which contradicts the selective certified everlasting security of Σ_{cegc}. $\square$

4.3 1-Bounded Construction with Adaptive Security

In this section, we convert the non-adaptive scheme constructed in the previous subsection to the adaptive one by using a certified everlasting secure RNC scheme (Definition D.1). See Appendix D.2 for how to achieve this building block.

Our 1-bounded adaptive certified everlasting secure FE scheme. We construct a 1-bounded adaptive certified everlasting secure FE scheme $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ from a 1-bounded non-adaptive certified everlasting secure FE scheme $\Sigma_{\mathsf{nad}} = \mathsf{NAD}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$, where the ciphertext space is $\mathcal{C} := \mathcal{Q}^{\otimes n}$, and a certified everlasting secure RNCE scheme $\Sigma_{\mathsf{cence}} = \mathsf{NCE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{F}ake, \mathsf{Reveal}, \mathcal{D}el, \mathsf{Vrfy})$ (Definition D.1). Let $\mathsf{NAD.Recover}$ be a QPT algorithm such that

$$\Pr\left[\mathsf{NAD.Vrfy}(\mathsf{nad.vk}, \mathsf{nad.cert}^*) \neq \top \;\middle|\; \begin{array}{l} (\mathsf{nad.MPK}, \mathsf{nad.MSK}) \leftarrow \mathsf{NAD.Setup}(1^\lambda) \\ (\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{E}nc(\mathsf{nad.MPK}, m) \\ a, c \leftarrow \{0,1\}^n \\ \mathsf{nad}.\widetilde{\mathsf{cert}} \leftarrow \mathsf{NAD}.\mathcal{D}el(Z^c X^a \mathsf{nad}.ct X^a Z^c) \\ \mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(a, c, \mathsf{nad}.\widetilde{\mathsf{cert}}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

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 $\mathsf{sk}_f := (\mathsf{nad.sk}_f, \mathsf{nce.sk})$.

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

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

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

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

$\mathcal{Del}(ct)$**:**

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

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

- Parse $\mathsf{vk} = (\mathsf{nad.vk}, \mathsf{nce.vk}, a, c)$ and $\mathsf{cert} = (\mathsf{nad}.\widetilde{\mathsf{cert}}, \mathsf{nce.cert})$.
- Compute $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(a, c, \mathsf{nad}.\widetilde{\mathsf{cert}})$.
- 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 4.11. *If Σ_{nad} satisfies the 1-bounded non-adaptive simulation-security (Definition 4.5) and Σ_{cence} satisfies the RNC security (Definition D.3), Σ_{cefe} satisfies the 1-bounded adaptive simulation-security (Definition 4.6).*

Its proof is similar to that of Theorem 4.12, and therefore we omit it.

Theorem 4.12. *If Σ_{nad} satisfies the 1-bounded non-adaptive certified everlasting simulation-security(Definition 4.4) and Σ_{cence} satisfies the certified everlasting RNC security (Definition D.4), Σ_{cefe} satisfies the 1-bounded adaptive certified everlasting simulation-security (Definition 4.2).*

Proof of Theorem 4.12. 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 $\mathsf{NAD}.\mathcal{Sim}$ be the simulating algorithm of the ciphertext $\mathsf{nad}.ct$. Let us describe how the simulator $\mathcal{Sim} = (\mathcal{Sim}_1, \mathcal{Sim}_2, \mathcal{Sim}_3)$ works below.

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

1. Parse $\mathsf{MPK} = (\mathsf{nad.MPK}, \mathsf{nce.pk})$ and $\mathcal{V} = (f, f(m), (\mathsf{nad.sk}_f, \mathsf{nce.sk}))$ *or* $\emptyset$. [18]
2. $\mathcal{Sim}_1$ does the following:
   - If $\mathcal{V} = \emptyset$, generate $\left|\widetilde{0^n 0^n}\right\rangle$ and $(\mathsf{nce.vk}, \widetilde{\mathsf{nce}.ct}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE}.\mathcal{Fake}(\mathsf{nce.pk})$. 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 $ct := (\Psi_A, \widetilde{\mathsf{nce}.ct})$ 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 $a, c \leftarrow \{0,1\}^n$, $(\mathsf{nce.vk}, \mathsf{nce}.ct) \leftarrow \mathsf{NCE}.\mathcal{Enc}(\mathsf{nce.pk}, (a, c))$, $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Sim}(\mathsf{nad.MPK}, (f, f(m), \mathsf{nad.sk}_f), 1^{|m|})$ and $\Psi := Z^c X^a \mathsf{nad}.ct X^a Z^c$. Output $ct := (\Psi, \mathsf{nce}.ct)$ and $\mathsf{st} := (\mathsf{nad.vk}, \mathsf{nce.vk}, a, c, 1)$.

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

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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Sim}(\mathsf{nad.MPK}, (f, f(m), \mathsf{nad.sk}_f), 1^{|m|})$. 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 $\mathsf{st}' := (\mathsf{nad.vk}, \mathsf{nce.vk}, x, z, 1)$.

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

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. $\mathcal{Sim}_3$ does the following:
   - If the final bit of st^* is 0, compute $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Sim}(\mathsf{nad.MPK}, \emptyset, 1^{|m|})$. 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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(0)$.

[18]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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$, $\Psi := Z^c X^a \mathsf{nad}.ct X^a Z^c$ and $(\mathsf{nce.vk}, \mathsf{nce}.ct) \leftarrow \mathsf{NCE}.\mathcal{Enc}(\mathsf{nce.pk}, (a, c))$, 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 $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(a, c, \mathsf{nad.cert})$. 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 $(\mathsf{nce.vk}, \widetilde{\mathsf{nce}.ct}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE}.\mathcal{Fake}(\mathsf{nce.pk})$ and sends $(\Psi, \widetilde{\mathsf{nce}.ct})$ to $\mathcal{A}_1$ instead of computing $(\mathsf{nce.vk}, \mathsf{nce}.ct) \leftarrow \mathsf{NCE}.\mathcal{Enc}(\mathsf{nce.pk}, (a, c))$ 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 $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(x, z, \mathsf{nad.cert})$ instead of computing $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(a, c, \mathsf{nad.cert})$ 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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$ 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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$ 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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$, 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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Sim}(\mathsf{nad.MPK}, \mathcal{V}, 1^{|m|})$ instead of computing $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$, 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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}}, \mathcal{A}}(\lambda, b)$ and $\mathcal{Sim} = (\mathcal{Sim}_1, \mathcal{Sim}_2, \mathcal{Sim}_3)$, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}}, \mathcal{A}}(\lambda, 0) = 1\right]$ and $\Pr[\mathsf{Hyb}_4 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}}, \mathcal{A}}(\lambda, 1) = 1\right]$. Therefore, Theorem 4.12 easily follows from Propositions 4.13 to 4.16. (Whose proof is given later.) ☐

Proposition 4.13. *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 4.14.

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

Proposition 4.15.

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

Proposition 4.16. *If* Σ_{nad} *is* 1*-bounded non-adaptive certified everlasting simulation-secure, it holds that*

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

Proof of Proposition 4.13. 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 $\mathsf{Exp}_{\Sigma_{\mathsf{cence}},\mathcal{B}}^{\mathsf{cert\text{-}ever\text{-}rec\text{-}nc}}(\lambda, b)$, 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 $(\mathsf{nad.vk}, \mathsf{nad}.ct) \leftarrow \mathsf{NAD}.\mathcal{Enc}(\mathsf{nad.MPK}, m)$, 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 $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(a, c, \mathsf{nad.cert})$ 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible, and therefore $|\Pr[1 \leftarrow \mathcal{B} \mid b = 0] - \Pr[1 \leftarrow \mathcal{B} \mid b = 1]|$ is non-negligible, which contradicts the certified everlasting RNC security of Σ_{cence}. □

Proof of Proposition 4.14. 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 4.15. The difference between Hyb_2 and Hyb_3 is only the order of operating the algorithm $\mathsf{NAD}.\mathcal{Enc}$ 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 4.16. 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 non-adaptive certified everlasting simulation-security of Σ_{nad}. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives nad.MPK from the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}noada\text{-}sim}}_{\Sigma_{\mathsf{nad}},\mathcal{B}}(\lambda, b)$, 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 $\mathsf{nad}.ct$ from the challenger, generates $a, c \leftarrow \{0,1\}^n$, $\Psi := Z^c X^a \mathsf{nad}.ct X^a Z^c$ and $(\mathsf{nce.vk}, \mathsf{nce}.ct) \leftarrow \mathsf{NCE}.\mathcal{Enc}(\mathsf{nce.pk}, (a, c))$, and sends $(\Psi, \mathsf{nce}.ct)$ to $\mathcal{A}_1$.
   - If a key query is not called in step 2, $\mathcal{B}$ generates $\left|\widetilde{0^n 0^n}\right\rangle$. Let $\Psi_A := \mathrm{Tr}_B(\left|\widetilde{0^n 0^n}\right\rangle\left\langle\widetilde{0^n 0^n}\right|)$ and $\Psi_B := \mathrm{Tr}_A(\left|\widetilde{0^n 0^n}\right\rangle\left\langle\widetilde{0^n 0^n}\right|)$. $\mathcal{B}$ computes $(\mathsf{nce.vk}, \widetilde{\mathsf{nce}.ct}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE}.\mathcal{Fake}(\mathsf{nce.pk})$ 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 $\mathsf{nad}.ct$, measures 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 $i \in [n]$. $\mathcal{B}$ 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 $\mathsf{nad}.ct$, and measures 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 $i \in [n]$.

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 $\mathsf{nad.cert}^* \leftarrow \mathsf{NAD.Recover}(x^*, z^*, \mathsf{nad.cert})$, 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 $\perp$ 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 $\perp$, sends $\perp$ to $\mathcal{A}_2$, and aborts.

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

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

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

4.4 q-Bounded Construction with Adaptive Security for NC^1 circuits

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

Our q-bounded adaptive certified everlasting secure FE scheme for NC^1 circuits. We consider the polynomial representation of circuits C in NC^1. The input message space is $\mathcal{M} := \mathbb{F}^\ell$, and for each 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 $\mathcal{G} := \{G_{C,\Delta}\}_{C\in\mathsf{NC}^1,\Delta\subseteq[S]}$, 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 secure FE scheme for all NC^1 circuits $\Sigma_{\mathsf{cefe}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ from a 1-bounded certified everlasting secure FE scheme $\Sigma_{\mathsf{one}} = \mathsf{ONE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$.

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

- 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 $\mathsf{sk}_C := (\Gamma, \Delta, \{\mathsf{one.sk}_{C,\Delta,i}\}_{i\in\Gamma})$.

$\mathcal{E}nc(\mathsf{MPK}, x)$**:**

- 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 $(\mathsf{one.vk}_i, \mathsf{one.}ct_i) \leftarrow \mathsf{ONE.}\mathcal{E}nc(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)))$.
- Output $\mathsf{vk} = \{\mathsf{one.vk}_i\}_{i\in[N]}$ and $ct := \{\mathsf{one.}ct_i\}_{i\in[N]}$.

$\mathcal{D}ec(\mathsf{sk}_C, ct)$**:**

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

$\mathcal{D}el(ct)$**:**

- Parse $ct = \{\mathsf{one.}ct_i\}_{i\in[N]}$.
- For $i \in [N]$, compute $\mathsf{one.cert}_i \leftarrow \mathsf{ONE.}\mathcal{D}el(\mathsf{one.}ct_i)$.
- 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 4.17. *If Σ_{one} satisfies the 1-bounded adaptive simulation-security, Σ_{cefe} satisfies the q-bounded adaptive simulation-security.*

Its proof is similar to that of Theorem 4.18, and therefore we omit it.

Theorem 4.18. *If Σ_{one} satisfies the 1-bounded adaptive certified everlasting simulation-security, Σ_{cefe} the q-bounded adaptive certified everlasting simulation-security.*

Proof of Theorem 4.18. Let us denote the simulating algorithm of Σ_{one} as $\mathsf{ONE}.\mathcal{S}im = \mathsf{ONE}.(\mathcal{S}im_1, \mathcal{S}im_2, \mathcal{S}im_3)$. Let us describe how the simulator $\mathcal{S}im = (\mathcal{S}im_1, \mathcal{S}im_2, \mathcal{S}im_3)$ works below.

$\mathcal{S}im_1(\mathsf{MPK}, \mathcal{V}, 1^{|x|})$**:** 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 $\mathcal{L} := \bigcup_{i\neq i'}(\Gamma_i \cap \Gamma_{i'})$. $\mathcal{S}im_1$ 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. $\mathcal{S}im_1$ uniformly and independently samples ℓ random degree t polynomials $\mu_1, \cdots, \mu_\ell$ whose constant terms are all 0.
4. $\mathcal{S}im_1$ 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 $j \leq q^*$, 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 $i \in \mathcal{L}$, $\mathcal{S}im_1$ computes

$$(\mathsf{one.vk}_i, \mathsf{one}.ct_i) \leftarrow \mathsf{ONE}.\mathcal{E}nc(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i))).$$

6. For each $i \notin \mathcal{L}$, $\mathcal{S}im_1$ does the following:

- If $i \in \Gamma_j$ for some $j \in [q^*]$ [19], computes

$$(\mathsf{one}.ct_i, \mathsf{one.st}_i) \leftarrow \mathsf{ONE}.\mathcal{Sim}_1(\mathsf{one.MPK}_i, (G_{C_j,\Delta_j,i}, \eta_j(i), \mathsf{one.sk}_{C_j,\Delta_j,i}), 1^{|m|}).$$

- If $i \notin \Gamma_j$ for all $j \in [q^*]$, computes

$$(\mathsf{one}.ct_i, \mathsf{one.st}_i) \leftarrow \mathsf{ONE}.\mathcal{Sim}_1(\mathsf{one.MPK}_i, \emptyset, 1^{|m|}).$$

7. Output $ct := \{\mathsf{one}.ct_i\}_{i\in[N]}$ and $\mathsf{st} := (\{\Gamma_i\}_{i\in[q]}, \{\Delta_i\}_{i\in[q]}, \{\eta_j(i)\}_{j\in\{q^*+1,\cdots,q\}, i\in\mathcal{L}}, \{\mathsf{one.st}_i\}_{i\in[N]\setminus\mathcal{L}}, \{\mathsf{one.vk}_i\}_{i\in\mathcal{L}})$.

$\mathcal{Sim}_2(\mathsf{MSK}, C_j, C_j(x), \mathsf{st})$**:** 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

$$(\mathsf{one.sk}_{C_j,\Delta_j,i}, \mathsf{one.st}_i^*) \leftarrow \mathsf{ONE}.\mathcal{Sim}_2(\mathsf{one.MSK}_i, \eta_j(i), G_{C_j,\Delta_j}, \mathsf{one.st}_i).$$

   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 $\mathsf{st}_j := (\{\Gamma_i\}_{i\in[q]}, \{\Delta_i\}_{i\in[q]}, \{\eta_j(i)\}_{j\in\{q^*+1,\cdots,q\}, i\in\mathcal{L}}, \{\mathsf{one.st}_i\}_{i\in[N]\setminus\mathcal{L}}, \{\mathsf{one.vk}_i\}_{i\in\mathcal{L}})$.

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

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 $\mathsf{one.vk}_i \leftarrow \mathsf{ONE}.\mathcal{Sim}_3(\mathsf{one.st}_i)$.
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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0)$.

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 $(\mathsf{one.vk}_i, \mathsf{one}.ct_i) \leftarrow \mathsf{ONE}.\mathcal{Enc}(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)))$, 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 $\mathcal{A}_1$.

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

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 $\mathcal{S}im_1$ 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 $\mathcal{S}im = (\mathcal{S}im_1, \mathcal{S}im_2, \mathcal{S}im_3)$ 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 $\mathcal{S}im_1$ described above.

From the definition of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, b)$ and $\mathcal{S}im = (\mathcal{S}im_1, \mathcal{S}im_2, \mathcal{S}im_3)$, it is clear that $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 0) = 1\right]$ and $\Pr[\mathsf{Hyb}_4 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{cefe}},\mathcal{A}}(\lambda, 1) = 1\right]$. Therefore, Theorem 4.18 easily follows from Propositions 4.19 to 4.22 (whose proofs are given later). ☐

Proposition 4.19. $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}(\lambda)$.

Proposition 4.20. $\Pr[\mathsf{Hyb}_1 = 1] = \Pr[\mathsf{Hyb}_2 = 1]$.

Proposition 4.21. *If Σ_{one} is 1-bounded adaptive certified everlasting simulation-secure,*

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

Proposition 4.22. $\Pr[\mathsf{Hyb}_3 = 1] = \Pr[\mathsf{Hyb}_4 = 1]$.

Proof of Proposition 4.19. Let 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 Hyb_0' except that it outputs $\bot$ 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 4.23 and 4.24 shows that $\Pr[|\mathcal{L}| > t] \leq 2^{-\Omega(\lambda)}$ and $\Pr[\mathsf{Collide}] \leq q2^{-\Omega(\lambda)}$, which completes the proof. ☐

Lemma 4.23 ([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 4.24 ([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* $\Delta_i \backslash (\bigcup_{j \neq i} \Delta_j) = \emptyset$*. Then, we have*

$$\Pr[\mathsf{Collide}] \leq q 2^{-\Omega(\lambda)}$$

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

Proof of Proposition 4.20. In the encryption in Hyb_1, ξ_{a^*} is chosen at random and $\eta_j(\cdot)$ is defined by the relation. $\mathcal{Sim}$ 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 4.21. To prove the proposition, let us define a hybrid experiment $\mathsf{Hyb}_2 s$ 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 $\{\mathsf{one}.ct_i\}_{i \in [s] \backslash \mathcal{L}}$ as in the simulator $\mathcal{Sim}_1$. 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 $\mathcal{Sim}_2$, where q' is the number of key queries that the adversary makes in total. Third, the challenger generates $\{\mathsf{one.vk}_i\}_{i \in [s] \backslash \mathcal{L}}$ as in the simulator $\mathcal{Sim}_3$.

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 4.21.

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 adaptive certified everlasting simulation-security of Σ_{one} as follows.

1. $\mathcal{B}$ receives $\mathsf{one.MPK}$ from the challenger of $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}ada\text{-}sim}}_{\Sigma_{\mathsf{one}}, \mathcal{A}}(\lambda, b)$. $\mathcal{B}$ 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 $\mathcal{Sim}_1$. For $i \in [s-1] \setminus \mathcal{L}$, $\mathcal{B}$ generates $\mathsf{one}.ct_i$ as in the simulator $\mathcal{Sim}_1$. For $i \in \{s+1, \cdots N\} \cup \mathcal{L}$, $\mathcal{B}$ generates $(\mathsf{one.vk}_i, \mathsf{one}.ct_i) \leftarrow \mathsf{ONE}.\mathcal{Enc}(\mathsf{one.MPK}_i, (\mu_1(i), \cdots, \mu_\ell(i), \xi_1(i), \cdots, \xi_S(i)))$. $\mathcal{B}$ 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 $\mathcal{S}im_2$. 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 $\mathcal{S}im_3$ [20].
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'.

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

Proof of Proposition 4.22. 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 4.22 follows from these observations. □

4.5 Discussion on q-Bounded Consturction for All Circuits

We discuss technical hurdles to achieve certified everlasting secure bounded collusion-resistant FE for $\mathsf{P/poly}$ from standard PKE.

Gorbunov, Vaikuntanathan, and Wee [GVW12] presented a conversion from FE for NC^1 to FE for $\mathsf{P/poly}$ by using randomized encoding or FHE. However, we cannot directly apply their techniques in the certified everlasting setting. When we use randomized encoding, we use a functional decryption key for circuit G_f that takes m as an input and outputs a randomized encoding $\widetilde{f(m)}$.[21] That is, we can obtain $\widetilde{f(m)}$ (and $f(m)$ via a decoding algorithm) from the functional decryption key and ciphertext of m since randomized encoding is computable in a constant-depth circuit [AIK06].

The first problem is that even if we use *certified everlasting secure* FE for NC^1, information about m remains in $\widetilde{f(m)}$ since the decryption result does not directly provide $f(m)$. More specifically, adversaries can keep $\widetilde{f(m)}$ (this is classical information) before deletion and an unbounded adversary could recover m from $\widetilde{f(m)}$ even after $\mathsf{Enc}(m)$ was deleted.

The second problem is that we cannot use certified everlasting secure randomized encoding to solve the first problem since we use FE for *classical* circuits here. In certified everlasting secure randomized encoding, $\widetilde{f(m)}$ must be quantum

[20] For $i \in \{s+1, \cdots N\} \cup \mathcal{L}$, $\mathcal{B}$ generated $\mathsf{one.vk}_i$ in step 5.
[21] For simplicity, we ignore how to set randomness for randomized encoding here since it is not an essential issue.

state, which cannot be supported by FE for classical circuits. We do not have certified everlasting secure FE that supports quantum circuits computing quantum state. Moreover, we do not know how to achieve certified everlasting secure randomized encoding. Thus, the approach using randomized encoding does not work.

The approach using FHE also has problems. In this approach, we consider a functional decryption key for circuit G_f that takes an FHE ciphertext $\mathsf{fhe.ct}$ and an FHE decryption key $\mathsf{fhe.sk}$ and outputs $\mathsf{fhe.ct}$ and $\mathsf{FHE.Dec}(\mathsf{fhe.sk}, \mathsf{FHE.Eval}(f, \mathsf{fhe.ct}))$. Here, we must output $\mathsf{fhe.ct}$ as the public part because we use FE for NC^1 and need to apply $f \in \mathsf{P/poly}$ by $\mathsf{FHE.Eval}$ in the public part (though the FHE decryption part is in NC^1).[22] That is, the FHE part must be also certified everlasting secure.

First, we cannot use certified everlasting secure FHE in a black-box way. We need to encrypt an FHE ciphertext by FE for NC^1 in this approach. However, if FHE is certified everlasting secure, a ciphertext is quantum state, which is not supported by our certified everlasting secure FE for NC^1.

Second, even if we use certified everlasting secure FHE in a non-black-box way like our compute-and-compare obfuscation construction in Section 5.2 (by separating the classical FHE part from the BB84 state), the approach does not work due to the following reason. To achieve certified everlasting security, $\mathsf{fhe.ct}$ is an encryption of $m \oplus \bigoplus_i \theta_i$ where θ is a basis choice as in Section 5.2. To unmask $\bigoplus_i \theta_i$, we need to coherently apply f to $\mathsf{fhe.ct}$ and BB84 state as the certified everlasting secure FHE by Bartusek and Khurana [BK23]. However, we cannot execute the coherent evaluation in the FE decryption mechanism (cannot take BB84 state as input). Hence, we obtain $f(m \oplus \bigoplus_i \theta_i)$ and the correctness does not hold. Thus, the approach using FHE does not work too.

Another plausible (but failed) approach is using the framework by Ananth and Vaikuntanathan [AV19]. They constructed bounded collusion-resistant FE for $\mathsf{P/poly}$ *without the bootstrapping method* by Gorbunov et al. [GVW12]. However, their construction heavily relies on a secure multi-party computation protocol based on *PRG*. It is hard to define certified everlasting security for PRG because there is nothing to delete. Thus, it is unclear how to use their framework in the certified everlasting setting.

Therefore, previous approaches for converting FE for NC^1 to FE for $\mathsf{P/poly}$ do not work in the certified everlasting setting.

5 Compute-and-Compare Obfuscation with Certified Everlasting Deletion

5.1 Definition

In this section, we introduce the notion of compute-and-compare obfuscation with certified everlasting security.

Definition 5.1 (Compute-and-Compare Obfuscation with Certified Everlasting Deletion (Syntax)). *A compute-and-compare obfuscation with certified everlasting deletion is a tuple of algorithms $(\mathcal{CCObf}, \mathcal{Del}, \mathsf{Vrfy})$ for the family of distributions $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$ and message space $\mathcal{M}$.*

$\mathcal{CCObf}(1^\lambda, P, \mathsf{lock}, m)$**:** *The obfuscation algorithm takes as input a security parameter 1^λ, a circuit P, a lock string $\mathsf{lock} \in \{0,1\}^{p(\lambda)}$ and a message $m \in \mathcal{M}$, and outputs an obfuscated circuit $\widetilde{\mathcal{P}}$ and a verification key* vk.

$\mathcal{Del}(\widetilde{\mathcal{P}}) \to \mathsf{cert}$**:** *The deletion algorithm takes as input an obfuscated circuit $\widetilde{\mathcal{P}}$ and outputs a classical certificate* cert.

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

Definition 5.2 (Correctness of Compute-and-Compare Obfuscation with Certified Everlasting Deletion). *The correctness of compute-and-compare obfuscation with certified everlasting deletion for the family of distributions $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$ and message space $\mathcal{M}$ is defined as follows.*

Functionality Preserving: *There exists a negligible function* negl *such that for all circuit P, all lock value* lock*, and all message $m \in \mathcal{M}$, it holds that*

$$\Pr\left[\forall x, \widetilde{\mathcal{P}}(x) = \mathbf{CC}[P, \mathsf{lock}, m](x) \mid \widetilde{\mathcal{P}} \leftarrow \mathcal{CCObf}(1^\lambda, P, \mathsf{lock}, m)\right] \geq 1 - \mathsf{negl}(\lambda).$$

[22] See [GVW12] for the detail.

Verification Correctness: *There exists a negligible function* negl *such that for all circuit* P*, all lock value* lock*, and all message* $m \in \mathcal{M}$*, it holds that*

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk},\mathsf{cert}) \neq \top \;\middle|\; \begin{array}{l}(\widetilde{\mathcal{P}},\mathsf{vk}) \leftarrow \mathcal{CCObf}(1^\lambda, P, \mathsf{lock}, m)\\ \mathsf{cert} \leftarrow \mathcal{Del}(\widetilde{\mathcal{P}})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 5.3 (Certified Everlasting Security of Compute-and-Compare Obfuscation). *Let* $\Sigma_{\mathsf{CCO}} = (\mathcal{CCObf}, \mathcal{Del}, \mathsf{Vrfy})$ *be a compute-and-compare obfuscation with certified everlasting deletion for the family of distributions* $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$ *and a message space* $\mathcal{M}$*. We consider experiments* $\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, b)$ *and* $\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, b)$ *played between a challenger and a non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$*. Let* $\mathcal{Sim}$ *be a QPT algorithm. The experiments are defined as follows:*

1. $\mathcal{A}_\lambda(|\psi\rangle_\lambda)$ *submits a message* $m \in \mathcal{M}$ *to the challenger.*
2. *The challenger chooses* $(P, \mathsf{lock}, \mathit{aux}) \leftarrow D_{\mathsf{param}}$.
3. *The challenger computes* $(\widetilde{\mathcal{P}}^{(0)}, \mathsf{vk}^{(0)}) \leftarrow \mathcal{CCObf}(1^\lambda, P, \mathsf{lock}, m)$ *or* $(\widetilde{\mathcal{P}}^{(1)}, \mathsf{vk}^{(1)}) \leftarrow \mathcal{Sim}(1^\lambda, \mathsf{pp}_P, 1^{|m|})$ *and sends* $(\widetilde{\mathcal{P}}^{(b)}, \mathit{aux})$ *to* $\mathcal{A}_\lambda$ *according to the bit b. Recall that a program P has an associated set of parameters* pp_P *(input size, output size, circuit size) which we do not need to hide.*
4. $\mathcal{A}_\lambda$ *submits a certificate of deletion* cert *and its internal state* ρ *to the challenger.*
5. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}^{(b)}, \mathsf{cert})$*. If the outcome is* $\top$*, the experiment* $\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, b)$ *outputs* ρ*; otherwise if the outcome is* $\bot$ *then* $\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, b)$ *outputs* $\bot$ *and ends.*
6. *The challenger sends the outcome of* $\mathsf{Vrfy}(\mathsf{vk}^{(b)}, \mathsf{cert})$ *to* $\mathcal{A}_\lambda$.
7. $\mathcal{A}_\lambda$ *outputs its guess* $b' \in \{0,1\}$ *which is the output of the experiment* $\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, b)$.

We say that the Σ_{CCO} *is certified everlasting secure if for any non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$*, it holds that*

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, 0), \mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, 1)) \leq \mathsf{negl}(\lambda),$$

and

$$\left|\Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CCO}},\mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

5.2 Construction

In this section, we construct a compute-and-compare obfuscation with certified everlasting deletion from classical compute-and-compare obfuscation and FHE.

Ingredients. We use the following building blocks.

1. $\Sigma_{\mathsf{fhe}} = \mathsf{FHE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a classical FHE scheme.
2. $\Sigma_{\mathsf{CCO}} = \mathsf{CC.Obf}$ be a classical compute-and-compare obfuscation scheme.

Certified everlasting compute-and-compare obfuscation for multi-bit message. We construct $\Sigma_{\mathsf{CECCO}} = (\mathcal{CCObf}, \mathcal{Del}, \mathsf{Vrfy})$ for the family of distribution $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$ and message space $\mathcal{M}$. We let the message space $\mathcal{M} := \{0,1\}^n$.

$\mathcal{CCObf}(1^\lambda, P, \mathsf{lock}, m)$:

1. Sample $R \leftarrow \{0,1\}^\lambda$.
2. Sample $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$.
3. Compute $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$ where $\mathsf{fDec}(\cdot) = \mathsf{FHE.Dec}(\mathsf{fsk}, \cdot)$.
4. Compute $\widetilde{I} \leftarrow \mathsf{CC.Obf}(1^\lambda, I, \mathsf{lock}, R)$ where $I(X) = X$ for every X.
5. Represent $(P \| \widetilde{I}) = (b_1, \ldots, b_\ell) \in \{0,1\}^\ell$.
6. Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$.
7. Set $\widetilde{b_i} := b_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
8. Denote $m = (m_1, \ldots, m_n) \in \{0,1\}^n$.
9. Sample $\boldsymbol{\theta}_{\ell+k}, \boldsymbol{z}_{\ell+k} \leftarrow \{0,1\}^\lambda$ for all $k \in [n]$.
10. Set $\widetilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$.
11. Compute $\mathsf{fct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\boldsymbol{\theta}_i, \widetilde{b}_i))$ for all $i \in [\ell+n]$.
12. Output $\widetilde{\mathcal{P}} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell+n]}, \mathsf{fpk})$ and $\mathsf{vk} := (\{\boldsymbol{z}_i, \boldsymbol{\theta}_i\}_{i\in[\ell+n]})$.

How to evaluate $\widetilde{\mathcal{P}}(x)$:

1. Parse $\widetilde{\mathcal{P}} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell+n]}, \mathsf{fpk})$.
2. Define a circuit $\widehat{U}_x$ as in Figure 2.
3. To compute an evaluated ciphertext for $\mathsf{FHE.Eval}(\mathsf{fpk}, \widehat{U}_x, \cdot)$, apply $\widehat{U}_x$ homomorphically in superposition with the input $(\{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell]}, (|\boldsymbol{z}_{\ell+k}\rangle_{\boldsymbol{\theta}_{\ell+k}}, \mathsf{fct}_{\ell+k}))$ and obtain a ciphertext $|\mathsf{fct}_{\ell+k,P}\rangle$ for each $k \in [n]$.
4. Apply $\widetilde{\mathsf{fDec}}(\cdot)$ in superposition with the input $|\mathsf{fct}_{\ell+k,P}\rangle$ and measure the output register in the standard basis to get a classical outcome β_k for each $k \in [n]$.
5. Set $m_k = 1$ if $\beta_k = 1$, else set $m_k = 0$, for each $k \in [n]$.
6. Output $m = (m_1, \ldots, m_n)$.

$\mathcal{Del}(\widetilde{\mathcal{P}})$:

1. Parse $\widetilde{\mathcal{P}} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell+n]}, \mathsf{fpk})$.
2. Measure $|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}$ in the Hadamard basis for all $i \in [\ell+n]$, and obtain $(\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_{\ell+n})$.
3. Output $\mathsf{cert} := (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_{\ell+n})$.

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

1. Parse $\mathsf{vk} = (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell+n]})$ and $\mathsf{cert} = (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_{\ell+n})$.
2. If $\Big((z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)\Big)$ holds for all $i \in [\ell+n]$ and $j \in [\lambda]$, then output $\top$; otherwise output $\bot$.

Circuit $\widehat{U}_x$

Hardwire: x
Input: $(\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i, \widetilde{b}_i)\}_{i\in[\ell]}, (\boldsymbol{z}_{\ell+k}, \boldsymbol{\theta}_{\ell+k}, \widetilde{b}_{\ell+k}))$

1. Compute $b_i := \widetilde{b}_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
2. Reconstruct $(C \| \widetilde{I})$ from $(b_1, \ldots, b_\ell)$.
3. Compute $m_k := \widetilde{b}_{\ell+k} \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$.
4. Output $m_k \cdot \widetilde{I}(C(x))$

Figure 2: The description of the circuit $\widehat{U}_x$

Security. We use Lemma 3.5 by Bartusek and Khurana [BK23] to prove the security of our construction.

Theorem 5.4. *If Σ_{CCO} is a secure compute-and-compare obfuscation and Σ_{fhe} is an IND-CPA secure fully homomorphic encryption then Σ_{CECCO} is a certified everlasting secure compute-and-compare obfuscation scheme for the family of distribution $D = \{D_{\mathsf{param}}\}_{\mathsf{param}}$.*

Proof of Theorem 5.4. We first describe the simulator of $\mathcal{CCObf}$, denoted as $\mathcal{Sim}$, before we go to the formal security analysis. Let CCO.Sim be the simulator the classical compute-and-compare obfuscation employed as a building block in the above construction. For $(P, \mathsf{lock}, \mathcal{aux}) \leftarrow D_{\mathsf{param}}$, the algorithm $\mathcal{Sim}$ works as follows:

$\mathcal{Sim}(1^\lambda, \mathsf{pp}_P, 1^n)$**:**

1. Sample $R \leftarrow \{0,1\}^\lambda$.
2. Sample $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$.
3. Compute $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$.
4. Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell+n]$.
5. Set $\widetilde{b}_i := 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell+n]$.
6. Compute $\mathsf{fct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\boldsymbol{\theta}_i, \widetilde{b}_i))$ for all $i \in [\ell+n]$.
7. Output $\widetilde{\mathcal{P}} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell+n]}, \mathsf{fpk})$ and $\mathsf{vk} := (\{\boldsymbol{z}_i, \boldsymbol{\theta}_i\}_{i\in[\ell+n]})$.

Note that $\mathcal{Sim}$ does not need information about $(P, \mathsf{lock}, \mathcal{aux})$ except pp_P. We show that

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda,0), \mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda,1)) \leq \mathsf{negl}(\lambda).$$

using Lemma 3.5 and the post-quantum security of Σ_{CCO} and Σ_{fhe}. We use the following sequence of hybrids to prove this.

Hyb_0**:** This is the same as $\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda,0)$. Let $\widetilde{\mathcal{P}}^{(0)} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i\in[\ell+n]}, \mathsf{fpk})$ be the compute-and-compare obfuscated circuit computed using the honest $\mathcal{CCObf}$ algorithm.

Hyb_1**:** This hybrid works as follows:

1. $\mathcal{A}$ submits a message $m \in \{0,1\}^n$ to the challenger.
2. The challenger chooses $(P, \mathsf{lock}, \mathcal{aux}) \leftarrow D_{\mathsf{param}}$.
3. The challenger computes the obfuscated circuit as follows:

(a) Sample $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and $R \leftarrow \{0,1\}^\lambda$.
(b) Compute $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$.
(c) Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell + n]$.
(d) Set $\widetilde{b}_i := 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for $i \in [\ell]$.
(e) Set $\widetilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$.
(f) Compute $\mathsf{fct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\boldsymbol{\theta}_i, \widetilde{b}_i))$ for all $i \in [\ell + n]$.
(g) Set $\widetilde{\mathcal{P}} := (\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i \in [\ell+n]}, \mathsf{fpk})$.

The challenge sends $\widetilde{\mathcal{P}}$ to $\mathcal{A}$.

4. $\mathcal{A}$ sends a deletion certificate $\mathsf{cert} := (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_{\ell+n})$ and its internal state ρ to the challenger.
5. The challenger checks if $\Big((z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)\Big)$ holds for all $i \in [\ell + n]$ and $j \in [\lambda]$. If the check fails, the experiment halts and returns $\perp$; otherwise, go to the next step.
6. The experiment outputs ρ as a final output.

Note that, the FHE ciphertexts $\{\mathsf{fct}_i\}_{i \in [\ell]}$ contain no information about the lock value lock, the random string R and the circuit P. To prove the indistinguishability between Hyb_0 and Hyb_1, we consider a sequence of intermediate hybrids $\mathsf{Hyb}_{1,k}$ for $k \in [0, \ell]$ where $\mathsf{Hyb}_{1,0}$ is identical to Hyb_0 and the only difference between $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ is that fct_k is an encryption of $(\boldsymbol{\theta}_k, b_k \oplus \bigoplus_{j:\theta_{k,j}=0} z_{k,j})$ where b_k in $\mathsf{Hyb}_{1,k-1}$ is the same as b_k in Hyb_0 and b_k in $\mathsf{Hyb}_{1,k}$ is set to zero for $k \in [\ell]$.

Now, we consider a sequence of experiments $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \boldsymbol{\theta}, \beta)$ for $k \in [\ell]$ between a QPT adversary $\mathcal{B}$ and a challenger $\mathcal{C}$ for $\boldsymbol{\theta} \in \{0,1\}^\lambda$ and $\beta \in \{0,1\}$. The experiment $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \boldsymbol{\theta}, \beta)$ is basically the same as $\mathsf{Hyb}_{1,k}$ where we take $\boldsymbol{\theta}_k = \boldsymbol{\theta}$, $\widetilde{b}_k = \beta$ and $\mathcal{B}$ plays the role of $\mathcal{A}$, $\mathcal{C}$ plays the role of the challenger. In particular, it works as follows:

$\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \boldsymbol{\theta}, \beta)$:

1. $\mathcal{B}$ submits a message $m \in \mathcal{M}$ to the challenger.
2. The challenger chooses $(P, \mathsf{lock}, aux) \leftarrow D_{\mathsf{param}}$.
3. The challenger computes the obfuscated circuit as follows:
   (a) Sample $(\mathsf{fpk}, \mathsf{fsk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$ and $R \leftarrow \{0,1\}^\lambda$.
   (b) Compute $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$.
   (c) Compute $\widetilde{I} \leftarrow \mathsf{CC.Obf}(1^\lambda, I, \mathsf{lock}, R)$.
   (d) Represent $(P \| \widetilde{I}) = (b_1, \ldots, b_\ell) \in \{0,1\}^\ell$.
   (e) Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell + n] \setminus \{k\}$.
   (f) Set $\widetilde{b}_i$ for $i \in [\ell]$ as follows:

$$\widetilde{b}_i := \begin{cases} 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [1, k-1] \\ \beta & \text{if } i = k \\ b_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [k+1, \ell] \end{cases}.$$

   (g) Set $\widetilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$.
   (h) Compute $\mathsf{fct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\boldsymbol{\theta}_i, \widetilde{b}_i))$ for all $i \in [\ell + n]$ where $\boldsymbol{\theta}_k := \boldsymbol{\theta}$.

   The challenge sends $(\widetilde{\mathsf{fDec}}, \{(|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}, \mathsf{fct}_i)\}_{i \in [\ell+n] \setminus \{k\}}, \mathsf{fct}_k, \mathsf{fpk})$ to $\mathcal{B}$.
4. $\mathcal{B}$ outputs a bit b' as the final output of the experiment.

Let us define $\mathcal{Z}_\lambda^k(\boldsymbol{\theta}) = \mathsf{Expt}^{1,k}_{\mathcal{B},C}(\lambda, \boldsymbol{\theta}, \beta)$. We first show that

$$\left|\Pr\left[\mathcal{Z}_\lambda^k(\boldsymbol{\theta}) = 1\right] - \Pr\left[\mathcal{Z}_\lambda^k(\mathbf{0}_\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda). \tag{5}$$

$\mathcal{Z}_\lambda^{k,1}$**:** This is exactly the same as $\mathcal{Z}_\lambda^k(\boldsymbol{\theta})$ except the challenger uses the bits of $\widetilde{I} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_I, 1^{|R|})$ instead of $\widetilde{I} \leftarrow \mathsf{CC.Obf}(1^\lambda, I, \mathsf{lock}, R)$ to set $\widetilde{b}_i$ for all $i \in [k+1, \ell]$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^k(\boldsymbol{\theta})$ and $\mathcal{Z}_\lambda^{k,1}$ follows from the post-quantum security of the classical compute-and-compare obfuscation scheme Σ_{CCO}.

$\mathcal{Z}_\lambda^{k,2}$**:** This is exactly the same as $\mathcal{Z}_\lambda^{k,1}$ except the challenger replaces $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$ with the simulated obfuscated circuit $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_{\mathsf{fDec}}, 1^1)$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^{k,1}$ and $\mathcal{Z}_\lambda^{k,2}$ follows from the post-quantum security of the classical compute-and-compare scheme Σ_{CCO}.

$\mathcal{Z}_\lambda^{k,3}$**:** This is exactly the same as $\mathcal{Z}_\lambda^{k,2}$ except the challenger computes $\mathsf{fct}_k \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\mathbf{0}_\lambda, \widetilde{b}_k))$ instead of encrypting $(\boldsymbol{\theta}, \widetilde{b}_k)$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^{k,2}$ and $\mathcal{Z}_\lambda^{k,3}$ follows from the post-quantum security of Σ_{fhe}.

$\mathcal{Z}_\lambda^{k,4}$**:** This is exactly the same as $\mathcal{Z}_\lambda^{k,3}$ except the challenger replaces $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_{\mathsf{fDec}}, 1^1)$ with the real obfuscated circuit $\widetilde{\mathsf{fDec}} \leftarrow \mathsf{CC.Obf}(1^\lambda, \mathsf{fDec}, R, 1)$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^{k,3}$ and $\mathcal{Z}_\lambda^{k,4}$ follows from the post-quantum security of the classical compute-and-compare obfuscation scheme Σ_{CCO}.

$\mathcal{Z}_\lambda^{k,5}$**:** This is exactly the same as $\mathcal{Z}_\lambda^{k,4}$ except the challenger uses the bits of $\widetilde{I} \leftarrow \mathsf{CC.Obf}(1^\lambda, I, \mathsf{lock}, R)$ instead of $\widetilde{I} \leftarrow \mathsf{CC.Sim}(1^\lambda, \mathsf{pp}_I, 1^1)$ to set $\widetilde{b}_i$ for all $i \in [k+1, \ell]$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^{k,4}$ and $\mathcal{Z}_\lambda^{k,5}$ follows from the post-quantum security of the classical compute-and-compare obfuscation scheme Σ_{CCO}.

Observe that, the distributions $\mathcal{Z}_\lambda^{k,5}$ and $\mathcal{Z}_\lambda^k(\mathbf{0}_\lambda)$ are identical. Hence, Equation (5) holds for all $k \in [\ell]$. Therefore, by Lemma 3.5, for any (unbounded) adversary $\mathcal{B}'$ we have

$$\mathsf{TD}(\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, 0), \widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, 1)) \leq \mathsf{negl}(\lambda) \tag{6}$$

where the experiment $\widetilde{\mathcal{Z}}_\lambda^k(b) = \widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, b)$ works as follows:

$\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, b)$:

1. Sample $\boldsymbol{z}, \boldsymbol{\theta} \leftarrow \{0,1\}^\lambda$.
2. $\mathcal{B}'$ receives $(1^\lambda, |\boldsymbol{z}\rangle_{\boldsymbol{\theta}})$ as input.
3. $\mathcal{B}'$ interacts with C as in $\mathsf{Expt}^{1,k}_{\mathcal{B},C}(\lambda, \boldsymbol{\theta}, b \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j})$ where $\mathcal{B}'$ plays the role of $\mathcal{B}$.
4. $\mathcal{B}'$ outputs a string $\boldsymbol{z}' \in \{0,1\}^\lambda$ and a quantum state ρ.
5. If $z_j = z'_j$ for all $j \in [\lambda]$ such that $\theta_j = 1$ then the experiment outputs ρ, and otherwise it outputs a special symbol $\perp$.

Note that the only difference between $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ is that $\widetilde{b}_k$ is set to be $b_k \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k-1}$ and $0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k}$. Let us assume $b_k = 1$, since otherwise $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ are identical. We construct $\mathcal{B}'$ that distinguishes $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, 0)$ and $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',C}(\lambda, 1)$ if $\mathcal{A}$ distinguishes between the hybrids $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$.

$\mathcal{B}'(1^\lambda, |\boldsymbol{z}\rangle_{\boldsymbol{\theta}})$:

1. $\mathcal{B}'$ plays the role of $\mathcal{A}$ in $\mathsf{Hyb}_{1,k}$ where the external challenger $\mathcal{C}$ of $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, b)$ is used to simulate the challenger of $\mathsf{Hyb}_{1,k}$. $\mathcal{C}$ sends the obfuscated circuit to $\mathcal{A}$.
2. Suppose $\mathcal{A}$ sends a certificate $\mathsf{cert} = (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_\ell)$ to the challenger where $\boldsymbol{z}'_i = (z'_{i,j})_{j\in[\lambda]}$ for all $i \in [\ell]$. Then, $\mathcal{B}'$ sets $\boldsymbol{z}' := \boldsymbol{z}'_k$.
3. Outputs $\boldsymbol{z}'$ and the internal state ρ of $\mathcal{A}$.

We observe that $\mathcal{B}'$ perfectly simulates $\mathsf{Hyb}_{1,k-1}$ if $b = 1$ and $\mathsf{Hyb}_{1,k}$ if $b = 0$ (since we are assuming $b_k = 1$). Therefore, we can write

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \le \mathsf{TD}(\widetilde{\mathcal{Z}}^k_\lambda(0), \widetilde{\mathcal{Z}}^k_\lambda(1)). \tag{7}$$

Combining Equations (6) and (7), we have

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \le \mathsf{negl}(\lambda). \tag{8}$$

Recall that $\mathsf{Hyb}_{1,0} \equiv \mathsf{Hyb}_0$ and $\mathsf{Hyb}_{1,\ell} \equiv \mathsf{Hyb}_1$. Therefore, combining the advantages of $\mathcal{A}$ in the sequence of intermediate hybrids as obtained in Equation 8, we have

$$\mathsf{TD}(\mathsf{Hyb}_0, \mathsf{Hyb}_1) \le \mathsf{negl}(\lambda).$$

Hyb_2**:** This is exactly the same as Hyb_1 except the fact that instead of encrypting the challenge message $m \in \{0,1\}^n$ the FHE ciphertexts $\{\mathsf{fct}_{\ell+k}\}_{k\in[n]}$ are encrypted to the message $\mathbf{0}_n$. More precisely, the challenger samples $\boldsymbol{\theta}_{\ell+k}, \boldsymbol{z}_{\ell+k} \leftarrow \{0,1\}^\lambda$ and sets $\widetilde{b}_{\ell+k} := 0 \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$ instead of setting $\widetilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$. Finally, it obtains $\mathsf{fct}_{\ell+k} \leftarrow \mathsf{FHE.Enc}(\mathsf{fpk}, (\boldsymbol{\theta}_{\ell+k}, \widetilde{b}_{\ell+k}))$ for all $k \in [n]$ where the encrypted bits $\{\widetilde{b}_{\ell+k}\}_{k\in[n]}$ contain no information about the message m. Since the FHE master secret key fsk is not required to simulate the hybrids, the indistinguishability between Hyb_1 and Hyb_2 is guaranteed by the post-quantum semantic security of FHE. We can follow a similar argument as in the previous hybrid and show that

$$\mathsf{TD}(\mathsf{Hyb}_1, \mathsf{Hyb}_2) \le \mathsf{negl}(\lambda).$$

We observe that Hyb_2 is equivalent to $\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda, 1)$. Therefore, by combing the advantages of $\mathcal{A}$ in the consecutive hybrids and applying the triangular inequality, we have

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda, 0), \mathsf{EV\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda, 1)) \le \mathsf{negl}(\lambda).$$

Finally, it is easy to show the computational indistinguishability, i.e.,

$$\left|\Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{sim\text{-}ccobf}}_{\Sigma_{\mathsf{CECCO}},\mathcal{A}}(\lambda, 1) = 1\right]\right| \le \mathsf{negl}(\lambda)$$

using the security of FHE and the post-quantum security of CCO. We skip the formal description as it follows from the similar sequence of hybrids that we used to establish Equation (5) except that fct_k is changed from encryption of $(\boldsymbol{\theta}, \widetilde{b}_k)$ to $(\boldsymbol{\theta}, 0 \oplus \bigoplus_{j:\theta_{k,j}=0} z_{k,j})$ (instead of changing it from $(\boldsymbol{\theta}, \widetilde{b}_k)$ to $(\mathbf{0}_\lambda, \widetilde{b}_k)$ in $\mathcal{Z}^{k,3}_\lambda$). This completes the proof. ☐

6 Predicate Encryption with Certified Everlastng Deletion

6.1 Definition

We describe the notion of PE with certified everlasting deletion which generates a quantum ciphertext that can be deleted when required and the deletion is verified using a classical certificate of deletion.

Definition 6.1 (PE with Ceritifed Everlasting Deletion (Syntax)). *A certified everlasting PE is tuple of QPT algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *with a class predicates* $\mathcal{P}$*, a class of attributes* $\mathcal{X}$ *and a message space* $\mathcal{M}$*.*

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

$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{P})$**:** *The key generation algorithm takes as input the master secret key* msk *and a predicate* $\mathsf{P} \in \mathcal{P}$, *and outputs a secret key* sk_P *corresponding to the predicate* P.

$\mathcal{Enc}(\mathsf{pk}, x, m) \to (ct, \mathsf{vk})$**:** *The encryption algorithm takes as input the public key* pk, *an attribute* $x \in \mathcal{X}$ *and a message* $m \in \mathcal{M}$, *and outputs a quantum ciphertext* ct *and a classical verification key* vk.

$\mathcal{Dec}(\mathsf{sk}_\mathsf{P}, ct) \to m'$ **or** $\bot$**:** *The decryption algorithm takes as input a secret key* sk_P *and a quantum ciphertext* ct, *and outputs a classical plaintext* m' *or* $\bot$.

$\mathcal{Del}(ct) \to \mathsf{cert}$**:** *The deletion algorithm takes as input the ciphertext* ct *and outputs a classical certificate* cert.

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

Definition 6.2 (Correctness of PE with Certified Everlasting Deletion). *The correctness of PE with certified deletion for a class of predicates* $\mathcal{P}$ *is defined as follows.*

Decryption correctness: *For any* $\lambda \in \mathbb{N}, \mathsf{P} \in \mathcal{P}, x \in \mathcal{X}, m \in \mathcal{M}$ *such that* $\mathsf{P}(x) = 1$,

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

Verification correctness: *For any* $\lambda \in \mathbb{N}, x \in \mathcal{X}, m \in \mathcal{M}$,

$$\Pr\left[\mathsf{Vrfy}(\mathsf{vk}, \mathsf{cert}) \neq \top \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (ct, \mathsf{vk}) \leftarrow \mathcal{Enc}(\mathsf{pp}, x, m) \\ \mathsf{cert} \leftarrow \mathcal{Del}(ct) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 6.3 (Certified Everlasting Security of PE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be a PE with certified everlasting deletion for a class of predicates* $\mathcal{P}$, *a class of attributes* $\mathcal{X}$ *and a message space* $\mathcal{M}$. *We consider two experiments* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *and* $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *played between a challenger and and a non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$. *The experiments are defined as follows:*

1. *The challenger computes* $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$ *and sends* pk *to* $\mathcal{A}_\lambda(|\psi\rangle_\lambda)$.
2. $\mathcal{A}_\lambda$ *sends* $\mathsf{P} \in \mathcal{P}$ *to the challenger and receives* $\mathsf{sk}_P \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{P})$ *from the challenger.*
3. $\mathcal{A}_\lambda$ *sends a pair of challenge attributes* (x_0, x_1) *and a pair of challenge messages* (m_0, m_1) *satisfying the fact that* $\mathsf{P}(x_0) = \mathsf{P}(x_1) = 0$ *for all* P *queried so far in the key query phase.*
4. *The challenger computes* $(ct_b, \mathsf{vk}_b) \leftarrow \mathcal{Enc}(\mathsf{pk}, x_b, m_b)$ *and sends* ct_b *to* $\mathcal{A}_\lambda$.
5. $\mathcal{A}_\lambda$ *can make further key queries with* P *satisfying* $\mathsf{P}(x_0) = \mathsf{P}(x_1) = 0$.
6. $\mathcal{A}_\lambda$ *sends a certificate of deletion* cert *and its internal state* ρ *to the challenger.*
7. *The challenger computes* $\mathsf{Vrfy}(\mathsf{vk}_b, \mathsf{cert})$. *If the outcome is* $\top$, *the experiment* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *outputs* ρ; *otherwise if the outcome is* $\bot$ *then* $\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *output* $\bot$ *and ends.*
8. *The challenger sends the outcome of* $\mathsf{Vrfy}(\mathsf{vk}^{(b)}, \mathsf{cert})$ *to* $\mathcal{A}_\lambda$.
9. *Again,* $\mathcal{A}_\lambda$ *can make key queries with polynomial number of policies* P *satisfying* $\mathsf{P}(x_0) = \mathsf{P}(x_1) = 0$.

10. $\mathcal{A}_\lambda$ *outputs its guess* $b' \in \{0,1\}$ *which is the output of the experiment* $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,b)$.

We say that the Σ *is adaptively certified everlasting secure if for any non-uniform QPT adversary* $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$, *it holds that*

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,0), \mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,1)) \leq \mathsf{negl}(\lambda),$$

and

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

We can define similar experiment $\mathsf{EV\text{-}Exp}^{\mathsf{sel\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,b)$ and $\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,b)$ where $\mathcal{A}_\lambda$ is restricted to submit the challenge attributes x_0, x_1 before it receives pk from the challenger. We say that the Σ is selectively certified everlasting secure if for any non-uniform QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda\in\mathbb{N}}$, it holds that

$$\mathsf{TD}(\mathsf{EV\text{-}Exp}^{\mathsf{sel\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,0), \mathsf{EV\text{-}Exp}^{\mathsf{sel\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda,1)) \leq \mathsf{negl}(\lambda),$$

and

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

6.2 Construction

In this section, we construct a PE with certified everlasting deletion from a compute-and-compare obfuscation wiht certified everlasting deletion introduced in Section 5 and a classical ABE.

Ingredients. We use the following building blocks.

1. $\Sigma_{\mathsf{abe}} = \mathsf{ABE}.(\mathsf{Setup},\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$ be a classical ABE scheme for a class of predicates $\mathcal{P}$ and message space $\mathcal{M}_{\mathsf{abe}} = \{0,1\}^{\lambda+1}$.
2. $\Sigma_{\mathsf{CECCO}} = \mathsf{CCO}.(\mathcal{Obf},\mathcal{Del},\mathsf{Vrfy})$ be a compute-and-compare obfuscation with certified everlasting deletion for a message space $\mathcal{M}_{\mathsf{pe}}$ and the family of distributions $D = \{D_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}\}_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}$.

Let $D = \{D_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}\}_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}$ be a family of distributions where $D_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}$ outputs $(\mathsf{aDec},\mathsf{lock},\mathcal{aux})$ generated as follows.

- Generate $\mathsf{act}_i \leftarrow \mathsf{ABE}.\mathsf{Enc}(\mathsf{apk},x,(\boldsymbol{\theta}_i,\bigoplus_{j:\theta_{i,j}=0} z_{i,j}))$ for all $i \in [\ell]$.
- Construct aDec described in Figure 3.
- Choose $\mathsf{lock} \leftarrow \{0,1\}^\ell = \mathcal{K}$.
- Output $(\mathsf{aDec},\mathsf{lock},\mathcal{aux} := \bot)$.

PE with certified everlasting deletion. We construct $\Sigma_{\mathsf{pe\text{-}ce}} = (\mathsf{Setup},\mathsf{KeyGen},\mathcal{Enc},\mathcal{Dec},\mathcal{Del},\mathsf{Vrfy})$ for a class of predicates $\mathcal{P}$, a class of attributes $\mathcal{X}$ and a message space $\mathcal{M}_{\mathsf{pe}}$.

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

1. Sample $(\mathsf{apk},\mathsf{amsk}) \leftarrow \mathsf{ABE}.\mathsf{Setup}(1^\lambda)$.
2. Output $(\mathsf{pk} := \mathsf{apk}, \mathsf{msk} := \mathsf{amsk})$.

Hardwire: $\{\mathsf{act}_i\}_{i\in[\ell]}$
Input: $(\{\boldsymbol{z}_i\}_{i\in[\ell]}, \mathsf{sk}_P)$

1. Compute $(\boldsymbol{\theta}_i, \widetilde{r}_i) \leftarrow \mathsf{ABE.Dec}(\mathsf{sk}_P, \mathsf{act}_i)$ for all $i \in [\ell]$.
2. Compute $r_i = \widetilde{r}_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
3. Set $R := (r_1, \ldots, r_\ell)$
4. Output R

Figure 3: The description of the circuit aDec

$\mathsf{KeyGen}(\mathsf{msk}, P)$

1. Parse $\mathsf{msk} = \mathsf{amsk}$.
2. Compute $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk}, P)$.
3. Output sk_P.

$\mathcal{E}nc(\mathsf{apk}, x, m)$

1. Parse $\mathsf{pk} = \mathsf{apk}$.
2. Sample $R \leftarrow \{0,1\}^\ell$ and denote $R = (r_1, \ldots, r_\ell)$.
3. Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$.
4. Set $\widetilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
5. Compute $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x, (\boldsymbol{\theta}_i, \widetilde{r}_i))$ for all $i \in [\ell]$.
6. $(\widetilde{a\mathcal{D}ec}, \mathsf{vk}_{\mathsf{aDec}}) \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R, m)$ where aDec is defined in Figure 3.
7. Output $ct := (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$ and $\mathsf{vk} := (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$.

$\mathcal{D}ec(\mathsf{sk}_P, ct)$

1. Parse $ct = (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$.
2. Apply $\widetilde{a\mathcal{D}ec}$ in superposition to the input $(\{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]}, \mathsf{sk}_P)$ and measure the output register to obtain m'.
3. Output m'.

$\mathcal{D}el(ct)$

1. Parse $ct = (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$.
2. Measure $|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}$ in the Hadamard basis for all $i \in [\ell]$ and obtain $\boldsymbol{z}' := (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_\ell)$.
3. Compute $\mathsf{cert}_{\mathsf{aDec}} \leftarrow \mathsf{CCO}.\mathcal{D}el(\widetilde{a\mathcal{D}ec})$.
4. Output $\mathsf{cert} := (\boldsymbol{z}', \mathsf{cert}_{\mathsf{aDec}})$.

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

1. Parse $\mathsf{vk} := (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$ and $\mathsf{cert} := (\boldsymbol{z}', \mathsf{cert}_{\mathsf{aDec}})$.
2. If $(z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)$ holds for all $i \in [\ell]$ and $j \in [\lambda]$ and $\mathsf{CCO.Vrfy}(\mathsf{vk}_{\mathsf{aDec}}, \mathsf{cert}_{\mathsf{aDec}}) = \top$, then output $\top$; otherwise output $\bot$.

Theorem 6.4. *If Σ_{CECCO} is a certified everlasting secure compute-and-compare obfuscation for a message space $\mathcal{M}_{pe}$ and the family of distributions $D = \{D_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}\}_{\mathsf{apk},x,\{\boldsymbol{\theta}_i\}_i,\{\boldsymbol{z}_i\}_i}$ and Σ_{abe} is an adaptively (resp. selectively) secure ABE for a class of predicates $\mathcal{P}$, then $\Sigma_{pe\text{-}ce}$ is an adaptively (resp. selectively) certified everlasting secure predicate encryption scheme for the class of predicates $\mathcal{P}$, message space $\mathcal{M}_{pe}$.*

We focus on the case of adaptive security.

Proof. To prove the theorem we consider an adversary $\mathcal{A}$ against the certified everlasting security of $\Sigma_{\mathsf{pe\text{-}ce}}$. We consider the following sequence of hybrids.

Hyb_0 : This is the original certified everlasting security experiment where the challenge bit is set to 0 ($\mathsf{EV\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma_{\mathsf{pe\text{-}ce}},\mathcal{A}}(\lambda, 0)$). More precisely, it works as follows:

1. The challenger computes $(\mathsf{apk}, \mathsf{amsk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, sets $\mathsf{pk} := \mathsf{apk}$, and sends pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends any polynomial number of secret key queries for $P \in \mathcal{P}$ at any point of the experiment. The challenger generates $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk}, P)$ and sends sk_P to $\mathcal{A}$.
3. $\mathcal{A}$ sends a pair of challenge attributes (x_0, x_1) and a pair of challenge messages (m_0, m_1) satisfying the fact that $P(x_0) = P(x_1) = 0$ for all P queried so far in the key query phase.
4. The challenger computes the challenge ciphertext as follows:
   (a) Sample $R^* = (r_1, \dots, r_\ell) \leftarrow \{0,1\}^\ell$.
   (b) Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$ where $\boldsymbol{\theta}_i = (\theta_{i,j})_{j\in[\lambda]}$ and $\boldsymbol{z}_i = (z_{i,j})_{j\in[\lambda]}$.
   (c) Set $\widetilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
   (d) Compute $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x_0, (\boldsymbol{\theta}_i, \widetilde{r}_i))$ for all $i \in [\ell]$.
   (e) $(\widetilde{a\mathcal{D}ec}, \mathsf{vk}_{\mathsf{aDec}}) \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R^*, m_0)$ where aDec is defined in Figure 3.
   (f) Set $ct^* := (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$ and $\mathsf{vk} := (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$.

   The challenger sends ct^* to $\mathcal{A}$.
5. $\mathcal{A}$ sends a certificate $\mathsf{cert} = (\boldsymbol{z}' = (\boldsymbol{z}'_1, \dots, \boldsymbol{z}'_\ell), \mathsf{cert}_{\mathsf{aDec}})$ and its internal state ρ to the challenger where $\boldsymbol{z}'_i = (z'_{i,j})_{j\in[\lambda]}$ for all $i \in [\ell]$.
6. The challenger checks if $(z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)$ holds for all $i \in [\ell]$ and $j \in [\lambda]$ and $\mathsf{CCO.Vrfy}(\mathsf{vk}_{\mathsf{aDec}}, \mathsf{cert}_{\mathsf{aDec}}) = \top$. If it does not hold, the challenger outputs $\bot$ as the final output of the experiment. Otherwise, go to the next step.
7. The experiment outputs ρ as a final output.

Hyb_1 : This hybrid proceeds exactly similar to Hybd_0 except that the ABE ciphertexts act_i is now replaced with encryption of zero string. In particular, the hardwired values of aDec are computed as $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x_0, (\boldsymbol{\theta}_i, 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}))$ for all $i \in [\ell]$.

To prove the indistinguishability between Hyb_0 and Hyb_1, we consider a sequence of intermediate hybrids $\mathsf{Hyb}_{1,k}$ for $k \in [\ell]$ where we take $\mathsf{Hyb}_{1,0}$ is identical to Hyb_0 and the only difference between $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ is that act_k is an encryption of $(\boldsymbol{\theta}_k, r_k \oplus \bigoplus_{j:\theta_{k,j}=0} z_{k,j})$ in $\mathsf{Hyb}_{1,k-1}$ whereas it is an encryption of $(\boldsymbol{\theta}_k, 0 \oplus \bigoplus_{j:\theta_{k,j}=0} z_{k,j})$ in $\mathsf{Hyb}_{1,k}$.

Now, we consider a sequence of experiments $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda, \boldsymbol{\theta}, \beta)$ for $k \in [\ell]$ between a QPT adversary $\mathcal{B}$ and a challenger $\mathcal{C}$ for $\boldsymbol{\theta} \in \{0,1\}^\lambda$ and $\beta \in \{0,1\}$. The experiment $\mathsf{Expt}^{1,0}_{\mathcal{B},\mathcal{C}}(\lambda, \boldsymbol{\theta}, \beta)$ is basically the same as Hyb_0 where $\mathcal{B}$ plays the role of $\mathcal{A}$ and $\mathcal{C}$ plays the role of the challenger. In particular, it works as follows:

$\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\boldsymbol{\theta},\beta)$:

1. $\mathcal{C}$ computes $(\mathsf{apk},\mathsf{amsk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, sets $\mathsf{pk} := \mathsf{apk}$, and sends pk to $\mathcal{B}$.
2. $\mathcal{B}$ sends any polynomial number of secret key queries for $P \in \mathcal{P}$ at any point of the experiment and $\mathcal{C}$ generates $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk},P)$ and sends sk_P to $\mathcal{B}$.
3. $\mathcal{B}$ sends a pair of challenge attributes (x_0,x_1) and a pair of challenge messages (m_0,m_1) satisfying the fact that $P(x_0)=P(x_1)=0$ for all P queried so far in the key query phase.
4. $\mathcal{C}$ computes the challenge ciphertext as follows:
   (a) Sample $R^* = (r_1,\ldots,r_\ell) \leftarrow \{0,1\}^\ell$.
   (b) Sample $\boldsymbol{z}_i,\boldsymbol{\theta}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]\setminus\{k\}$ where $\boldsymbol{\theta}_i = (\theta_{i,j})_{j\in[\lambda]}$ and $\boldsymbol{z}_i = (z_{i,j})_{j\in[\lambda]}$.
   (c) Set $\widetilde{r}_i$ as follows:
$$\widetilde{r}_i := \begin{cases} 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [1,k-1] \\ \beta & \text{if } i = k \\ r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [k+1,\ell] \end{cases}.$$
   (d) Compute act_i as follows:
$$\mathsf{act}_i \leftarrow \begin{cases} \mathsf{ABE.Enc}(\mathsf{apk},x_0,(\boldsymbol{\theta},\widetilde{r}_k)) & \text{if } i = k \\ \mathsf{ABE.Enc}(\mathsf{apk},x_0,(\boldsymbol{\theta}_i,\widetilde{r}_i)) & \text{if } i \in [\ell]\setminus\{k\} \end{cases}.$$
   (e) $(\widetilde{a\mathcal{D}ec},\mathsf{vk}_{\mathsf{aDec}}) \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda,\mathsf{aDec},R^*,m_0)$ where aDec is defined in Figure 3.

   The challenger sends $(\widetilde{a\mathcal{D}ec},\{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]\setminus\{k\}})$ to $\mathcal{B}$.
5. $\mathcal{B}$ outputs a bit b' as the final output of the experiment.

Since all the secret keys sk_P corresponding to predicates P queried by the adversary satisfy the condition that $P(x_0)=0$, the semantic security of ABE ensures that

$$\left|\Pr\left[\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\boldsymbol{\theta},\beta)=1\right] - \Pr\left[\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\mathbf{0}_\lambda,\beta)=1\right]\right| \le \mathsf{negl}(\lambda).$$

Therefore, by Lemma 3.7, for any QPT (unbounded) adversary $\mathcal{B}'$, we have

$$\mathsf{TD}(\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,0),\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,1)) \le \mathsf{negl}(\lambda) \tag{9}$$

where the experiment $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,b)$ works as follows:

$\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,b)$:

1. Sample $\boldsymbol{z},\boldsymbol{\theta} \leftarrow \{0,1\}^\lambda$.
2. $\mathcal{B}'$ takes $(1^\lambda,|\boldsymbol{z}\rangle_{\boldsymbol{\theta}})$ as input.
3. $\mathcal{B}'$ interacts with $\mathcal{C}$ as in $\mathsf{Expt}^{1,k}_{\mathcal{B},\mathcal{C}}(\lambda,\boldsymbol{\theta},b\oplus\bigoplus_{j:\theta_{i,j}=0} z_{i,j})$ where $\mathcal{B}'$ plays the role of $\mathcal{B}$.
4. $\mathcal{B}'$ outputs a string $\boldsymbol{z}' \in \{0,1\}^\lambda$ and a quantum state ρ.
5. If $z_j = z'_j$ for all $j\in[\lambda]$ such that $\theta_j = 1$ then the experiment outputs ρ, and otherwise it outputs a special symbol $\perp$.

Note that the only difference between $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ is that $\widetilde{r}_k$ is set to be $r_k \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k-1}$ and $0\oplus\bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\mathsf{Hyb}_{1,k}$. Let us assume $r_k = 1$, since if r_k is 0 then $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$ are identical. We construct $\mathcal{B}'$ that distinguishes $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,0)$ and $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda,1)$ if $\mathcal{A}$ distinguishes between the hybrids $\mathsf{Hyb}_{1,k-1}$ and $\mathsf{Hyb}_{1,k}$.

$\mathcal{B}'(1^\lambda, |z\rangle_{\theta})$:

1. $\mathcal{B}'$ plays the role of $\mathcal{A}$ in $\mathsf{Hyb}_{1,k}$ where the external challenger $\mathcal{C}$ of $\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, b)$ is used to simulate the challenger of $\mathsf{Hyb}_{1,k}$. $\mathcal{C}$ provides everything that should be sent to $\mathcal{A}$ (as in Hyb_0).
2. Suppose $\mathcal{A}$ sends a certificate $\mathsf{cert} = ((z'_1, \ldots, z'_\ell), \mathsf{cert}_{\mathsf{aDec}})$ to the challenger where $z'_i = (z'_{i,j})_{j\in[\lambda]}$ for all $i \in [\ell]$. Then, $\mathcal{B}'$ sets $z' = z'_k$.
3. Outputs z' and the internal state ρ of $\mathcal{A}$ which it sends to $\mathcal{A}_2$.

We observe that $\mathcal{B}'$ perfectly simulates $\mathsf{Hyb}_{1,k}$ if $b = 0$ and $\mathsf{Hyb}_{1,k-1}$ if $b = 1$ (since we are assuming $r_k = 1$). Therefore, we can write

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \leq \mathsf{TD}(\widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, 0), \widetilde{\mathsf{Expt}}^{1,k}_{\mathcal{B}',\mathcal{C}}(\lambda, 1)). \tag{10}$$

Combining Equations 9 and 10, we have

$$\mathsf{TD}(\mathsf{Hyb}_{1,k-1}, \mathsf{Hyb}_{1,k}) \leq \mathsf{negl}(\lambda). \tag{11}$$

Recall that $\mathsf{Hyb}_{1,0} \equiv \mathsf{Hyb}_0$ and $\mathsf{Hyb}_{1,\ell} \equiv \mathsf{Hyb}_1$. Therefore, combining the advantages of $\mathcal{A}$ in the sequence of intermediate hybrids as obtained in Equation 11, we have

$$\mathsf{TD}(\mathsf{Hyb}_0, \mathsf{Hyb}_1) \leq \mathsf{negl}(\lambda).$$

Hyb_2 : This hybrid proceeds exactly similar to Hyb_1 except that the obfuscated circuit is now replaced with a simulated version of it. In particular, $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R^*, m_0)$ is replaced with the circuit $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{S}im(1^\lambda, \mathsf{pp}_{\mathsf{aDec}}, 1^{|m_b|})$. In particular, the hybrid works as follows:

1. The challenger computes $(\mathsf{apk}, \mathsf{amsk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, sets $\mathsf{pk} := \mathsf{apk}$, and sends pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends any polynomial number of secret key queries for $P \in \mathcal{P}$ at any point of the experiment. The challenger generates $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk}, P)$ and sends sk_P to $\mathcal{A}$.
3. $\mathcal{A}$ sends a pair of challenge attributes (x_0, x_1) and a pair of challenge messages (m_0, m_1) satisfying the fact that $P(x_0) = P(x_1) = 0$ for all P queried so far in the key query phase.
4. The challenger computes the challenge ciphertext as follows:
   (a) Sample $\theta_i, z_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$ where $\theta_i = (\theta_{i,j})_{j\in[\lambda]}$ and $z_i = (z_{i,j})_{j\in[\lambda]}$.
   (b) $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{S}im(1^\lambda, \mathsf{pp}_{\mathsf{aDec}}, 1^{|m_b|})$ where aDec is defined in Figure 3. (Note that, we do not need to compute ABE ciphertexts since we only require the lengths of a ABE ciphertext in order to calculate $\mathsf{pp}_{\mathsf{aDec}}$.)
   (c) Set $ct^* := (\widetilde{a\mathcal{D}ec}, \{|z_i\rangle_{\theta_i}\}_{i\in[\ell]})$ and $\mathsf{vk} := (\{(z_i, \theta_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$.

   The challenger sends ct^* to $\mathcal{A}$.
5. $\mathcal{A}$ sends a certificate $\mathsf{cert} = (z' = (z'_1, \ldots, z'_\ell), \mathsf{cert}_{\mathsf{aDec}})$ and its internal state ρ to the challenger where $z'_i = (z'_{i,j})_{j\in[\lambda]}$ for all $i \in [\ell]$.
6. The challenger checks if $(z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)$ holds for all $i \in [\ell]$ and $j \in [\lambda]$ and $\mathsf{CCO.Vrfy}(\mathsf{vk}_{\mathsf{aDec}}, \mathsf{cert}_{\mathsf{aDec}}) = \top$. If it does not hold, the challenger outputs $\perp$ as the final output of the experiment. Otherwise, go to the next step.
7. The experiment outputs ρ as a final output.

Since the information of lock string R^* is not used in generating the ABE ciphertexts act_i, the certified everlasting security of compute-and-compare obfuscation guarantees that Hyb_1 and Hyb_2 are indistinguishable to $\mathcal{A}$. In other words, we have

$$\mathsf{TD}(\mathsf{Hyb}_1, \mathsf{Hyb}_2) \leq \mathsf{negl}(\lambda).$$

Hyb_3 : This hybrid proceeds exactly similar to Hyb_2 except that the simulated circuit is now replaced with a honestly obfuscated version of it. In particular, the obfuscated circuit is computed as $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R^*, m_1)$ where the circuit aDec is defined using the hardwired values $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x_1, (\boldsymbol{\theta}_i, 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}))$. In particular, the hybrid works as follows:

1. The challenger computes $(\mathsf{apk}, \mathsf{amsk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, sets $\mathsf{pk} := \mathsf{apk}$, and sends pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends any polynomial number of secret key queries for $P \in \mathcal{P}$ at any point of the experiment. The challenger generates $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk}, P)$ and sends sk_P to $\mathcal{A}$.
3. $\mathcal{A}$ sends a pair of challenge attributes (x_0, x_1) and a pair of challenge messages (m_0, m_1) satisfying the fact that $P(x_0) = P(x_1) = 0$ for all P queried so far in the key query phase.
4. The challenger computes the challenge ciphertext as follows:
   (a) Sample $R^* = (r_1, \ldots, r_\ell) \leftarrow \{0,1\}^\ell$.
   (b) Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$ where $\boldsymbol{\theta}_i = (\theta_{i,j})_{j\in[\lambda]}$ and $\boldsymbol{z}_i = (z_{i,j})_{j\in[\lambda]}$.
   (c) Set $\widetilde{r}_i := 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
   (d) Compute $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x_1, (\boldsymbol{\theta}_i, \widetilde{r}_i))$ for all $i \in [\ell]$.
   (e) $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R^*, m_1)$ where aDec is defined in Figure 3.
   (f) Set $ct^* := (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$ and $\mathsf{vk} := (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$.

   The challenger sends ct^* to $\mathcal{A}$.
5. $\mathcal{A}$ sends a certificate $\mathsf{cert} = (\boldsymbol{z}' = (\boldsymbol{z}'_1, \ldots, \boldsymbol{z}'_\ell), \mathsf{cert}_{\mathsf{aDec}})$ and its internal state ρ to the challenger where $\boldsymbol{z}'_i = (z'_{i,j})_{j\in[\lambda]}$ for all $i \in [\ell]$.
6. The challenger checks if $(z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)$ holds for all $i \in [\ell]$ and $j \in [\lambda]$ and $\mathsf{CCO.Vrfy}(\mathsf{vk}_{\mathsf{aDec}}, \mathsf{cert}_{\mathsf{aDec}}) = \top$. If it does not hold, the challenger outputs $\perp$ as the final output of the experiment. Otherwise, go to the next step.
7. The experiment outputs ρ as a final output.

By similar argument as in the previous hybrid, the hybrids Hyb_2 and Hyb_3 are indistinguishable by the certified everlasting security of compute-and-compare obfuscation. In other words, we have

$$\mathsf{TD}(\mathsf{Hyb}_2, \mathsf{Hyb}_3) \leq \mathsf{negl}(\lambda).$$

Hyb_4 : This hybrid proceeds exactly similar to Hyb_3 except that the ABE ciphertexts act_i is now replaced with encryption of $(\boldsymbol{\theta}_i, \widetilde{r}_i)$ where $\widetilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$. In particular, the hybrid works as follows:

1. The challenger computes $(\mathsf{apk}, \mathsf{amsk}) \leftarrow \mathsf{ABE.Setup}(1^\lambda)$, sets $\mathsf{pk} := \mathsf{apk}$, and sends pk to $\mathcal{A}$.
2. The adversary $\mathcal{A}$ sends any polynomial number of secret key queries for $P \in \mathcal{P}$ at any point of the experiment. The challenger generates $\mathsf{sk}_P \leftarrow \mathsf{ABE.KeyGen}(\mathsf{amsk}, P)$ and sends sk_P to $\mathcal{A}$.
3. $\mathcal{A}$ sends a pair of challenge attributes (x_0, x_1) and a pair of challenge messages (m_0, m_1) satisfying the fact that $P(x_0) = P(x_1) = 0$ for all P queried so far in the key query phase.
4. The challenger computes the challenge ciphertext as follows:
   (a) Sample $R^* = (r_1, \ldots, r_\ell) \leftarrow \{0,1\}^\ell$.
   (b) Sample $\boldsymbol{\theta}_i, \boldsymbol{z}_i \leftarrow \{0,1\}^\lambda$ for all $i \in [\ell]$ where $\boldsymbol{\theta}_i = (\theta_{i,j})_{j\in[\lambda]}$ and $\boldsymbol{z}_i = (z_{i,j})_{j\in[\lambda]}$.
   (c) Set $\widetilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
   (d) Compute $\mathsf{act}_i \leftarrow \mathsf{ABE.Enc}(\mathsf{apk}, x_1, (\boldsymbol{\theta}_i, \widetilde{r}_i))$ for all $i \in [\ell]$.
   (e) $\widetilde{a\mathcal{D}ec} \leftarrow \mathsf{CCO}.\mathcal{O}bf(1^\lambda, \mathsf{aDec}, R^*, m_1)$ where aDec is defined in Figure 3.
   (f) Set $ct^* := (\widetilde{a\mathcal{D}ec}, \{|\boldsymbol{z}_i\rangle_{\boldsymbol{\theta}_i}\}_{i\in[\ell]})$ and $\mathsf{vk} := (\{(\boldsymbol{z}_i, \boldsymbol{\theta}_i)\}_{i\in[\ell]}, \mathsf{vk}_{\mathsf{aDec}})$.

The challenger sends ct^* to $\mathcal{A}$.

5. $\mathcal{A}$ sends a certificate $\mathsf{cert} = (z' = (z'_1, \ldots, z'_\ell), \mathsf{cert}_{\mathsf{aDec}})$ and its internal state ρ to the challenger where $z'_i = (z'_{i,j})_{j \in [\lambda]}$ for all $i \in [\ell]$.
6. The challenger checks if $(z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1)$ holds for all $i \in [\ell]$ and $j \in [\lambda]$ and $\mathsf{CCO.Vrfy}(\mathsf{vk}_{\mathsf{aDec}}, \mathsf{cert}_{\mathsf{aDec}}) = \top$. If it does not hold, the challenger outputs $\perp$ as the final output of the experiment. Otherwise, go to the next step.
7. The experiment outputs ρ as a final output.

Since all the secret keys sk_P corresponding to predicates P queried by the adversary satisy the condition that $P(x_1) = 0$, we can depend on the semantic security of ABE and show that the hybrids Hyb_3 and Hyb_4 are indistinguishable from $\mathcal{A}$'s point of view using the similar argument that we used while establishing the indistinguishability between the hybrids Hyb_0 and Hyb_1. In other words, we have

$$\mathsf{TD}(\mathsf{Hyb}_3, \mathsf{Hyb}_4) \leq \mathsf{negl}(\lambda).$$

Finally, we note that Hyb_4 is the original certified everlasting experiment of $\Sigma_{\mathsf{pe\text{-}ce}}$ where the challenge bit is set to 1. Therefore, combing the advantages of $\mathcal{A}$ in the consecutive hybrids and applying the triangular inequality, we have

$$\mathsf{TD}(\mathsf{Hyb}_0, \mathsf{Hyb}_4) \leq \mathsf{negl}(\lambda).$$

Finally, it is easy to show the computational indistinguishability, i.e.,

$$\left|\Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma_{\mathsf{pe\text{-}ce}}, \mathcal{A}}(\lambda, 0) = 1\right] - \Pr\left[\mathsf{C\text{-}Exp}^{\mathsf{ada\text{-}ind}}_{\Sigma_{\mathsf{pe\text{-}ce}}, \mathcal{A}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

We skip the formal description as it follows from the security of ABE and the security of CECCO. We can erase information about $R = (r_, \ldots, r_\ell)$ by the security of ABE. Then, we can apply the security of CECCO. This completes the proof. ☐

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. TH is supported by JSPS research fellowship and by JSPS KAKENHI No. JP22J21864.

References

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 657–677. Springer, Heidelberg, August 2015. (Cited on page 15.)

[AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 500–518. Springer, Heidelberg, August 2013. (Cited on page 36.)

[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 16.)

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006. (Cited on page 53.)

[AKYY23] Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Broadcast, trace and revoke with optimal parameters from polynomial hardness. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 605–636. Springer, Heidelberg, April 2023. (Cited on page 5.)

[AL21] Prabhanjan Ananth and Rolando L. La Placa. Secure software leasing. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 501–530. Springer, Heidelberg, October 2021. (Cited on page 14.)

[ALL+21] Scott Aaronson, Jiahui Liu, Qipeng Liu, Mark Zhandry, and Ruizhe Zhang. New approaches for quantum copy-protection. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 526–555, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 15.)

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Heidelberg, May 2020. (Cited on page 4, 15, 23.)

[AS16] Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016. (Cited on page 7, 83, 86.)

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, December 2019. (Cited on page 4, 54.)

[AYY22] Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 590–621. Springer, Heidelberg, August 2022. (Cited on page 5.)

[BCKM21] James Bartusek, Andrea Coladangelo, Dakshita Khurana, and Fermi Ma. On the round complexity of secure quantum computation. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 406–435, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 11.)

[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 5, 16.)

[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 14.)

[BGG+14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014. (Cited on page 22.)

[BGG+23] James Bartusek, Sanjam Garg, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Obfuscation and outsourced computation with certified deletion. Cryptology ePrint Archive, Report 2023/265, 2023. `https://eprint.iacr.org/2023/265`. (Cited on page 6.)

[BGI+12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *Journal of the ACM*, 59(2):6:1–6:48, 2012. (Cited on page 23.)

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018. (Cited on page 15, 23.)

[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 4, 14, 20, 21.)

[BJL+21] Anne Broadbent, Stacey Jeffery, Sébastien Lord, Supartha Podder, and Aarthi Sundaram. Secure software leasing without assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 90–120. Springer, Heidelberg, November 2021. (Cited on page 15.)

[BK23] James Bartusek and Dakshita Khurana. Cryptography with certified deletion. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 192–223. Springer, 2023. (Cited on page 6, 7, 8, 12, 13, 14, 25, 26, 27, 37, 39, 54, 57.)

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. (Cited on page 4.)

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on Computing*, 43(2):831–871, 2014. (Cited on page 23.)

[CCP18] California consumer privacy act. 2018. (Cited on page 4.)

[CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996. (Cited on page 76.)

[CHK05] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168. Springer, Heidelberg, February 2005. (Cited on page 76.)

[CHVW19] Yilei Chen, Minki Hhan, Vinod Vaikuntanathan, and Hoeteck Wee. Matrix PRFs: Constructions, attacks, and applications to obfuscation. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 55–80. Springer, Heidelberg, December 2019. (Cited on page 15, 23.)

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 556–584, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page 6.)

[CMP20] Andrea Coladangelo, Christian Majenz, and Alexander Poremba. Quantum copy-protection of compute-and-compare programs in the quantum random oracle model. Cryptology ePrint Archive, Report 2020/1194, 2020. `https://eprint.iacr.org/2020/1194`. (Cited on page 15.)

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369. Springer, Heidelberg, November 2018. (Cited on page 5.)

[FFMV23] Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption from learning with errors. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 573–604. Springer, Heidelberg, April 2023. (Cited on page 5.)

[GDP16] Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46 (general data protection regulation). *Official Journal of the European Union (OJ)*, pages 1–88, 2016. (Cited on page 4.)

[GGG$^+$14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014. (Cited on page 6, 28, 76, 79.)

[GGH$^+$16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, 45(3):882–929, 2016. (Cited on page 4, 7, 15.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986. (Cited on page 17.)

[GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. i-Hop homomorphic encryption and rerandomizable Yao circuits. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 155–172. Springer, Heidelberg, August 2010. (Cited on page 10.)

[GJO16] Vipul Goyal, Aayush Jain, and Adam O'Neill. Multi-input functional encryption with unbounded-message security. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 531–556. Springer, Heidelberg, December 2016. (Cited on page 6.)

[GKW17] Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, *58th FOCS*, pages 612–621. IEEE Computer Society Press, October 2017. (Cited on page 4, 5, 13, 15, 24.)

[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 4.)

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. (Cited on page 20.)

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 75–92. Springer, Heidelberg, August 2013. (Cited on page 23.)

[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 4, 10, 12, 15, 36, 37, 38, 47, 52, 53, 54.)

[GVW15a] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. *Journal of the ACM*, 62(6):45:1–45:33, 2015. (Cited on page 22.)

[GVW15b] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015. (Cited on page 5, 15.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 17.)

[HMNY21] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Quantum encryption with certified deletion, revisited: Public key, attribute-based, and classical communication. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part I*, volume 13090 of *LNCS*, pages 606–636. Springer, Heidelberg, December 2021. (Cited on page 4, 14, 20.)

[HMNY22a] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting functional encryption. Cryptology ePrint Archive, Report 2022/969, 2022. https://eprint.iacr.org/2022/969. (Cited on page 6.)

[HMNY22b] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting zero-knowledge proof for QMA. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 239–268. Springer, Heidelberg, August 2022. (Cited on page 4, 10, 14, 97, 98, 99.)

[JL00] Stanislaw Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 221–242. Springer, Heidelberg, May 2000. (Cited on page 76.)

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *53rd ACM STOC*, pages 60–73. ACM Press, June 2021. (Cited on page 4, 15.)

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from LPN over $\mathbb{F}_p$, DLIN, and PRGs in NC^0. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 670–699. Springer, Heidelberg, May / June 2022. (Cited on page 4, 15.)

[Klu22] Kamil Kluczniak. Lockable obfuscation from circularly insecure fully homomorphic encryption. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 69–98. Springer, 2022. (Cited on page 6, 15.)

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Functional encryption with secure key leasing. In *Asiacrypt 2022*, 2022. (Cited on page 14.)

[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 10, 77, 104.)

[KNY21] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Secure software leasing from standard assumptions. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 31–61. Springer, Heidelberg, November 2021. (Cited on page 15.)

[KNY23] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Publicly verifiable deletion from minimal assumptions. In Guy N. Rothblum and Hoeteck Wee, editors, *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*, volume 14372 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2023. (Cited on page 5, 7.)

[KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008. (Cited on page 5.)

[KT20] Srijita Kundu and Ernest Tan. Composably secure device-independent encryption with certified deletion. *arXiv*, 2011.12704, 2020. (Cited on page 4, 14.)

[LC97] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Phys. Rev. Lett.*, 78:3410–3413, 1997. (Cited on page 4.)

[LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009. (Cited on page 10, 17.)

[May97] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, 78:3414–3417, 1997. (Cited on page 4.)

[MW18] Sanketh Menda and John Watrous. Oracle separations for quantum statistical zero-knowledge. *arXiv:1801.08967*, 2018. (Cited on page 4.)

[Por23] Alexander Poremba. Quantum proofs of deletion for learning with errors. In Yael Tauman Kalai, editor, *14th Innovations in Theoretical Computer Science Conference, ITCS 2023, January 10-13, 2023, MIT, Cambridge, Massachusetts, USA*, volume 251 of *LIPIcs*, pages 90:1–90:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. (Cited on page 4, 14.)

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):34:1–34:40, 2009. (Cited on page 20.)

[Sha79] Adi Shamir. How to share a secret. *cacm*, 22(11):612–613, nov 1979. (Cited on page 47.)

[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 16.)

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 463–472. ACM Press, October 2010. (Cited on page 10, 39.)

[Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. (Cited on page 4, 10, 14, 16, 98, 99, 100, 102.)

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 678–697. Springer, Heidelberg, August 2015. (Cited on page 15.)

[Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999. (Cited on page 25.)

[WZ17] Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, *58th FOCS*, pages 600–611. IEEE Computer Society Press, October 2017. (Cited on page 4, 5, 13, 15, 24.)

[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 10, 111.)

[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 16.)

A Omitted Proofs for Collusion-Resistant FE

We prove the adaptive security of our collusion-resistant scheme CED in Section 3.3. That is, we show

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

Let $\mathcal{A}$ be a QPT adversary against the adaptive security. We consider the following sequence of hybrids.

Hyb_0**:** This is the original adaptive security experiment where the challenge bit is set to be 0. Specifically, it works as follows:

1. The challenger generates $(\mathsf{fe.MPK},\mathsf{fe.MSK})\leftarrow\mathsf{FE.Setup}(1^\lambda)$, sets $\mathsf{MPK}:=\mathsf{fe.MPK}$ and $\mathsf{MSK}:=\mathsf{fe.MSK}$, and sends MPK to $\mathcal{A}$.

2. $\mathcal{A}$ can make arbitrarily many key queries at any point of the experiment. When it makes a key query f, the challenger generates $\mathsf{fe.sk}_{g[f]} \leftarrow \mathsf{FE.KeyGen}(\mathsf{fe.MSK}, g[f])$ and returns $\mathsf{sk}_f = \mathsf{fe.sk}_{g[f]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $(m^{(0)}, m^{(1)})$ to the challenger.[23] It must satisfy $f(m^{(0)}) = f(m^{(1)})$ for all key queries f that are made before or after sending $(m^{(0)}, m^{(1)})$.
4. The challenger generates $(ct, \mathsf{vk}) \leftarrow \mathcal{E}nc(\mathsf{MPK}, m^{(0)})$. Specifically,
   (a) Generate $z_i, \theta_i \leftarrow \{0,1\}^\lambda$ for every $i \in [2n+1]$.
   (b) Generate $u_{i,j,b} \leftarrow \{0,1\}^\lambda$ and compute $v_{i,j,b} \leftarrow \mathsf{PRG}(u_{i,j,b})$ for every $i \in [2n+1]$, $j \in [\lambda]$ and $b \in \{0,1\}$ and set $U = (u_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$ and $V := (v_{i,j,b})_{i\in[2n+1],j\in[\lambda],b\in\{0,1\}}$.
   (c) Generate a state
   $$\left|\psi_{i,j}\right\rangle := \begin{cases} \left|z_{i,j}\right\rangle \left|u_{i,j,z_{i,j}}\right\rangle & \text{if } \theta_{i,j} = 0 \\ \left|0\right\rangle \left|u_{i,j,0}\right\rangle + (-1)^{z_{i,j}} \left|1\right\rangle \left|u_{i,j,1}\right\rangle & \text{if } \theta_{i,j} = 1 \end{cases}$$
   where $\theta_{i,j}$ (resp. $z_{i,j}$) is the j-th bit of θ_i (resp. z_i) for every $i \in [2n+1]$ and $j \in [\lambda]$.
   (d) Generate
   $$\beta_i := \begin{cases} m_i^{(0)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n+1] \end{cases}.$$
   (e) Generate $\mathsf{fe.ct} \leftarrow \mathsf{FE.Enc}(\mathsf{fe.MPK}, V\|\theta_1\|\ldots\|\theta_{2n+1}\|\beta_1\|\ldots\|\beta_{2n+1})$.
   (f) Set $ct = (\mathsf{fe.ct}, \bigotimes_{i\in[2n+1],j\in[\lambda]} \left|\psi_{i,j}\right\rangle)$ and $\mathsf{vk} = (U, (z_i, \theta_i)_{i\in[2n+1]})$.

   The challenger sends ct to $\mathcal{A}$.
5. If $\mathcal{A}$ sends a certificate of deletion cert, the challenger computes $\mathsf{Vrfy}(\mathsf{vk}_0, \mathsf{cert})$ and sends the result to $\mathcal{A}$.
6. Again, the challenger answers key queries from $\mathcal{A}$.
7. When $\mathcal{A}$ outputs a bit b', the experiment outputs b' if $f_\ell(m_0) = f_\ell(m_1)$ holds for all key queries f_ℓ.

Hyb_1**:** This is identical to Hyb_0 except that $v_{i,j,1\oplus z_{i,j}}$ is uniformly chosen from $\{0,1\}^{2\lambda}$ instead of being set to be $\mathsf{PRG}(u_{i,j,1\oplus z_{i,j}})$ for all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$.

Hyb_2**:** This is identical to Hyb_1 except that $(\beta_i)_{i\in[2n+1]}$ is generated as

$$\beta_i := \begin{cases} m_i^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n+1] \end{cases}.$$

Hyb_3**:** This is identical to Hyb_2 except that $v_{i,j,b}$ is set to be $\mathsf{PRG}(u_{i,j,b})$ for all $i \in [2n+1]$, $j \in [\lambda]$, and $b \in \{0,1\}$.

Note that Hyb_3 is identical to the original adaptive security experiment where the challenge bit is set to be 1. Thus, we only have to prove

$$|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| \leq \mathsf{negl}(\lambda). \tag{12}$$

We prove Equation (12) by the following lemmata.

Lemma A.1. *If* PRG *is a secure PRG,*

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

[23] We use $(m^{(0)}, m^{(1)})$ instead of (m_0, m_1) to denote a pair of challenge messages to avoid a notational collision.

Proof. Noting that $u_{i,j,1\oplus z_{i,j}}$ for $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$ is used only for generating $v_{i,j,1\oplus z_{i,j}}$ in Hyb_0, Lemma A.1 directly follows from the security of PRG. Note that we can simulate $\mathsf{Vrfy}(\mathsf{vk}_0, \mathsf{cert})$ where $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i,j}$ since we need $\{z_{i,j}\}_{i,j}$ and $\{u_{i,j,b}\}_{i,j,b}$ such that $\theta_{i,j} = 1$ for verification. ☐

Lemma A.2. *If* FE *is adaptively secure,*

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

Proof. For each $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$, there is no u such that $\mathsf{PRG}(u) = v_{i,j,1\oplus z_{i,j}}$ except for probability $2^{-\lambda}$. Let Good be the event that the above holds for all $i \in [2n+1]$ and $j \in [\lambda]$. We have $\Pr[\mathsf{Good}] \geq 1 - (2n+1)\lambda 2^{-\lambda} = 1 - \mathsf{negl}(\lambda)$. We prove that whenever Good occurs, we have

$$\begin{aligned} & g[f]((V, \theta_1, \ldots, \theta_{2n+1}, \beta_1^{(0)}, \ldots, \beta_{2n+1}^{(0)}), (b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}) \\ =& g[f]((V, \theta_1, \ldots, \theta_{2n+1}, \beta_1^{(1)}, \ldots, \beta_{2n+1}^{(1)}), (b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}) \end{aligned} \tag{13}$$

for all key queries f and $(b_{i,j}, u_{i,j})_{i\in[2n+1], j\in[\lambda]}$ where

$$\beta_i^{(a)} := \begin{cases} m_i^{(a)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n] \\ 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j} & \text{if } i \in [n+1, 2n+1] \end{cases}$$

for $a \in \{0, 1\}$. If this is proven, Lemma A.2 directly follows from the adaptive security of FE.

Below, we prove Equation (13). We consider the following two cases.

- If $\mathsf{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ holds for every $i \in [2n+1]$ and $j \in [\lambda]$, then by the assumption that Good occurs, we have $b_{i,j} = z_{i,j}$ for all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. Then we have $\beta_i^{(a)} \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j} = m_i^{(a)}$ for $i \in [n]$ and $\beta_{2n+1}^{(a)} \oplus \bigoplus_{j:\theta_{2n+1,j}=0} b_{2n+1,j} = 0$ for $a \in \{0, 1\}$. Then the LHS of Equation (13) is equal to $f(m^{(0)})$ and the RHS of Equation (13) is equal to $f(m^{(1)})$. By the restriction on $\mathcal{A}$ in the adaptive security experiment, we have $f(m^{(0)}) = f(m^{(1)})$. Therefore, both sides of Equation (13) are equal to $f(m^{(0)}) = f(m^{(1)})$.
- Otherwise, both sides of Equation (13) are equal to $\perp$.

In either case, Equation (13) holds. Note that we can simulate $\mathsf{Vrfy}(\mathsf{vk}_b, \mathsf{cert})$ where $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i,j}$ since we need $\{z_{i,j}\}_{i,j}$ and $\{u_{i,j,b}\}_{i,j,b}$ such that $\theta_{i,j} = 1$ for verification. This completes the proof of Lemma A.2. ☐

Lemma A.3. *If* PRG *is a secure PRG,*

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

Proof. Noting that $u_{i,j,1\oplus z_{i,j}}$ for $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$ is used only for generating $v_{i,j,1\oplus z_{i,j}}$ in Hyb_3, Lemma A.1 directly follows from the security of PRG. Note that we can simulate $\mathsf{Vrfy}(\mathsf{vk}_1, \mathsf{cert})$ where $\mathsf{cert} = (c_{i,j}, d_{i,j})_{i,j}$ since we need $\{z_{i,j}\}_{i,j}$ and $\{u_{i,j,b}\}_{i,j,b}$ such that $\theta_{i,j} = 1$ for verification. ☐

B Adaptively Secure Public-Slot PKFE

In this section, we present an adaptively secure public-slot PKFE scheme based on

- Selectively secure PKFE,
- Selectively single-key function private SKFE, and
- Adaptively single-key single-ciphertext public-slot SKFE.

We need to show how to achieve adaptively single-key single-ciphertext public-slot SKFE since it is an essentail building block. Our adaptively secure public-slot PKFE scheme is presented in Appendix B.4.

We present an adaptively single-key single-ciphertext public-slot SKFE scheme based on

- Selectively single-key single-ciphertext public-slot SKFE and
- Receiver non-committing encryption

in Appendix B.3.

We also present a selectively secure single-ciphertext SKFE with public scheme based on IO and OWFs. This construction uses an MIFE scheme whose arity is 2 (i.e., 2-input FE) by Goldwasser et al. [GGG+14]. We introduce necessary tools and definitions in Appendices B.1 and B.2.

B.1 Building Blocks

We introduce building blocks for our adaptively single-key single-ciphertext public-slot SKFE scheme.

Non-committing encryption. We recall the notion of (secret-key) receiver non-committing encryption (NCE) [CFGN96, JL00, CHK05].

Definition B.1 (Secret-Key RNCE (Syntax)). *A secret-key NCE scheme is a tuple of PPT algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal})$ *with plaintext space* $\mathcal{M}$.

$\mathsf{KeyGen}(1^\lambda) \to (\mathsf{ek}, \mathsf{dk}, \mathsf{aux})$**:** *The key generation algorithm takes as input the security parameter* 1^λ *and outputs a key pair* $(\mathsf{ek}, \mathsf{dk})$ *and an auxiliary information* aux.

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

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

$\mathsf{Fake}(\mathsf{ek}, \mathsf{aux}) \to \widetilde{\mathsf{ct}}$**:** *The fake encryption algorithm takes* dk *and* aux, *and outputs a fake ciphertext* $\widetilde{\mathsf{ct}}$.

$\mathsf{Reveal}(\mathsf{ek}, \mathsf{aux}, \widetilde{\mathsf{ct}}, m) \to \widetilde{\mathsf{dk}}$**:** *The reveal algorithm takes* $\mathsf{ek}, \mathsf{aux}, \widetilde{\mathsf{ct}}$ *and* m, *and outputs a fake secret key* $\widetilde{\mathsf{dk}}$.

Definition B.2 (Correctness of secret-key NCE). *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,

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

Definition B.3 (Receiver Non-Committing (RNC) Security for SKE). *A secret-key NCE scheme is RNC secure if it satisfies the following. Let* $\Sigma = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Fake}, \mathsf{Reveal})$ *be a secret-key NCE scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{sk\text{-}rec}}_{\Sigma,\mathcal{A}} nc(\lambda, b)$.

1. *The challenger computes* $(\mathsf{ek}, \mathsf{dk}, \mathsf{aux}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and sends* 1^λ *to the adversary* $\mathcal{A}$.
2. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes and returns* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ *to* $\mathcal{A}$. *This process can be repeated polynomially many times.*
3. $\mathcal{A}$ *sends a query* $m \in \mathcal{M}$ *to the challenger.*
4. *The challenger does the following.*
   - *If* $b = 0$, *the challenger generates* $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}, m)$ *and returns* $(\mathsf{ct}, \mathsf{dk})$ *to* $\mathcal{A}$.
   - *If* $b = 1$, *the challenger generates* $\widetilde{\mathsf{ct}} \leftarrow \mathsf{Fake}(\mathsf{ek}, \mathsf{aux})$ *and* $\widetilde{\mathsf{dk}} \leftarrow \mathsf{Reveal}(\mathsf{ek}, \mathsf{aux}, \widetilde{\mathsf{ct}}, m)$ *and returns* $(\widetilde{\mathsf{ct}}, \widetilde{\mathsf{dk}})$ *to* $\mathcal{A}$.

5. *Again $\mathcal{A}$ can send encryption queries.*

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

Let $\mathsf{Adv}^{\mathsf{sk\text{-}rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda)$ be the advantage of the experiment above. We say that the Σ is RNC secure if for any QPT adversary, it holds that

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

Theorem B.4 ([KNTY19, Section 7.2 in the eprint version]). *If there exists an IND-CPA secure SKE scheme (against QPT adversaries), there exists an RNC secure secret-key NCE scheme (against QPT adversaries) with plaintext space $\{0,1\}^\ell$, where ℓ is some polynomial, respectively.*

Functional Encryption.

Definition B.5 (Public-Key FE (Syntax)). *A public-key functional encryption (PKFE) scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$, master public key space $\mathcal{MPK}$, master secret key space $\mathcal{MSK}$, and secret key space $\mathcal{SK}$, that work as follows.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{MPK}, \mathsf{MSK})$**:** *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 key generation algorithm takes MSK and $f \in \mathcal{F}$ as input, and outputs a secret key $\mathsf{sk}_f \in \mathcal{SK}$.*

$\mathsf{Enc}(\mathsf{MPK}, m) \to \mathsf{CT}$**:** *The encryption algorithm takes MPK and $m \in \mathcal{M}$ as input, and outputs 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 $\bot$.*

We require that a PKFE scheme satisfies correctness defined below.

Definition B.6 (Correctness of PKFE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ and $f \in \mathcal{F}$*

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

Definition B.7 (Selective Security of PKFE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a PKFE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{sel\text{-}ind}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

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

2. *The challenger runs $(\mathsf{MPK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, computes $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MPK}, m_b)$, and sends $(\mathsf{MPK}, \mathsf{CT})$ to $\mathcal{A}$.*

3. *$\mathcal{A}$ is allowed to make arbitrarily many key queries. 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 $\mathcal{A}$.*

4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. If $f_\ell(m_0) = f_\ell(m_1)$ holds for all key queries f_ℓ, the experiment outputs b'. Otherwise, it outputs $\bot$.*

We say that Σ is adaptively secure if for any QPT adversary $\mathcal{A}$ it holds that

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

Definition B.8 (Secret-Key FE (Syntax)). *A secret-key functional encryption (SKFE) scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$, master secret key space $\mathcal{MSK}$, and secret key space $\mathcal{SK}$, that work as follows.*

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

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

$\mathsf{Enc}(\mathsf{MSK}, m) \to \mathsf{CT}$**:** *The encryption algorithm takes MSK and $m \in \mathcal{M}$ as input, and outputs 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 $\bot$.*

We require that an SKFE scheme satisfies correctness defined below.

Definition B.9 (Correctness of SKFE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ and $f \in \mathcal{F}$*

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

Definition B.10 (Adaptive Single-Key Single-Ciphertext Security of SKFE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an SKFE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{ada\text{-}1key\text{-}1ct}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger runs $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *The adversary makes the following encryption query and key query in no particular order.*
   - *$\mathcal{A}$ sends $f \in \mathcal{F}$ to the challenger. The challenger computes $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f)$, and returns sk_f to $\mathcal{A}$. $\mathcal{A}$ can do this process one once.*
   - *$\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger. The challenger computes $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b)$, and returns CT to $\mathcal{A}$. $\mathcal{A}$ can do this process one once.*
3. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. If $f(m_0) = f(m_1)$ holds, the experiment outputs b'. Otherwise, it outputs $\bot$.*

We say that Σ is adaptively single-key single-ciphertext secure if for any QPT adversary $\mathcal{A}$ it holds that

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

Definition B.11 (Selective Single-Key Function Privacy of SKFE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be an SKFE scheme. We consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{sel\text{-}1key\text{-}fp}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger sends $(m_{1,0}, m_{1,1}), \ldots, (m_{q,0}, m_{q,1}) \in \mathcal{M}^{2q}$ to the challenger.*
2. *The challenger runs $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$, computes $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_{i,b})$ for all $i \in [q]$, and returns $(\mathsf{CT}_1, \ldots, \mathsf{CT}_q)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ sends $(f_0, f_1) \in \mathcal{F}^2$ to the challenger. The challenger computes $\mathsf{sk}_{f_b} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_b)$, and returns sk_{f_b} to $\mathcal{A}$. $\mathcal{A}$ can do this process only once.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. If $f_0(m_{i,0}) = f_1(m_{i,1})$ holds for all $i \in [q]$, the experiment outputs b'. Otherwise, it outputs $\bot$.*

We say that Σ is selectively single-key function private if for any QPT adversary $\mathcal{A}$ it holds that

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

2-input FE. We recall the notion of 2-input FE. The following definitions are special cases of multi-input functional encryption (MIFE) by Goldwasser et al. [GGG+14].

Definition B.12 (2-input FE (Syntax)). *A 2-input FE scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\mathcal{M}$, ciphertext space $\mathcal{C}$, master secret key space $\mathcal{MSK}$, and secret key space $\mathcal{SK}$, that work as follows.*

$\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{MSK}$**:** *The setup algorithm takes the security parameter 1^λ as input, and outputs a master secret key $\mathsf{MSK} \in \mathcal{MSK}$ and two encryption keys EK_1 and EK_2.*

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

$\mathsf{Enc}(\mathsf{EK}_i, x) \rightarrow \mathsf{CT}_i$**:** *The encryption algorithm takes EK_i and $x \in \mathcal{M}$ as input, and outputs a ciphertext $\mathsf{CT}_i \in \mathcal{C}$.*

$\mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}_1, \mathsf{CT}_2) \rightarrow z$ **or** $\bot$**:** *The decryption algorithm takes sk_f and $(\mathsf{CT}_1, \mathsf{CT}_2)$ as input, and outputs y or $\bot$.*

We require that a 2-input FE scheme satisfies correctness defined below.

Definition B.13 (Correctness of 2-input FE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $(x, y) \in \mathcal{M}^2$ and $f \in \mathcal{F}$*

$$\Pr\left[z \neq f(x,y) \;\middle|\; \begin{array}{l} (\mathsf{MSK}, \mathsf{EK}_1, \mathsf{EK}_2) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ \mathsf{CT}_1 \leftarrow \mathsf{Enc}(\mathsf{EK}_1, x), \mathsf{CT}_2 \leftarrow \mathsf{Enc}(\mathsf{EK}_2, y) \\ z \leftarrow \mathsf{Dec}(\mathsf{sk}_f, \mathsf{CT}_1, \mathsf{CT}_2) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

Definition B.14 ($(1,1)$-sel-ind Security of 2-FE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a 2-input FE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The adversary sends $((x_0, y_0), (x_1, y_1))$ to the challenger.*
2. *The challenger runs $(\mathsf{MSK}, \mathsf{EK}_1, \mathsf{EK}_2) \leftarrow \mathsf{Setup}(1^\lambda)$, computes $\mathsf{CT}_1 \leftarrow \mathsf{Enc}(\mathsf{EK}_1, x_b)$ and $\mathsf{CT}_2 \leftarrow \mathsf{Enc}(\mathsf{EK}_2, y_b)$, and sends $(\mathsf{EK}_2, \mathsf{CT}_1, \mathsf{CT}_2)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ can send a key query $f_i \in \mathcal{F}$ to the challenger. The challenger computes $\mathsf{sk}_{f_i} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f_i)$, and returns sk_{f_i} to $\mathcal{A}$. $\mathcal{A}$ can send unbounded polynomially many key queries. Let q_k be the total number of the key queries.*
4. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. If $f_i(x_0, y) = f_i(x_1, y)$ holds for all $y \in \mathcal{M}$ and $i \in [q_k]$ (we call $\mathcal{A}$ is valid), the experiment outputs b'. Otherwise, it outputs $\bot$.*

We say that Σ is $(1,1)$-sel-ind secure if for any QPT adversary $\mathcal{A}$ it holds that

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

Here, $(1,1)$ means that $\mathcal{A}$ is given one *encryption key EK_2 and* one *challenge ciphertext vector $(\mathsf{CT}_1, \mathsf{CT}_2)$.*

The security definition above is a special case of (t, q)-sel-ind security by Goldwasser et al. [GGG+14], where t is the number of corrupted encryption keys and q is the number of challenge ciphertext vectors.

Theorem B.15 ([GGG+14]). *If there exist IO and OWFs, there exists $(1,1)$-sel-ind secure 2-input FE for all polynomial-size circuits.*

Although Goldwasser et al. proved a more general theorem ((t, q)-sel-ind secure n-input FE where $t \leq n$ and q is an a-priori bounded polynomial), the simplified version above is sufficient for our purpose.

B.2 Variants of Security Definitions

We can consider the secret-key variant of Definition 3.9, where Setup generates only a master secret key MSK and Enc uses MSK instead of MPK. Correctness of public-slot SKFE is a natural extension of Definition 3.10. We omit syntax and correctness for public-slot SKFE.

Below, we introduce variants of security definitions for public-slot SKFE.

Definition B.16 (Single-Key Security of Public-Slot SKFE). *Let $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-slot SKFE scheme. We consider the following security experiment $\mathsf{Exp}^{\mathsf{ada\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger runs $\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$.*
2. *$\mathcal{A}$ is allowed to make arbitrarily many key queries. 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 $\mathcal{A}$.*
3. *$\mathcal{A}$ can send a challenge plaintext pair $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.*
4. *The challenger computes $\mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, m_b)$ and sends CT to $\mathcal{A}$.*
5. *Again, $\mathcal{A}$ is allowed to make arbitrarily many key queries.*
6. *$\mathcal{A}$ outputs $b' \in \{0,1\}$. If $f_\ell(m_0, \mathsf{pub}) = f_\ell(m_1, \mathsf{pub})$ holds for all key queries f_ℓ and public inputs $\mathsf{pub} \in \mathcal{P}$, the experiment outputs b'. Otherwise, it outputs $\perp$.*

We say that Σ is adaptively single-ciphertext secure if for any QPT adversary $\mathcal{A}$ it holds that

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

If the adversary must declare the challenge plaintext pair (m_0, m_1) at the very beginning of the experiment, we say that Σ is selectively single-ciphertext secure and denote the advantage and experiment by $\mathsf{Adv}^{\mathsf{sel\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda)$, respectively.

If the adversary is allowed to make only one key query, we say Σ is adaptively/selectively single-key single-ciphertext secure and denote the advantage and experiment by $\mathsf{Adv}^{\mathsf{xxx\text{-}1key\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda)$ and $\mathsf{Exp}^{\mathsf{xx\text{-}1key\text{-}1ct}}_{\Sigma,\mathcal{A}}(\lambda)$, respectively, where $\mathsf{xxx} \in \{\mathsf{sel}, \mathsf{ada}\}$.

B.3 Adaptively Single-Key Single-Ciphertext Public-Slot SKFE Scheme

Selectively single-ciphertext public-slot SKFE. First, we present our selectively single-ciphertext public-slot SKFE scheme $\mathsf{1selFE}$.

Ingredients.

- $(1,1)$-sel-ind secure 2-input FE $\mathsf{2FE} = \mathsf{2FE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for all polynomial-size circuits.

Scheme description. Our scheme $\mathsf{1selFE} = \mathsf{1selFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows.

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

1. Generate $(\mathsf{2fe.msk}, \mathsf{2fe.ek}_1, \mathsf{2fe.ek}_2) \leftarrow \mathsf{2FE.Setup}(1^\lambda)$.
2. Output $\mathsf{MSK} := (\mathsf{2fe.msk}, \mathsf{2fe.ek}_1, \mathsf{2fe.ek}_2)$.

$\mathsf{1selFE.KeyGen}(\mathsf{MSK}, f)$**:**

1. Parse $\mathsf{MSK} = (\mathsf{2fe.msk}, \mathsf{2fe.ek}_1, \mathsf{2fe.ek}_2)$.
2. Generate $\mathsf{2fe.SK}_f \leftarrow \mathsf{2FE.KeyGen}(\mathsf{2fe.msk}, f)$.

3. Output $\mathsf{SK}_f := \mathsf{2fe.sk}_f$.

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

1. Parse $\mathsf{MSK} = (\mathsf{2fe.msk}, \mathsf{2fe.ek}_1, \mathsf{2fe.ek}_2)$.
2. Generate $\mathsf{2fe.ct}_1 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.ek}_1, m)$.
3. Output $\mathsf{CT} := (\mathsf{2fe.ct}_1, \mathsf{2fe.ek}_2)$.

$\mathsf{1selFE.Dec}(\mathsf{SK}_f, \mathsf{CT}, \mathsf{pub})$**:**

1. Parse $\mathsf{SK}_f = \mathsf{2fe.sk}_f$ and $\mathsf{CT} = (\mathsf{2fe.ct}_1, \mathsf{2fe.ek}_2)$.
2. Compute $\mathsf{2fe.ct}_2 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.ek}_2, \mathsf{pub})$.
3. Compute and output $y := \mathsf{2FE.Dec}(\mathsf{2fe.sk}_f, \mathsf{2fe.ct}_1, \mathsf{2fe.ct}_2)$.

Correctness. It is easy to see correctness holds due to correctness of $\mathsf{2FE}$.

Theorem B.17. *If* $\mathsf{2FE}$ *is* $(1,1)$-sel-ind *secure* 2*-input FE for all polynomial-size circuits,* $\mathsf{1selFE}$ *is selectively single-ciphertext public-slot SKFE for all polynomial-size circuits.*

This theorem immediately yields a selectively single-key single-ciphertext public-slot SKFE for all polynomial-size circuits.

Proof. Let $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{1selFE},\mathcal{A}}(\lambda, b)$ denote the selective single-ciphertext security of public-slot SKFE. We construct an algorithm $\mathcal{B}$ that breaks $(1,1)$-sel-ind security of $\mathsf{2FE}$ by using an adversary $\mathcal{A}$ that breaks selectively single-ciphertext security of $\mathsf{1selFE}$. $\mathcal{B}$ does the following.

1. First, $\mathcal{A}$ sends (m_0, m_1). Then, $\mathcal{B}$ chooses a random $y \leftarrow \mathcal{M}$, sets $(x_0, x_1) := (m_0, m_1)$ and $(y_0, y_1) := (y, y)$, and passes $((x_0, y_0), (x_1, y_1))$ to its challenger.

2. When $\mathcal{B}$ receives $(\mathsf{2fe.ek}_2, \mathsf{2fe.ct}_1, \mathsf{2fe.ct}_2)$ from its challenger, $\mathcal{B}$ sets $\mathsf{CT} := (\mathsf{2fe.ct}_1, \mathsf{2fe.ek}_2)$ and passes CT to $\mathcal{A}$

3. When $\mathcal{A}$ sends a key query f_i, $\mathcal{B}$ passes f_i to its challenger, receives $\mathsf{2fe.sk}_{f_i} \leftarrow \mathsf{2FE.KeyGen}(\mathsf{2fe.msk}, f_i)$, and passes $\mathsf{SK}_{f_i} := \mathsf{2fe.sk}_{f_i}$ to $\mathcal{A}$.

4. When $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs b'.

If $\mathcal{A}$ is valid in the experiment of selective single-ciphertext security for public-slot SKFE $\mathsf{1selFE}$, it holds $f_i(x_0, y') = f_i(x_1, y')$ for all $i \in [q]$ and $y' \in \mathcal{M}$. Then, $\mathcal{B}$ is also a valid adversary in the experiment of $(1,1)$-sel-ind security for $\mathsf{2FE}$ since $\mathcal{B}$ received $\mathsf{2fe.ek}_2$. It is easy to see the following.

- If $\mathsf{2fe.ct}_1 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.msk}, x_0)$ and $\mathsf{2fe.ct}_2 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.msk}, y)$, $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{1selFE},\mathcal{A}}(\lambda, 0)$.

- If $\mathsf{2fe.ct}_1 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.msk}, x_1)$ and $\mathsf{2fe.ct}_2 \leftarrow \mathsf{2FE.Enc}(\mathsf{2fe.msk}, y)$, $\mathcal{B}$ perfectly simulates $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{1selFE},\mathcal{A}}(\lambda, 1)$.

Thus, if $\mathcal{A}$ distinguishes $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{1selFE},\mathcal{A}}(\lambda, 0)$ from $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{1selFE},\mathcal{A}}(\lambda, 1)$, $\mathcal{B}$ distinguishes $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{2FE},\mathcal{B}}(\lambda, 0)$ from $\mathsf{Exp}^{\mathsf{sel\text{-}1ct}}_{\mathsf{2FE},\mathcal{B}}(\lambda, 1)$. This completes the proof. ☐

Adaptively single-key single-ciphertext public-slot SKFE. Next, we present our adaptively single-key single-ciphertext public-slot SKFE scheme $\mathsf{1adaFE}$.

Ingredients.

- Selectively single-key single-ciphertext public-slot SKFE $\mathsf{1selFE} = \mathsf{1selFE.(Setup, KeyGen, Enc, Dec)}$ for all polynomial-size circuits.

- Receiver non-committing encryption $\mathsf{NCE} = \mathsf{NCE.(KeyGen, Enc, Dec, Fake, Reveal)}$.

Scheme description. Our scheme $\mathsf{1adaFE} = \mathsf{1adaFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows.

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

1. Generate $\mathsf{sel.msk} \leftarrow \mathsf{1selFE}.\mathsf{Setup}(1^\lambda)$.
2. Generate $(\mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE}.\mathsf{KeyGen}(1^\lambda)$.
3. Output $\mathsf{MSK} := (\mathsf{sel.msk}, \mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce.aux})$.

$\mathsf{1adaFE}.\mathsf{KeyGen}(\mathsf{MSK}, f)$**:**

1. Parse $\mathsf{MSK} = (\mathsf{sel.MSK}, \mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce.aux})$.
2. Generate $\mathsf{sel.sk}_f \leftarrow \mathsf{1selFE}.\mathsf{KeyGen}(\mathsf{sel.msk}, f)$.
3. Generate $\mathsf{nce.ct} \leftarrow \mathsf{NCE}.\mathsf{Enc}(\mathsf{nce.ek}, \mathsf{fe.sk}_f)$.
4. Output $\mathsf{SK}_f := \mathsf{nce.ct}$.

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

1. Parse $\mathsf{MSK} = (\mathsf{sel.msk}, \mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce.aux})$.
2. Generate $\mathsf{sel.ct} \leftarrow \mathsf{1selFE}.\mathsf{Enc}(\mathsf{sel.msk}, m)$.
3. Output $\mathsf{CT} := (\mathsf{sel.ct}, \mathsf{nce.dk})$.

$\mathsf{1adaFE}.\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT}, \mathsf{pub})$**:**

1. Parse $\mathsf{SK}_f = \mathsf{nce.ct}$ and $\mathsf{CT} = (\mathsf{sel.ct}, \mathsf{nce.dk})$.
2. Compute $\mathsf{sk}'_f \leftarrow \mathsf{NCE}.\mathsf{Dec}(\mathsf{nce.dk}, \mathsf{nce.ct})$.
3. Compute and output $y := \mathsf{1selFE}.\mathsf{Dec}(\mathsf{sk}'_f, \mathsf{sel.ct}, \mathsf{pub})$.

Correctness. It is easy to see correctness holds due to correctness of $\mathsf{1selFE}$ and NCE.

Theorem B.18. *If* $\mathsf{1selFE}$ *is selectively single-key single-ciphertext secure public-slot SKFE for all polynomial-size circuits and* NCE *is RNC secure,* $\mathsf{1adaFE}$ *is adaptively single-key single-ciphertext public-slot SKFE for all polynomial-size circuits.*

Proof. Let $\mathsf{Hyb}_0(b)$ denote $\mathsf{Exp}^{\mathsf{ada\text{-}1key\text{-}1ct}}_{\mathsf{1adaFE},\mathcal{A}}(\lambda, b)$. We define a hybrid game $\mathsf{Hyb}_1(b)$ as follows.

$\mathsf{Hyb}_1(b)$**:** This is the same as $\mathsf{Exp}^{\mathsf{ada\text{-}1key\text{-}1ct}}_{\mathsf{1adaFE},\mathcal{A}}(\lambda, b)$ except that:

1. if $\mathcal{A}$ sends a key query f before an encryption query (m_0, m_1), we generate $\mathsf{nce}.\widetilde{\mathsf{ct}} \leftarrow \mathsf{NCE}.\mathsf{Fake}(\mathsf{nce.ek}, \mathsf{nce.aux})$ instead of $\mathsf{nce.ct} \leftarrow \mathsf{NCE}.\mathsf{Enc}(\mathsf{nce.ek}, \mathsf{sel.sk}_f)$ and return $\mathsf{SK}_f := \mathsf{nce}.\widetilde{\mathsf{ct}}$ for the key query.
2. when $\mathcal{A}$ sends an encryption query (m_0, m_1) after the key query f above, we generate $\mathsf{sel.ct} \leftarrow \mathsf{1selFE}.\mathsf{Enc}(\mathsf{sel.msk}, m_b)$, $\mathsf{sel.sk}_f \leftarrow \mathsf{1selFE}.\mathsf{KeyGen}(\mathsf{sel.msk}, f)$, and $\mathsf{nce}.\widetilde{\mathsf{dk}} \leftarrow \mathsf{NCE}.\mathsf{Reveal}(\mathsf{nce.ek}, \mathsf{nce.aux}, \mathsf{nce}.\widetilde{\mathsf{ct}}, \mathsf{sel.sk}_f)$, and return $\mathsf{CT} := (\mathsf{sel.ct}, \mathsf{nce}.\widetilde{\mathsf{dk}})$ for the encryption query.

First, we show the following.

Proposition B.19. *It holds* $|\Pr[\mathsf{Hyb}_0(b) = 1] - \Pr[\mathsf{Hyb}_1(b) = 1]| \leq \mathsf{negl}(\lambda)$ *if* NCE *is RNC secure.*

We construct an algorithm $\mathcal{B}$ that breaks RNC security of NCE by using an adversary $\mathcal{A}$ that breaks adaptive single-key single-ciphertext security of $\mathsf{1adaFE}$. Note that if $\mathcal{A}$ sends an encryption query (m_0, m_1) before a key query f, these two games are the same. We focus on the case where $\mathcal{A}$ sends a key query f before an encryption query (m_0, m_1). $\mathcal{B}$ does the following.

1. First, $\mathcal{B}$ generates $\mathsf{sel.msk} \leftarrow \mathsf{1selFE}.\mathsf{Setup}(1^\lambda)$.

2. When $\mathcal{A}$ sends a key query f, $\mathcal{B}$ generates $\mathsf{sel.sk}_f \leftarrow \mathsf{1selFE.KeyGen}(\mathsf{sel.msk}, f)$, sends $\mathsf{sel.sk}_f$ to its challenger, and receives $(\mathsf{nce.ct}^*, \mathsf{nce.dk}^*)$. $\mathcal{B}$ passes $\mathsf{SK}_f := \mathsf{nce.ct}^*$ to $\mathcal{A}$.

3. After the key query f above, for an encryption query (m_0, m_1), $\mathcal{B}$ generates $\mathsf{sel.ct} \leftarrow \mathsf{1selFE.Enc}(\mathsf{sel.msk}, m_b)$ and returns $\mathsf{CT} := (\mathsf{sel.ct}, \mathsf{nce.dk}^*)$ to $\mathcal{A}$.

4. When $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs b'.

It is easy to see the following.

- If $\mathsf{nce.ct}^* \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.ek}, \mathsf{sel.sk}_f)$ and $\mathsf{nce.dk}^* = \mathsf{nce.dk}$ where $(\mathsf{nce.ek}.\mathsf{nce.dk}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE.KeyGen}(1^\lambda)$, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_0(b)$.
- If $\mathsf{nce.ct}^* := \mathsf{nce}.\widetilde{\mathsf{ct}} \leftarrow \mathsf{NCE.Fake}(\mathsf{nce.ek}, \mathsf{nce.aux})$ and $\mathsf{nce.dk}^* \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.ek}, \mathsf{nce.aux}, \mathsf{nce}.\widetilde{\mathsf{ct}}, \mathsf{sel.sk}_f)$, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_1(b)$.

Thus, if $\mathcal{A}$ distinguishes $\mathsf{Hyb}_0(b)$ from $\mathsf{Hyb}_1(b)$, $\mathcal{B}$ distinguishes $\mathsf{Exp}^{\mathsf{sk\text{-}rec\text{-}nc}}_{\mathsf{NCE},\mathcal{B}}(\lambda, b)$.

Next, we show the following.

Proposition B.20. *It holds* $|\Pr[\mathsf{Hyb}_1(0) = 1] - \Pr[\mathsf{Hyb}_1(1) = 1]| \leq \mathsf{negl}(\lambda)$ *if* $\mathsf{1selFE}$ *is selectively single-key single-ciphertext secure.*

We construct an algorithm $\mathcal{B}$ that breaks selective single-key single-ciphertext security of $\mathsf{1selFE}$ by using an adversary $\mathcal{A}$ that breaks adaptive single-key single-ciphertext security of $\mathsf{1adaFE}$. $\mathcal{B}$ does the following.

1. First, $\mathcal{B}$ generates $(\mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce.aux}) \leftarrow \mathsf{NCE.KeyGen}(1^\lambda)$.

2. There are the following two cases:
   - When $\mathcal{A}$ sends a key query f before an encryption query (m_0, m_1), $\mathcal{B}$ generates $\mathsf{nce}.\widetilde{\mathsf{ct}} \leftarrow \mathsf{NCE.Fake}(\mathsf{nce.ek}, \mathsf{nce.aux})$ passes $\mathsf{SK}_f := \mathsf{nce}.\widetilde{\mathsf{ct}}$ to $\mathcal{A}$. After the key query f above, for an encryption query (m_0, m_1), $\mathcal{B}$ passes (m_0, m_1) to its challenger and receives $\mathsf{sel.ct}^*$. Then, $\mathcal{B}$ sends f to its challenger and receives $\mathsf{sel.sk}_f \leftarrow \mathsf{1selFE}(\mathsf{sel.msk}, f)$. Finally, $\mathcal{B}$ generates $\mathsf{nce}.\widetilde{\mathsf{dk}} \leftarrow \mathsf{NCE.Reveal}(\mathsf{nce.ek}, \mathsf{nce.dk}, \mathsf{nce}.\widetilde{\mathsf{ct}}, \mathsf{sel.sk}_f)$ and sends $\mathsf{CT} := (\mathsf{sel.ct}^*, \mathsf{nce}.\widetilde{\mathsf{dk}})$ to $\mathcal{A}$.
   - When $\mathcal{A}$ sends an encryption query (m_0, m_1) before a key query f, $\mathcal{B}$ passes (m_0, m_1) to its challenger, receives $\mathsf{sel.ct}^*$, and returns $\mathsf{CT} := (\mathsf{sel.ct}^*, \mathsf{nce.dk})$ to $\mathcal{A}$. After the encryption query (m_0, m_1) above, for a key query f, $\mathcal{B}$ passes f to its challenger and receives $\mathsf{sel.sk}_f \leftarrow \mathsf{1selFE.KeyGen}(\mathsf{sel.msk}, f)$. $\mathcal{B}$ returns $\mathsf{SK}_f := \mathsf{nce.ct} \leftarrow \mathsf{NCE.Enc}(\mathsf{nce.ek}, \mathsf{sel.sk}_f)$ to $\mathcal{A}$.

3. When $\mathcal{A}$ outputs b', $\mathcal{B}$ outputs b'.

Note that $\mathcal{B}$ sends (m_0, m_1) to its challenger before it sends f as a key query in the both cases above. If $\mathcal{A}$ is a valid adversary in the experiment of adaptive single-key single-ciphertext security for public-lot SKFE $\mathsf{1adaFE}$, it holds $f(m_0, y') = f(m_1, y')$ for all $y' \in \mathcal{M}$. Then, $\mathcal{B}$ is also a valid adversary in the experiment of selective single-key single-ciphertext security for public-slot SKFE $\mathsf{1selFE}$. In addition, it is easy to see the following.

- If $\mathsf{sel.ct}^* \leftarrow \mathsf{1selFE.Enc}(\mathsf{sel.msk}, m_0)$, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_1(0)$.
- If $\mathsf{sel.ct}^* \leftarrow \mathsf{1selFE.Enc}(\mathsf{sel.msk}, m_1)$, $\mathcal{B}$ perfectly simulates $\mathsf{Hyb}_1(1)$.

Thus, if $\mathcal{A}$ distinguishes $\mathsf{Hyb}_1(0)$ from $\mathsf{Hyb}_1(1)$, $\mathcal{B}$ distinguishes $\mathsf{Exp}^{\mathsf{sel\text{-}1key\text{-}1ct}}_{\mathsf{1selFE},\mathcal{B}}(\lambda, 0)$ from $\mathsf{Exp}^{\mathsf{sel\text{-}1key\text{-}1ct}}_{\mathsf{1selFE},\mathcal{B}}(\lambda, 1)$.

Therefore, we obtain $|\Pr[\mathsf{Hyb}_0(0) = 1] - \Pr[\mathsf{Hyb}_0(1) = 1]| \leq \mathsf{negl}(\lambda)$, which is our goal. $\square$

B.4 Adaptively Secure Public-Slot PKFE Scheme

Note that the construction in this section is bassically the same as that by Ananth and Sahai [AS16] except that we use single-key single-ciphertext public-slot SKFE as a building block istead of single-key single-ciphertext standard SKFE.

Ingredients.

- Selectively secure PKFE $\mathsf{PKFE} = \mathsf{PKFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for all polynonial-size circuits.
- Selectively single-key function private SKFE $\mathsf{1KeySKFE} = \mathsf{1KeySKFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for polynomial-size circuits.
- Adaptively single-key single-ciphertext public-slot SKFE $\mathsf{SKFE} = \mathsf{SKFE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for polynomial-size circuits.
- A PRF $\mathsf{PRF} : \mathcal{K} \times \{0,1\}^\lambda \to \{0,1\}^\lambda$.
- SKE with pseudorandom ciphertext $\mathsf{SKE} = \mathsf{SKE}.(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$.

Scheme description. The adaptively secure public-slot PKFE scheme $\mathsf{FE} = \mathsf{FE}.(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is as follows.

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

1. Generate $(\mathsf{pkfe.MPK}, \mathsf{pkfe.MSK}) \leftarrow \mathsf{PKFE.Setup}(1^\lambda)$.
2. Output $\mathsf{MPK} := \mathsf{pkfe.MPK}$ and $\mathsf{MSK} := \mathsf{pkfe.MSK}$.

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

1. Parse $\mathsf{MSK} = \mathsf{pkfe.MSK}$.
2. Sample $C_{\mathsf{ske}} \leftarrow \{0,1\}^{\ell_{\mathsf{ske}}(\lambda)}$ where $\ell_{\mathsf{ske}}(\lambda)$ is the length of a SKE ciphertext that encrypts a string of length $\ell_{\mathsf{skfe}}(\lambda) + \ell_{\mathsf{1keyskfe}}(\lambda)$. We denote $\ell_{\mathsf{skfe}}(\lambda)$ by the length of a SKFE secret key and $\ell_{\mathsf{1keyskfe}}(\lambda)$ by the length of a 1KeySKFE ciphertext.
3. Sample $\tau \leftarrow \{0,1\}^{4\lambda}$
4. Generate $\mathsf{pkfe.sk}_{g[f, C_{\mathsf{ske}}, \tau]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f, C_{\mathsf{ske}}, \tau])$ where $g[f, C_{\mathsf{ske}}, \tau]$ is a function described in Figure 4.
5. Output $\mathsf{sk}_f = \mathsf{pkfe.sk}_{g[f, C_{\mathsf{ske}}, \tau]}$.

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

1. Parse $\mathsf{MPK} = \mathsf{pkfe.MPK}$.
2. Sample $\mathsf{K} \leftarrow \mathcal{K}$.
3. Generate $\mathsf{1keyskfe.MSK} \leftarrow \mathsf{1KeySKFE.Setup}(1^\lambda)$.
4. Generate $\mathsf{1keyskfe.sk}_{h[m]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}, h[m])$ where $h[m]$ is a function described in Figure 5.
5. Compute $\mathsf{pkfe.ct} \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\mathsf{1keyskfe.MSK}, \mathsf{K}, \perp, 0))$.
6. Output $\mathsf{ct} = (\mathsf{1keyskfe.sk}_{h[m]}, \mathsf{pkfe.ct})$.

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

1. Parse $\mathsf{sk}_f = \mathsf{pkfe.sk}_{g[f]}$ and $\mathsf{ct} = (\mathsf{1keyskfe.sk}_{h[m]}, \mathsf{pkfe.ct})$.
2. Compute $(\mathsf{skfe.sk}_f, \mathsf{1keyskfe.ct}) \leftarrow \mathsf{PKFE.Dec}(\mathsf{pkfe.sk}_{g[f]}, \mathsf{pkfe.ct})$.
3. Compute $\mathsf{skfe.ct} \leftarrow \mathsf{1KeySKFE.Dec}(\mathsf{1keyskfe.sk}_{h[m]}, \mathsf{1keyskfe.ct})$.
4. Compute $m' \leftarrow \mathsf{SKFE.Dec}(\mathsf{skfe.sk}_f, \mathsf{skfe.ct}, y)$
5. Output m'.

$\underline{g[f, C_{\mathsf{ske}}, \tau]}$

Input: $\mathsf{1keyskfe.MSK}, \mathsf{K}, \mathsf{ske.SK}, \beta$

1. Parse $\tau = (\tau_0 \| \tau_1 \| \tau_2 \| \tau_3)$.
2. If $\beta = 0$ then
   - Compute $R_i \leftarrow \mathsf{PRF}(\mathsf{K}, \tau_i)$ for $i \in \{0, 1, 2, 3\}$.
   - Generate $\mathsf{skfe.MSK} \leftarrow \mathsf{SKFE.Setup}(1^\lambda; R_0)$.
   - Compute $\mathsf{skfe.sk}_f \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{skfe.MSK}, f; R_1)$.
   - Compute $\mathsf{1keyskfe.ct} \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}, (\mathsf{skfe.MSK}, R_2, 0); R_3)$.
   - Output $(\mathsf{skfe.sk}_f, \mathsf{1keyskfe.ct})$.
3. Else,
   - Compute $(\mathsf{skfe.sk}_f, \mathsf{1keyskfe.ct}) \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.SK}, C_{\mathsf{ske}})$.
   - Output $(\mathsf{skfe.sk}_f, \mathsf{1keyskfe.ct})$.

Figure 4: The description of the function $g[f, C_E, \tau]$

$\underline{h[m]}$

Input: $\mathsf{skfe.MSK}, R, \alpha$

1. If $\alpha = 0$ then
   - Compute $\mathsf{skfe.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}, m; R)$.
   - Output $\mathsf{skfe.ct}$.
2. Else, output $\perp$.

Figure 5: The description of the function $h[m]$

The security proof is almost the same as that of Ananth and Sahai [AS16]. We provide the proof for confirmation since we use adaptively single-key single-ciphertext public-slot SKFE.

Theorem B.21. *If* PKFE *is selectively secure PKFE for* $\mathsf{P/poly}$*,* $\mathsf{1KeySKFE}$ *is selectively single-key function private SKFE for* $\mathsf{P/poly}$*,* SKFE *is adaptively single-key single-ciphertext public-slot SKFE for* $\mathsf{P/poly}$*,* PRF *is a secure PRF, and* SKE *is ciphertext pseudorandom, FE is adaptively indistinguishable-secure public-slot PKFE for* $\mathsf{P/poly}$*.*

We immediately obtain Theorem 3.12 from the thereom above.

Correctness. Let $\mathsf{ct} = (\mathsf{1keyskfe.sk}_{h[m]}, \mathsf{pkfe.ct})$ be an honestly generated ciphertext encrypting a message m and $\mathsf{sk}_f = \mathsf{pkfe.sk}_{g[f]}$ be an honestly generated secret key corresponding to a function f. Firstly, we note that $\mathsf{pkfe.ct}$ is an encryption of the message $(\mathsf{1keyskfe.MSK}, \mathsf{K}, \perp, 0)$ and $g[f]$ is a function that takes $(\mathsf{1keyskfe.MSK}, \mathsf{K}, \perp, 0)$ as input and outputs a SKFE secret key $\mathsf{skfe.sk}_f$ corresponding to the function f and a single key SKFE ciphertext $\mathsf{1keyskfe.ct}$. Therefore, by the correctness of PKFE, the decryption algorithm $\mathsf{PKFE.Dec}(\mathsf{pkfe.sk}_{g[f]}, \mathsf{pkfe.ct})$ yields $g[f](\mathsf{1keyskfe.MSK}, \mathsf{K}, \perp, 0) = (\mathsf{skfe.sk}_f, \mathsf{1keyskfe.ct})$. Secondly, we observe that $\mathsf{1keyskfe.ct}$ encrypts a message $(\mathsf{SKFE.MSK}, R_2, 0)$ and $h[m]$ is a function that takes $(\mathsf{SKFE.MSK}, R_2, 0)$ an input and outputs a SKFE ciphertext $\mathsf{skfe.ct}$. Therefore, by the correctness of SKFE, the decryption algorithm $\mathsf{1KeySKFE.Dec}(\mathsf{1keyskfe.sk}_{h[m]}, \mathsf{1keyskfe.ct})$ yields $h[m](\mathsf{SKFE.MSK}, R_2, 0) = \mathsf{skfe.ct}$. Finally, we note that $\mathsf{skfe.ct}$ encrypts the message m and f is a function that takes (m, y) as input and outputs $f(m, y)$ where y is an input to the public slot. Thus, by the correctness of public-slot SKFE, we obtain $\mathsf{SKFE.Dec}(\mathsf{skfe.sk}_f, \mathsf{skfe.ct}, y) = m' = f(m, y)$.

Adaptive Security. We prove Theorem B.21.

Proof of Theorem B.21. Let $\mathcal{A}$ be a PPT adversary against the adaptive security of the public-slot PKFE. We use the following sequence of hybrids to prove the security. Let $\Pr[\mathsf{Hyb}_i = 1]$ be the winning probability of $\mathcal{A}$ in Hyb_i for all i.

Hyb_0**:** This is the original adaptive security experiment where the challenge bit set to 0. Specifically, it works as follows:

1. The challenger generates $(\mathsf{pkfe.MPK}, \mathsf{pkfe.MSK}) \leftarrow \mathsf{PKFE.Setup}(1^\lambda)$, sets $\mathsf{MPK} := \mathsf{pkfe.MPK}$ and $\mathsf{MSK} := \mathsf{pkfe.MSK}$, and sends MPK to $\mathcal{A}$.
2. The challenger samples a PRF key $\mathsf{K}^* \leftarrow \mathcal{K}$ and generate a master secret key $\mathsf{1keyskfe.MSK}^* \leftarrow \mathsf{1KeySKFE.Setup}(1^\lambda)$.
3. The challenger computes $\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\mathsf{1keyskfe.MSK}^*, \mathsf{K}^*, \perp, 0))$.
4. $\mathcal{A}$ can make arbitrarily many key queries at any point of the experiment. When it makes the j-th key query for a function f_j, the challenger works as follows:
   (a) Sample $C_{j,\mathsf{ske}}, \tau_j = (\tau_{j,0} \| \tau_{j,1} \| \tau_{j,2} \| \tau_{j,3})$ uniformly at random.
   (b) Generate $\mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f_j, C_{j,\mathsf{ske}}, \tau_j])$ where $g[f_j, C_{j,\mathsf{ske}}, \tau_j]$ is a function described in Figure 4.
   (c) Set $\mathsf{sk}_{f_j} := \mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$.

   The challenger sends sk_{f_j} to $\mathcal{A}$.
5. $\mathcal{A}$ sends (m_0, m_1) to the challenger. It must satisfy $f(m_0, y) = f(m_1, y)$ for any public input y and for all key queries f that are made before or after sending (m_0, m_1).
6. The challenger computes the ciphertext as follows:
   (a) Generate $\mathsf{1keyskfe.sk}^*_{h[m_0]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}^*, h[m_0])$ where $h[m_0]$ is a function described in Figure 5.
   (b) Set $\mathsf{ct}^* := (\mathsf{1keyskfe.sk}^*_{h[m_0]}, \mathsf{pkfe.ct}^*)$ where $\mathsf{pkfe.ct}^*$ is computed in Step 3.

   The challenger sends ct^* to $\mathcal{A}$.
7. $\mathcal{A}$ outputs a bit b' which is the final output of the experiment.

Note that, the challenger can sample a PRF key K^*, a master secret key for the single key function-private SKFE $\mathsf{1keyskfe.MSK}^*$ before it answers any secret key query. Moreover, the challenger can also compute the part of the challenge ciphertext $\mathsf{pkfe.ct}^*$ before the key query phase.

Hyb_1**:** This hybrid is identical to Hyb_0 except the challenger samples a SKE key $\mathsf{ske.SK}^*$ before it answers any key query and sets $C_{j,\mathsf{ske}}$ to be the ciphertext of SKE which corresponds to the challenge ciphertext. More specifically, the challenger answers to the j-th key query for a function f_j as follows:

(a) Sample $\tau_j = (\tau_{j,0} \| \tau_{j,1} \| \tau_{j,2} \| \tau_{j,3})$ uniformly at random.

(b) Compute $R_{j,i} = \mathsf{PRF}(\mathsf{K}^*, \tau_{j,i})$ for all $i \in \{0, 1, 2, 3\}$.

(c) Generate $\mathsf{skfe.MSK}_j \leftarrow \mathsf{SKFE.Setup}(1^\lambda; R_{j,0})$.

(d) Compute $\mathsf{skfe.sk}_{f_j} \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{skfe.MSK}_j, f_j; R_{j,1})$.

(e) Compute $\mathsf{1keyskfe.ct}_j \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_j, R_{j,2}, 0); R_{j,3})$.

(f) Compute $C_{j,\mathsf{ske}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.SK}^*, u_j)$ where $u_j = (\mathsf{skfe.sk}_{f_j}, \mathsf{1keyskfe.ct}_j)$.

(g) Generate $\mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f_j, C_{j,\mathsf{ske}}, \tau_j])$ where $g[f_j, C_{j,\mathsf{ske}}, \tau_j]$ is a function described in Figure 4.

(h) Set $\mathsf{sk}_f := \mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$.

The challenger sends sk_{f_j} to $\mathcal{A}$. The indistinguishability between Hyb_0 and Hyb_1 follows from the security of SKE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{ske.SK}^*$ and using the challenger of the security experiment of SKE. In particular, consider $\mathcal{B}_1$ to be an adversary against the security of SKE. When $\mathcal{A}$ queries a secret key for a function f_j, $\mathcal{B}_1$ proceeds as in Step (a) to (f) and sends the message $u_j = (\mathsf{skfe.sk}_{f_j}, \mathsf{1keyskfe.ct}_j)$ to it's challenger. Upon receiving a ciphertext $C_{j,\mathsf{ske}}$ from the challenger, $\mathcal{B}_1$ computes $\mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$ and sends it to $\mathcal{A}$. If $\mathcal{B}_1$ receives a random string then it simulates Hyb_0, otherwise, if $\mathcal{B}_1$ is sent an encryption of u_j then it simulates Hyb_1. Therefore, the wining probability of $\mathcal{B}_1$ is the same as $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]| \leq \mathsf{negl}$. Hence, by the security of SKE, it holds that

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

Hyb_2**:** This hybrid is identical to Hyb_1 except the challenger computes

$$\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\bot, \bot, \mathsf{ske.SK}^*, 1))$$

instead of computing $\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\mathsf{1keyskfe.MSK}^*, \mathsf{K}^*, \bot, 0))$. In particular, the mode of decryption is changed to $\beta = 1$ from $\beta = 0$ meaning that $\mathsf{ske.SK}^*$ is used to decrypt $C_{j,\mathsf{ske}}$ to get an output of $(\mathsf{skfe.sk}_{f_j}, \mathsf{1keyskfe.ct}_j)$ while decryption of ct^* is performed by the j-th secret key $\mathsf{sk}_f := \mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$. The indistinguishability between Hyb_1 and Hyb_2 follows from the security of PKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{pkfe.MSK}$ and using the challenger of the security experiment of PKFE. Let us consider an adversary $\mathcal{B}_2$ against the security of PKFE. Firstly, $\mathcal{B}_2$ sends a pair of challenge message $((\mathsf{1keyskfe.MSK}^*, \mathsf{K}^*, \bot, 0), (\bot, \bot, \mathsf{ske.SK}^*, 1))$ to it's challenger and receives the public key $\mathsf{pkfe.MPK}$ and a ciphertext $\mathsf{pkfe.ct}^*$. Note that $\mathcal{B}_2$ can choose the challenge message independent of all the queries of $\mathcal{A}$. Therefore, a selectively secure PKFE is sufficient for arguing the indistinguishability between the hybrids. Whenever $\mathcal{B}_2$ receives a secret key query from $\mathcal{A}$ for a function f_j, it queries for a secret key to it's challenger for a function $g[f_j, C_{j,\mathsf{ske}}, \tau_j]$ and returns the output to $\mathcal{A}$. Firstly, $\mathcal{B}_2$ is an admissible adversary as

$$g[f_j, C_{j,\mathsf{ske}}, \tau_j](\mathsf{1keyskfe.MSK}^*, \mathsf{K}^*, \bot, 0) = g[f_j, C_{j,\mathsf{ske}}, \tau_j](\bot, \bot, \mathsf{ske.SK}^*, 1)$$

holds for all j. If $\mathcal{B}_2$ receives an encryption of $(\mathsf{1keyskfe.MSK}^*, \mathsf{K}^*, \bot, 0)$ then it simulates Hyb_1, otherwise, if $\mathcal{B}_2$ receives an encryption of $(\bot, \bot, \mathsf{ske.SK}^*, 1)$ then it simulates Hyb_2. Therefore, the winning probability of $\mathcal{B}_2$ is essentially the same as $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| \leq \mathsf{negl}$. Hence, by the security of PKFE, it holds that

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

$\underline{h[m, m', v]}$

Input: $\mathsf{skfe.MSK}, R, \alpha$

1. If $\alpha = 0$ then
   - Compute $\mathsf{skfe.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}, m; R)$.
   - Output $\mathsf{skfe.ct}$.
2. If $\alpha = 1$ then
   - Compute $\mathsf{skfe.ct} \leftarrow \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}, m'; R)$.
   - Output $\mathsf{skfe.ct}$.
3. Else, output v.

Figure 6: The description of the function $h[m, m', v]$

Hyb_3**:** This hybrid is identical to Hyb_2 except the challenger samples $R_{j,i}$ uniformly at random for all j, i, while answering the secret key queries instead of computing these values using the PRF key K^*. The indistinguishability between Hyb_2 and Hyb_3 follows from the security of PRF since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of K^* and using the challenger of the security experiment of PRF. In other words, if $\mathcal{B}_3$ is an adversary against the security of PRF then the winning probability of $\mathcal{B}_3$ is the same as $|\Pr[\mathsf{Hyb}_2 = 1] - \Pr[\mathsf{Hyb}_3 = 1]| \leq \mathsf{negl}$. Hence, by the security of PRF, it holds that

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

Hyb_4**:** This hybrid is identical to Hyb_3 except the challenger generates

$$\mathsf{1keyskfe.sk}^*_{h[m_0,m_1,v]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}^*, h[m_0, m_1, v])$$

and sets the challenge ciphertext as $\mathsf{ct}^* := (\mathsf{1keyskfe.sk}^*_{h[m_0,m_1,v]}, \mathsf{pkfe.ct}^*)$ where $h[m_0, m_1, v]$ is a function described in Figure 6 and v is a random string. The indistinguishability between Hyb_3 and Hyb_4 follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of 1KeySKFE. Let us consider an adversary $\mathcal{B}_4$ against the security of 1KeySKFE. We assume that Q be the total number of secret key queries the adversary $\mathcal{A}$ makes in the experiment. At first, $\mathcal{B}_4$ prepares a list of Q challenge messages $(M_1, \ldots, M_Q)$ where $M_j = (\mathsf{skfe.MSK}_j, R_{j,2}, 0)$ for $j \in [Q]$. More precisely, $\mathcal{B}$ sends (M_j, M_j) for all $j \in [Q]$ and receives the ciphertexts as $\{\mathsf{1keyskfe.ct}_j\}_{j\in[Q]}$ which is used in answering $\mathcal{A}$'s secret key queries as in Hyb_3 or Hyb_4. When $\mathcal{A}$ sends the challenge message tuple (m_0, m_1), $\mathcal{B}_4$ queries for a secret key with the pair of functions $(h[m_0], h[m_0, m_1, v])$ to it's challenger. Then, $\mathcal{B}_4$ uses the output from it's challenger to create the challenge ciphertext for $\mathcal{A}$. It is easy to see that $\mathcal{B}_4$ is an admissible adversary for the security experiment of 1KeySKFE since

$$h[m_0](\mathsf{skfe.MSK}_j, R_{j,2}, 0) = h[m_0, m_1, v](\mathsf{skfe.MSK}_j, R_{j,2}, 0)$$

holds for all $j \in [Q]$. If $\mathcal{B}_4$ receives a secret key $\mathsf{1keyskfe.sk}^*_{h[m_0]}$ then it simulates Hyb_3, otherwise, if $\mathcal{B}_4$ receives a secret key $\mathsf{1keyskfe.sk}^*_{h[m_0,m_1,v]}$ then it simulates Hyb_4. Therefore, the winning probability of $\mathcal{B}_4$ is essentially the same as $|\Pr[\mathsf{Hyb}_3 = 1] - \Pr[\mathsf{Hyb}_4 = 1]| \leq \mathsf{negl}$. Hence, by the security of 1KeySKFE, it holds that

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

Hyb_5**:** This hybrid is identical to Hyb_4 except the challenger computes

$$\mathsf{1keyskfe.ct}_j \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_j, R_{j,2}, 1); R_{j,3})$$

while generating the j-th secret key corresponding to a function f_j for all j. In Lemma B.22, we show that

$$|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \leq \mathsf{negl}(\lambda).$$

Hyb_6**:** This hybrid is identical to Hyb_5 except the challenger generates

$$\mathsf{1keyskfe.sk}^*_{h[m_1,m_1,v]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}^*, h[m_1, m_1, v])$$

and sets the challenge ciphertext as $\mathsf{ct}^* := (\mathsf{1keyskfe.sk}^*_{h[m_1,m_1,v]}, \mathsf{pkfe.ct}^*)$ where $h[m_1, m_1, v]$ is a function as described in Figure 6. The indistinguishability between Hyb_5 and Hyb_6 follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of 1KeySKFE. The simulation strategy is similar to Hyb_4. By the security of 1KeySKFE, it holds that

$$|\Pr[\mathsf{Hyb}_5 = 1] - \Pr[\mathsf{Hyb}_6 = 1]| \leq \mathsf{negl}(\lambda).$$

Hyb_7**:** This hybrid is identical to Hyb_6 except the challenger computes

$$\mathsf{1keyskfe.ct}_j \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_j, R_{j,2}, 0); R_{j,3})$$

while generating the j-th secret key corresponding to a function f_j for all j. Moreover, the challenger generates

$$\mathsf{1keyskfe.sk}^*_{h[m_1]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}^*, h[m_1])$$

and sets the challenge ciphertext as $\mathsf{ct}^* := (\mathsf{1keyskfe.sk}^*_{h[m_1]}, \mathsf{pkfe.ct}^*)$ where $h[m_1]$ is a function as described in Figure 5. The indistinguishability between Hyb_6 and Hyb_7 follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of 1KeySKFE. Let us consider an adversary $\mathcal{B}_7$ against the security of 1KeySKFE.

At first, $\mathcal{B}_7$ prepares a list of Q challenge message pairs $((M_1^{(0)}, M_1^{(1)}), \ldots, (M_Q^{(0)}, M_Q^{(1)}))$ where

$$M_j^{(0)} = (\mathsf{skfe.MSK}_j, R_{j,2}, 1) \quad \text{and} \quad M_j^{(1)} = (\mathsf{skfe.MSK}_j, R_{j,2}, 0)$$

for $j \in [Q]$. More precisely, $\mathcal{B}_7$ sends $(M_j^{(0)}, M_j^{(1)})$ for all $j \in [Q]$ and receives the ciphertexts as $\{\mathsf{1keyskfe.ct}_j\}_{j\in[Q]}$ which is used in answering $\mathcal{A}$'s secret key queries. When $\mathcal{A}$ sends the challenge message tuple (m_0, m_1), $\mathcal{B}_7$ queries for a secret key with the pair of functions $(h[m_1, m_1, v], h[m_1])$ to it's challenger. Then, $\mathcal{B}_7$ uses the output from it's challenger to create the challenge ciphertext for $\mathcal{A}$. It is easy to see that $\mathcal{B}_7$ is an admissible adversary for the security experiment of 1KeySKFE since

$$h[m_1, m_1, v](\mathsf{skfe.MSK}_j, R_{j,2}, 1) = h[m_1](\mathsf{skfe.MSK}_j, R_{j,2}, 0)$$

holds for all $j \in [Q]$. If $\mathcal{B}_7$ receives ciphertexts for the messages $M_j^{(0)}$ and a secret key $\mathsf{1keyskfe.sk}^*_{h[m_1,m_1,v]}$ then it simulates Hyb_6, otherwise, if $\mathcal{B}_7$ receives ciphertexts for the messages $M_j^{(1)}$ and a secret key $\mathsf{1keyskfe.sk}^*_{h[m_1]}$ then it simulates Hyb_7. Therefore, the winning probability of $\mathcal{B}_7$ is essentially the same as $|\Pr[\mathsf{Hyb}_6 = 1] - \Pr[\mathsf{Hyb}_7 = 1]| \leq \mathsf{negl}$. Hence, by the security of 1KeySKFE, it holds that

$$|\Pr[\mathsf{Hyb}_6 = 1] - \Pr[\mathsf{Hyb}_7 = 1]| \leq \mathsf{negl}(\lambda).$$

Hyb_8: This hybrid is identical to Hyb_7 except the challenger samples

$$R_{j,i} = \mathsf{PRF}(\mathsf{K}^*, \tau_{j,i}) \text{ for all } i \in \{0,1,2,3\}$$

instead of sampling these values uniformly at random for all j, i, while answering the secret key queries. The indistinguishability between Hyb_7 and Hyb_8 follows from the security of PRF since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of K^* and using the challenger of the security experiment of PRF. In other words, it holds that

$$|\Pr[\mathsf{Hyb}_7 = 1] - \Pr[\mathsf{Hyb}_8 = 1]| \le \mathsf{negl}(\lambda).$$

Hyb_9: This hybrid is identical to Hyb_8 except the challenger computes

$$\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (1\mathsf{keyskfe.MSK}^*, \mathsf{K}^*, \perp, 0))$$

instead of computing $\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\perp, \perp, \mathsf{ske.SK}^*, 1))$. The indistinguishability between Hyb_8 and Hyb_9 follows from the security of PKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{pkfe.MSK}$ and using the challenger of the security experiment of PKFE. In other words, it holds that

$$|\Pr[\mathsf{Hyb}_8 = 1] - \Pr[\mathsf{Hyb}_9 = 1]| \le \mathsf{negl}(\lambda).$$

Hyb_{10}: This hybrid is identical to Hyb_9 except the challenger does not sample $\mathsf{ske.SK}^*$ and chooses $C_{j,\mathsf{ske}}$ uniformly at random while answering to the j-th secret key query of $\mathcal{A}$ for all j. More specifically, the challenger answers to the j-th key query for a function f_j as follows:

(a) Sample $C_{j,\mathsf{ske}}, \tau_j = (\tau_{j,0} \| \tau_{j,1} \| \tau_{j,2} \| \tau_{j,3})$ uniformly at random.

(b) Generate $\mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f_j, C_{j,\mathsf{ske}}, \tau_j])$ where $g[f_j, C_{j,\mathsf{ske}}, \tau_j]$ is a function described in Figure 4.

(c) Set $\mathsf{sk}_{f_j} := \mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$.

The challenger sends sk_{f_j} to $\mathcal{A}$. The indistinguishability between Hyb_9 and Hyb_{10} follows from the security of SKE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{ske.SK}$ and using the challenger of the security experiment of SKE. In other words, it holds that

$$|\Pr[\mathsf{Hyb}_9 = 1] - \Pr[\mathsf{Hyb}_{10} = 1]| \le \mathsf{negl}(\lambda).$$

We observe that Hyb_{10} is identical to original adaptive security experiment where the challenge bit set to 1. Combining the advantage of $\mathcal{A}$ in all the consecutive hybrids and applying the triangular inequality, we have $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_{10} = 1]| \le \mathsf{negl}(\lambda)$.

This completes the proof of Theorem B.21 if we prove Lemma B.22. □

Lemma B.22. *If* 1KeySKFE *is selectively single key function-private secure and public-slot SKFE* SKFE *is adaptively single-key single-ciphertext secure then for any* $\lambda \in [n]$,

$$|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \le \mathsf{negl}(\lambda).$$

Proof of Lemma B.22. We prove this lemma using a sequence of hybrids $\mathsf{Hyb}_{4,q,1}, \mathsf{Hyb}_{4,q,2}, \mathsf{Hyb}_{4,q,3}, \mathsf{Hyb}_{4,q,4}$ for $q \in [Q]$ where Q denotes the total number of secret key queried made by the adversary $\mathcal{A}$. Let us denote $\mathsf{Hyb}_{4,Q+1,1} = \mathsf{Hyb}_5$.

$\mathsf{Hyb}_{4,q,1}$: This is exactly the same as Hyb_4 except the challenger sets v to be the output of $\mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_0; R_{q,2})$. More precisely, the hybrid works as follows:

1. The challenger generates $(\mathsf{pkfe.MPK}, \mathsf{pkfe.MSK}) \leftarrow \mathsf{PKFE.Setup}(1^\lambda)$, sets $\mathsf{MPK} := \mathsf{pkfe.MPK}$ and $\mathsf{MSK} := \mathsf{pkfe.MSK}$, and sends MPK to $\mathcal{A}$.
2. The challenger generates a master secret key $\mathsf{1keyskfe.MSK}^* \leftarrow \mathsf{1KeySKFE.Setup}(1^\lambda)$.
3. The challenger generates a secret key $\mathsf{ske.SK}^* \leftarrow \mathsf{SKE.Setup}(1^\lambda)$.
4. The challenger computes $\mathsf{pkfe.ct}^* \leftarrow \mathsf{PKFE.Enc}(\mathsf{pkfe.MPK}, (\perp, \perp, \mathsf{ske.SK}^*, 1))$.
5. The challenger sets u_q as follows:
   (a) Sample $\tau_q = (\tau_{q,0} \| \tau_{q,1} \| \tau_{q,2} \| \tau_{q,3})$ and $R_{q,i}$ uniformly at random for all $i \in \{0, 1, 2, 3\}$.
   (b) Generate $\mathsf{skfe.MSK}_q \leftarrow \mathsf{SKFE.Setup}(1^\lambda; R_{q,0})$.
   (c) Compute $\mathsf{skfe.sk}_{f_q} \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{skfe.MSK}_q, f_q; R_{q,1})$.
   (d) Compute $\mathsf{1keyskfe.ct}_q \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_q, R_{q,2}, 0); R_{q,3})$.
   (e) Set $u_q = (\mathsf{skfe.sk}_{f_q}, \mathsf{1keyskfe.ct}_q)$.
6. $\mathcal{A}$ can make arbitrarily many key queries at any point of the experiment. When it makes the j-th key query for a function f_j, the challenger works as follows:

   If $j \neq q$:
   (a) Sample $\tau_j = (\tau_{j,0} \| \tau_{j,1} \| \tau_{j,2} \| \tau_{j,3})$ and $R_{j,i}$ uniformly at random for all $i \in \{0, 1, 2, 3\}$.
   (b) Generate $\mathsf{skfe.MSK}_j \leftarrow \mathsf{SKFE.Setup}(1^\lambda; R_{j,0})$.
   (c) Compute $\mathsf{skfe.sk}_{f_j} \leftarrow \mathsf{SKFE.KeyGen}(\mathsf{skfe.MSK}_j, f_j; R_{j,1})$.
   (d) Compute
   $$\begin{array}{lll} \mathsf{1keyskfe.ct}_j \leftarrow & \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_j, R_{j,2}, 1); R_{j,3}) & \text{if } j < q \\ \mathsf{1keyskfe.ct}_j \leftarrow & \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.MSK}_j, R_{j,2}, 0); R_{j,3}) & \text{if } j > q \end{array}$$
   (e) Compute $C_{j,\mathsf{ske}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.SK}^*, u_j)$ where $u_j = (\mathsf{skfe.sk}_{f_j}, \mathsf{1keyskfe.ct}_j)$.
   (f) Generate $\mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f_j, C_{j,\mathsf{ske}}, \tau_j])$ where $g[f_j, C_{j,\mathsf{ske}}, \tau_j]$ is a function described in Figure 4.
   (g) Set $\mathsf{sk}_f := \mathsf{pkfe.sk}_{g[f_j, C_{j,\mathsf{ske}}, \tau_j]}$.

   If $j = q$:
   (a) Set $C_{q,\mathsf{ske}} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.SK}^*, u_q)$.
   (b) Generate $\mathsf{pkfe.sk}_{g[f_q, C_{q,\mathsf{ske}}, \tau_q]} \leftarrow \mathsf{PKFE.KeyGen}(\mathsf{pkfe.MSK}, g[f_q, C_{q,\mathsf{ske}}, \tau_q])$ where $g[f_q, C_{q,\mathsf{ske}}, \tau_q]$ is a function described in Figure 4.

   The challenger sends sk_{f_j} to $\mathcal{A}$.
7. $\mathcal{A}$ sends (m_0, m_1) to the challenger. It must satisfy $f(m_0, y) = f(m_1, y)$ for any public input y and for all key queries f that are made before or after sending (m_0, m_1).
8. The challenger computes the ciphertext as follows:
   (a) Set $v := \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_0; R_{q,2})$
   (b) Generate $\mathsf{1keyskfe.sk}^*_{h[m_0, m_1, v]} \leftarrow \mathsf{1KeySKFE.KeyGen}(\mathsf{1keyskfe.MSK}^*, h[m_0, m_1, v])$ where $h[m_0, m_1, v]$ is a function described in Figure 6.
   (c) Set $\mathsf{ct}^* := (\mathsf{1keyskfe.sk}^*_{h[m_0, m_1, v]}, \mathsf{pkfe.ct}^*)$ where $\mathsf{pkfe.ct}^*$ is computed in Step 3.

   The challenger sends ct^* to $\mathcal{A}$.
9. $\mathcal{A}$ outputs a bit b' which is the final output of the experiment.

The indistinguishability between Hyb_4 and $\mathsf{Hyb}_{4,1,1}$ follows from the security of $\mathsf{1KeySKFE}$ since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of $\mathsf{1KeySKFE}$. Let us consider an adversary $\mathcal{B}_{4,1}$ against the security of $\mathsf{1KeySKFE}$.

At first, $\mathcal{B}_{4,1}$ prepares a list of Q challenge message pairs $(M_1, \ldots, M_Q)$ where

$$\begin{array}{rll} M_j = & (\mathsf{skfe.MSK}_j, R_{j,2}, 1) & \text{if } 1 \leq j < q \\ M_j = & (\mathsf{skfe.MSK}_j, R_{j,2}, 0) & \text{if } q \leq j \leq Q. \end{array}$$

for $j \in [Q]$. More precisely, $\mathcal{B}_{4,1}$ sends (M_j, M_j) for all $j \in [Q]$ and receives the ciphertexts as $\{\mathsf{1keyskfe.ct}_j\}_{j\in[Q]}$ which is used in answering $\mathcal{A}$'s secret key queries. When $\mathcal{A}$ sends the challenge message tuple (m_0, m_1), $\mathcal{B}_{4,1}$ queries for a secret key with the pair of functions $(h[m_0, m_1, v], h[m_0, m_1, v'])$ to it's challenger where v is a random string of appropriate length and $v' = \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_0; R_{q,2})$. Then, $\mathcal{B}_{4,1}$ uses the output from it's challenger to create the challenge ciphertext for $\mathcal{A}$. It is easy to see that $\mathcal{B}_{4,1}$ is an admissible adversary for the security experiment of $\mathsf{1KeySKFE}$ since $h[m_0, m_1, v](M_j) = h[m_0, m_1, v'](M_j)$ holds for all $j \in [Q]$. This is because $h[m_0, m_1, v](*, *, k) = h[m_0, m_1, v'](*, *, k)$ if $k \neq 2$. If $\mathcal{B}_{4,1}$ receives a secret key $\mathsf{1keyskfe.sk}^*_{h[m_0,m_1,v]}$ then it simulates Hyb_4, otherwise, if $\mathcal{B}_{4,1}$ receives a secret key $\mathsf{1keyskfe.sk}^*_{h[m_0,m_1,v']}$ then it simulates $\mathsf{Hyb}_{4,1,1}$. Therefore, the winning probability of $\mathcal{B}_{4,1}$ is essentially the same as $\left|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_{4,1,1} = 1]\right| \leq \mathsf{negl}$. Hence, by the security of $\mathsf{1KeySKFE}$, it holds that

$$\left|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_{4,1,1} = 1]\right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{Hyb}_{4,q,2}$: This is exactly the same as $\mathsf{Hyb}_{4,q,1}$ except the challenger changes the mode from $\alpha = 0$ to $\alpha = 2$ while decrypting the challenge ciphertext using the q-th secret key. More precisely, the challenger computes $\mathsf{1keyskfe.ct}_q$ as follows:

$$\mathsf{1keyskfe.ct}_q \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (0, 0, 2); R_{q,3}).$$

The indistinguishability between $\mathsf{Hyb}_{4,q,1}$ and $\mathsf{Hyb}_{4,q,2}$ follows from the security of $\mathsf{1KeySKFE}$ since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of $\mathsf{1KeySKFE}$. This can be shown similarly as we discussed in the previous hybrid since

$$h[m_0, m_1, v](\mathsf{skfe.MSK}_q, R_{q,2}, 0) = h[m_0, m_1, v](0, 0, 2)$$

holds where $v = \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_0; R_{q,2})$. Hence, by the security of $\mathsf{1KeySKFE}$, it holds that

$$\left|\Pr\left[\mathsf{Hyb}_{4,q,1} = 1\right] - \Pr\left[\mathsf{Hyb}_{4,q,2} = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{Hyb}_{4,q,3}$: This is exactly the same as $\mathsf{Hyb}_{4,q,2}$ except the challenger changes v to be the encryption of m_1, that is, it sets v as follows:

$$v := \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_1; R_{q,2})$$

The indistinguishability between $\mathsf{Hyb}_{4,q,2}$ and $\mathsf{Hyb}_{4,q,3}$ follows from the security of SKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{skfe.MSK}_q$ and using the challenger of the security experiment of SKFE. Let us consider an adversary $\mathcal{B}_{4,3}$ against the security of adaptively single-key single-ciphertext secure public-slot SKFE. Note that, $\mathcal{B}_{4,3}$ queries only a single secret key $\mathsf{skfe.sk}_{f_q}$ corresponding to the function f_q and a single ciphertext v corresponding to the challenge message pair (m_0, m_1). In particular, $\mathcal{B}_{4,3}$ can adaptively query the secret key $\mathsf{skfe.sk}_{f_q}$ at any point whenever $\mathcal{A}$ asks for a secret key for f_q and sets $u_q = (\mathsf{skfe.sk}_{f_q}, \mathsf{1keyskfe.ct}_q)$. We observe that $\mathcal{B}_{4,3}$ is an admissible adversary since $\mathcal{A}$ is only allowed to query for a secret key for f_q and challenge message pair (m_0, m_1) such that $f_q(m_0, y) = f_q(m_1, y)$ holds for is an arbitrary input y to the public slot of f. If $\mathcal{B}_{4,3}$ receives a ciphertext $v = \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_0)$ then it simulates $\mathsf{Hyb}_{4,q,2}$, otherwise, if $\mathcal{B}_{4,3}$ receives a ciphertext $v = \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_1)$ then it simulates $\mathsf{Hyb}_{4,q,3}$. Therefore, the winning probability of $\mathcal{B}_{4,3}$ is essentially the same as $\left|\Pr\left[\mathsf{Hyb}_{4,q,2} = 1\right] - \Pr\left[\mathsf{Hyb}_{4,q,3} = 1\right]\right| \leq$ negl. Hence, by the security of SKFE, it holds that

$$\left|\Pr\left[\mathsf{Hyb}_{4,q,2} = 1\right] - \Pr\left[\mathsf{Hyb}_{4,q,3} = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{Hyb}_{4,q,4}$: This is exactly the same as $\mathsf{Hyb}_{4,q,3}$ except the challenger changes the mode from $\alpha = 2$ to $\alpha = 1$ while decrypting the challenge ciphertext using the q-th secret key. More precisely, the challenger computes $\mathsf{1keyskfe.ct}_q$ as follows:

$$\mathsf{1keyskfe.ct}_q \leftarrow \mathsf{1KeySKFE.Enc}(\mathsf{1keyskfe.MSK}^*, (\mathsf{skfe.ct}_q, R_{q,2}; 1); R_{q,3}).$$

The indistinguishability between $\mathsf{Hyb}_{4,q,3}$ and $\mathsf{Hyb}_{4,q,4}$ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of $\mathsf{1keyskfe.MSK}^*$ and using the challenger of the security experiment of 1KeySKFE. This can be shown similarly as we discussed in the previous hybrid since

$$h[m_0, m_1, v](0, 0, 2) = h[m_0, m_1, v](\mathsf{skfe.MSK}_q, R_{q,2}, 1)$$

holds where $v = \mathsf{SKFE.Enc}(\mathsf{skfe.MSK}_q, m_1; R_{q,2})$. Hence, by the security of 1KeySKFE, it holds that

$$\left|\Pr\left[\mathsf{Hyb}_{4,q,3} = 1\right] - \Pr\left[\mathsf{Hyb}_{4,q,4} = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Combining the advantage of $\mathcal{A}$ in all the consecutive hybrids and applying the triangular inequality, we have $|\Pr[\mathsf{Hyb}_4 = 1] - \Pr[\mathsf{Hyb}_5 = 1]| \leq \mathsf{negl}(\lambda)$. This completes the proof of Lemma B.22. □

C Secret and Public Key Encryption with Certified Everlasting Deletion

In Appendix C.1, we define SKE and PKE with certified everlasting deletion. In Appendix C.2 and Appendix C.3, we construct a certified everlasting secure SKE scheme with and without QROM, respectively. In Appendix C.4 and Appendix C.5, we construct a certified everlasting secure PKE scheme with and without QROM, respectively.

C.1 Definition

Definition C.1 (SKE with Certified Everlasting Deletion (Syntax)). *Let λ be a security parameter and let p, q, r and s be some polynomials. An SKE with certified everlasting deletion scheme is a tuple of algorithms $\Sigma = (\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ 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 $\mathsf{sk} \in \mathcal{SK}$.*

$\mathcal{E}nc(\mathsf{sk}, m) \to (\mathsf{vk}, ct)$: *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 $ct \in \mathcal{C}$.*

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

$\mathcal{D}el(\mathsf{ct}) \to \mathsf{cert}$: *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** $\perp$: *The verification algorithm takes vk and cert as input, and outputs $\top$ or $\perp$.*

Remark C.2. Although we consider quantum certificates in Appendix C.3, we consider classical certificates by default. In the quantum certificate case, we need to use $cert$ and $\mathcal{V}rfy$ in the syntax.

We require that an SKE with certified everlasting deletion scheme satisfies correctness defined below.

Definition C.3 (Correctness for SKE with Certified Everlasting Deletion). *There are three types of correctness, namely, decryption correctness, verification correctness, and special correctness.*

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

$$\Pr\left[m' \neq m \middle| \begin{array}{l} \mathsf{sk} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{vk}, ct) \leftarrow \mathcal{E}nc(\mathsf{sk}, m) \\ m' \leftarrow \mathcal{D}ec(\mathsf{sk}, ct) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

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

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

Minimum requirements for correctness are decryption correctness and verification correctness. However, we also require special correctness and verification correctness with QOTP in this work because we need special correctness for the construction of the garbling scheme in Appendix E.2, and verification correctness with QOTP for the construction of FE in Section 4.3.

Definition C.4 (Special Correctness). *There exists a negligible function* negl *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$*,*

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

Definition C.5 (Verification Correctness with QOTP). *There exists a negligible function* negl *and a PPT algorithm* $\mathsf{Recover}$ *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$*,*

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

As security, we consider two definitions, Definition C.6 and Definition C.7 given below. The former is just the standard 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 C.6 (IND-CPA Security for SKE with Certified Everlasting Deletion). *Let* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be an SKE with certified everlasting deletion 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* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{sk}, m)$*, and returns* (vk, 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* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{sk}, m_b)$*, and sends* ct *to* $\mathcal{A}$*.*
5. $\mathcal{A}$ *sends an encryption query* m *to the challenger. The challenger computes* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{sk}, m)$*, and returns* (vk, 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 C.7 (Certified Everlasting IND-CPA Security for SKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *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_i *to the challenger. The challenger computes* $(\mathsf{vk}_i, ct_i) \leftarrow \mathcal{Enc}(\mathsf{sk}, m_i)$, *and returns* (vk_i, ct_i) *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* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{sk}, m_b)$, *and sends* ct *to* $\mathcal{A}_1$.
5. $\mathcal{A}_1$ *sends an encryption query* m_i *to the challenger. The challenger computes* $(\mathsf{vk}_i, ct_i) \leftarrow \mathsf{Enc}(\mathsf{sk}, m_i)$, *and returns* (vk_i, ct_i) *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*

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

Definition C.8 (PKE with Certified Everlasting Deletion (Syntax)). *Let* λ *be a security parameter and let* p, q, r, s *and* t *be polynomials. A PKE with certified everlasting deletion scheme is a tuple of algorithms* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *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* $\mathcal{D} := \mathcal{Q}^{\otimes t(\lambda)}$.

$\mathsf{KeyGen}(1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$**:** *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* $\mathsf{sk} \in \mathcal{SK}$.

$\mathcal{Enc}(\mathsf{pk}, m) \rightarrow (\mathsf{vk}, ct)$**:** *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* $ct \in \mathcal{C}$.

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

$\mathcal{Del}(ct) \rightarrow \mathsf{cert}$**:** *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* $\bot$.

Remark C.9. Although we consider quantum certificates in Appendix C.5, we consider classical certificates by default. In the quantum certificate case, we need to use $\mathcal{cert}$ and $\mathcal{Vrfy}$ in the syntax.

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

Definition C.10 (Correctness for PKE with Certified Everlasting Deletion). *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* $m \in \mathcal{M}$,

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

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

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

Minimum requirements for correctness are decryption correctness and verification correctness. However, we also require verification correctness with QOTP in this work because we need it for the construction of FE in Section 4.3.

Definition C.11 (Verification Correctness with QOTP). *There exists a negligible function* negl *and a PPT algorithm* $\mathsf{Recover}$ *such that for any* $\lambda \in \mathbb{N}$ *and* $m \in \mathcal{M}$*,*

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

As security, we consider two definitions, Definition C.12 and Definition C.13 given below. The former is just the standard 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 C.12 (IND-CPA Security for PKE with Certified Everlasting Deletion). *Let* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be a PKE with certified everlasting deletion 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* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{pk}, m_b)$*, 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 C.13 (Certified Everlasting IND-CPA Security for PKE). *Let* $\Sigma = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ *be a PKE with certified everlasting deletion 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* $(\mathsf{vk}, ct) \leftarrow \mathcal{Enc}(\mathsf{pk}, m_b)$*, 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* $\bot$*, the challenger outputs* $\bot$*, and sends* $\bot$ *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* $\bot$*.*

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*

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

C.2 SKE Scheme with QROM

In this section, we construct an SKE with certified everlasting deletion scheme with QROM. Our construction is similar to that of the certified everlasting commitment scheme in [HMNY22b]. The difference is that we use SKE instead of commitment.

Our certified everlasting secure SKE scheme. We construct a certified everlasting secure SKE scheme $\Sigma_{\mathsf{cesk}} = (\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ from the following primitives.

- A one-time SKE with certified deletion scheme (Definition 2.19) $\Sigma_{\mathsf{skcd}} = \mathsf{CD}.(\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$.
- 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 $\mathsf{sk} := \mathsf{ske.sk}$.

$\mathcal{E}nc(\mathsf{sk}, m)$**:**

- 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 $\mathsf{cd}.ct \leftarrow \mathsf{CD}.\mathcal{E}nc(\mathsf{cd.sk}, m)$.
- Output $ct := (h, \mathsf{ske.ct}, \mathsf{cd}.ct)$ and $\mathsf{vk} := \mathsf{cd.sk}$.

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

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

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

- Parse $ct = (h, \mathsf{ske.ct}, \mathsf{cd}.ct)$.
- Compute $\mathsf{cd.cert} \leftarrow \mathsf{CD}.\mathcal{D}el(\mathsf{cd}.ct)$.
- 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 $b \leftarrow \mathsf{CD.Vrfy}(\mathsf{cd.sk}, \mathsf{cd.cert})$.
- Output b.

Correctness: It is easy to see that correctness of Σ_{cesk} comes from those of Σ_{sk} and Σ_{skcd}. Special correctness holds due to that of Σ_{sk}. Verifcation correctness with QOTP holds due to that of Σ_{skcd}.

Security: The following two theorems hold.

Theorem C.14. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.11) and Σ_{skcd} satisfies the OT-CD security (Definition 2.19), Σ_{cesk} satisfies the IND-CPA security (Definition C.6).*

Its proof is similar to that of Theorem C.15, and therefore we omit it.

Theorem C.15. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.11) and Σ_{skcd} satisfies the OT-CD security (Definition 2.19), Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition C.7).*

Its proof is similar to that of [HMNY22b, Theorem 5.8].

C.3 SKE Scheme without QROM

In this section, we construct an SKE with certified everlasting deletion scheme without QROM. Note that unlike the construction with QROM (Appendix C.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 secure 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 secure SKE scheme $\Sigma_{\mathsf{cesk}} = (\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ 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.

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

$\mathcal{E}nc(\mathsf{sk}, m)$:

- 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 $\mathsf{ske.ct} \leftarrow \mathsf{SKE.Enc}(\mathsf{ske.sk}, (B, Q, r, y))$.
- Let U_Q be the unitary that permutes the qubits in Q into the first half of the system. (I.e., $U_Q |x_1 x_2 \cdots x_{p+q}\rangle = \left|x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\right\rangle$ 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 $ct := (|\Psi\rangle, \mathsf{ske.ct}, u, h)$ and $\mathsf{vk} := (B, Q, r)$.

$\mathcal{D}ec(\mathsf{sk}, ct)$:

- Parse $\mathsf{sk} = \mathsf{ske.sk}$, $ct = (|\Psi\rangle, \mathsf{ske.ct}, u, h)$.

- Compute $(B,Q,r,y)/\bot \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk},\mathsf{ske.ct})$. If $\bot \leftarrow \mathsf{SKE.Dec}(\mathsf{ske.sk},\mathsf{ske.ct})$, $\mathsf{Dec}(\mathsf{sk},\mathsf{ct})$ 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 $m' := h \oplus x \oplus y$.

$\mathsf{Del}(ct)$:

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

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

- Parse $\mathsf{vk} = (B,Q,r)$ and $cert = |\Psi\rangle$.
- 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}. Special correctness holds due to that of Σ_{sk}. Verifcation correctness with QOTP holds since Recover is the decryption algorithm of QOTP.

Security. The following two theorems hold.

Theorem C.16. *If Σ_{sk} is IND-CPA secure (Definition 2.12), then Σ_{cesk} is IND-CPA secure (Definition C.6).*

Its proof is straightforward, so we omit it.

Theorem C.17. *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 C.7).*

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.

C.4 PKE Scheme with QROM

In this section, we construct a certified everlasting secure PKE scheme with QROM. Our construction is similar to that of the certified everlasting commitment scheme in [HMNY22b]. The difference is that we use PKE instead of commitment.

Our certified everlasting secure PKE scheme. We construct a certified everlasting secure PKE scheme $\Sigma_{\mathsf{cepk}} = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ from a one-time SKE with certified deletion scheme $\Sigma_{\mathsf{skcd}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ (Definition 2.19), a PKE scheme $\Sigma_{\mathsf{pk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with plaintext space $\{0,1\}^\lambda$ (Definition 2.15) 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 $\mathsf{sk} := \mathsf{pke.sk}$.

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

- 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 $\mathsf{ske}.ct \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}, m)$.
- Output $ct := (h, \mathsf{ske}.ct, \mathsf{pke.ct})$ and $\mathsf{vk} := \mathsf{ske.sk}$.

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

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

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

- Parse $ct = (h, \mathsf{ske}.ct, \mathsf{pke.ct})$.
- Compute $\mathsf{ske.cert} \leftarrow \mathsf{SKE}.\mathcal{Del}(\mathsf{ske}.ct)$.
- 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}. Verifcation correctness with QOTP holds due to that of Σ_{skcd}.

Security: The following two theorems hold. Their proofs are similar to those of Theorems C.14 and C.15, and therefore we omit them.

Theorem C.18. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.17) and Σ_{skcd} satisfies the OT-CD security (Definition 2.22), Σ_{cepk} is IND-CPA secure (Definition C.12).*

Theorem C.19. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.17) and Σ_{skcd} satisfies the OT-CD security (Definition 2.22), Σ_{cepk} is certified everlasting IND-CPA secure (Definition C.13).*

C.5 PKE Scheme without QROM

In this section, we construct a certified everlasting secure 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 secure 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 secure PKE scheme $\Sigma_{\mathsf{cepk}} = (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$, 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 2.15) $\Sigma_{\mathsf{pk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ 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 $\mathsf{sk} := \mathsf{pke.sk}$.

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

- 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., $U_Q \left|x_1 x_2 \cdots x_{p+q}\right\rangle = \left|x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\right\rangle$ with $Q := \{a_1, a_2, \cdots, a_p\}$ and $\{1, 2, \cdots, p+q\} \setminus Q := \{b_1, b_2, \cdots, b_q\}$.)
- 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 $ct := (|\Psi\rangle, \mathsf{pke.ct}, u, h)$ and $\mathsf{vk} := (B, Q, r)$.

$\mathcal{Dec}(\mathsf{sk}, ct)$:

- Parse $\mathsf{sk} = \mathsf{pke.sk}$ and $ct = (|\Psi\rangle, \mathsf{pke.ct}, u, h)$.
- 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 $m' := h \oplus x \oplus y$.

$\mathcal{Del}(ct)$:

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

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

- Parse $\mathsf{vk} = (B, Q, r)$ and $cert = |\Psi\rangle$.
- 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}. Verifcation correctness with QOTP holds since Recover is the decryption algorithm of QOTP.

Security. The following two theorems hold.

Theorem C.20. *If Σ_{pk} is IND-CPA secure (Definition 2.18), then Σ_{cepk} is IND-CPA secure (Definition C.12).*

Its proof is straightforward, and therefore we omit it.

Theorem C.21. *If Σ_{pk} is IND-CPA secure (Definition 2.18) and $tp/(p+q) - 4(k_1 - k_2)\ln 2$ is superlogarithmic, then Σ_{cepk} is certified everlasting IND-CPA secure (Definition C.13).*

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.

D Receiver Non-Committing Encryption with Certified Everlasting Deletion

In this section, we define and construct receiver non-committing encryption with certified everlasting deletion. In Appendix D.1, we define RNCE with certified everlasting deletion. In Appendix D.2, we construct a certified everlasting RNCE scheme from certified everlasting secure PKE (Appendix C).

D.1 Definition

Definition D.1 (RNCE with Certified Everlasting Deletion (Syntax)). *Let λ be the security parameter and let p, q, r, s, t, u, and v be polynomials. An RNCE with certified everlasting deletion scheme is a tuple of algorithms $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{F}ake, \mathsf{Reveal}, \mathcal{D}el, \mathsf{Vrfy})$ 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 $\mathcal{AUX} := \{0,1\}^{v(\lambda)}$.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pk}, \mathsf{MSK})$**:** *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 $\mathsf{MSK} \in \mathcal{MSK}$.*

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

$\mathcal{E}nc(\mathsf{pk}, m) \to (\mathsf{vk}, ct)$**:** *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 $ct \in \mathcal{C}$.*

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

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

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

$\mathcal{D}el(ct) \to \mathsf{cert}$**:** *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 an RNCE with certified everlasting deletion scheme satisfies correctness defined below.

Definition D.2 (Correctness for RNCE with Certified Everlasting Deletion). *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* $m \in \mathcal{M}$,

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

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

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

As security, we consider two definitions, Definition D.3 and Definition D.4 given below. The former is just the standard 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 D.3 (Receiver Non-Committing Security for RNCE with Certified Everlasting Deletion). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Fake}, \mathsf{Reveal}, \mathcal{Del}, \mathsf{Vrfy})$ *be an RNCE with certified everlasting deletion scheme. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{rec\text{-}nc}}_{\Sigma,\mathcal{A}}(\lambda, b)$ *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* $(\mathsf{vk}, \mathit{ct}) \leftarrow \mathcal{Enc}(\mathsf{pk}, m)$ *and* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK})$, *and sends* $(\mathit{ct}, \mathsf{sk})$ *to* $\mathcal{A}$.
   - *If* $b = 1$, *the challenger generates* $(\mathsf{vk}, \widetilde{\mathit{ct}}, \mathsf{aux}) \leftarrow \mathcal{Fake}(\mathsf{pk})$ *and* $\widetilde{\mathsf{sk}} \leftarrow \mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m)$, *and sends* $(\widetilde{\mathit{ct}}, \widetilde{\mathsf{sk}})$ *to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs* $b' \in \{0, 1\}$.

We say that Σ *is receiver non-committing (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 D.4 (Certified Everlasting RNC Security for RNCE). *Let* $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Fake}, \mathsf{Reveal}, \mathcal{Del}, \mathsf{Vrfy})$ *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* $(\mathsf{vk}, \mathit{ct}) \leftarrow \mathcal{Enc}(\mathsf{pk}, m)$ *and* $\mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{MSK})$, *and sends* $(\mathit{ct}, \mathsf{sk})$ *to* $\mathcal{A}_1$.
   - *If* $b = 1$, *the challenger generates* $(\mathsf{vk}, \widetilde{\mathit{ct}}, \mathsf{aux}) \leftarrow \mathcal{Fake}(\mathsf{pk})$ *and* $\widetilde{\mathsf{sk}} \leftarrow \mathsf{Reveal}(\mathsf{pk}, \mathsf{MSK}, \mathsf{aux}, m)$, *and sends* $(\widetilde{\mathit{ct}}, \widetilde{\mathsf{sk}})$ *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*

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

D.2 Construction

In this section, we construct a certified everlasting RNCE scheme from a certified everlasting PKE scheme (Definition C.8). Our construction is similar to that of the secret-key RNCE scheme presented in [KNTY19]. The difference is that we use a certified everlasting secure PKE scheme instead of a standard SKE scheme.

Our certified everlasting secure RNCE scheme. We construct a certified everlasting secure RNCE scheme $\Sigma_{\mathsf{cence}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Fake}, \mathsf{Reveal}, \mathcal{Del}, \mathsf{Vrfy})$ from a certified everlasting secure PKE scheme $\Sigma_{\mathsf{cepk}} = \mathsf{PKE}.(\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$, which was introduced in Definition C.8.

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

- 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 $\mathsf{sk} := (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]})$.

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

- Parse $\mathsf{pk} = \{\mathsf{pke.pk}_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke}.ct_{i,\alpha}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,\alpha}, m[i])$ 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 $ct := \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.

$\mathcal{Dec}(\mathsf{sk}, ct)$:

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

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

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

$\mathcal{Del}(ct)$**:**

- Parse $ct = \{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$.
- Compute $\mathsf{pke.cert}_{i,\alpha} \leftarrow \mathsf{PKE}.\mathcal{Del}(\mathsf{pke}.ct_{i,\alpha})$ 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 D.5. *If Σ_{cepk} is IND-CPA secure (Definition C.12), Σ_{cence} is RNC secure (Definition D.3).*

Its proof is similar to that of Theorem D.6, and therefore we omit it.

Theorem D.6. *If Σ_{cepk} is certified everlasting IND-CPA secure (Definition C.13), Σ_{cence} is certified everlasting RNC secure (Definition D.4).*

Proof of Theorem D.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 $(\mathsf{pke.vk}_{i,\alpha}, \mathsf{pke}.ct_{i,\alpha}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,\alpha}, m[i])$ 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 $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke}.ct_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i] \oplus 1)$ for all $i \in [n]$ instead of computing $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke}.ct_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{Enc}(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i])$ 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 $(\mathsf{pke.vk}_{i,x^*[i]}, \mathsf{pke}.ct_{i,x^*[i]}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,x^*[i]}, 0)$ and $(\mathsf{pke.vk}_{i,x^*[i]\oplus 1}, \mathsf{pke}.ct_{i,x^*[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,x^*[i]\oplus 1}, 1)$ instead of computing $(\mathsf{pke.vk}_{i,x[i]}, \mathsf{pke}.ct_{i,x[i]}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,x[i]}, m[i])$ and $(\mathsf{pke.vk}_{i,x[i]\oplus 1}, \mathsf{pke}.ct_{i,x[i]\oplus 1}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,x[i]\oplus 1}, m[i] \oplus 1)$.
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 D.6 easily follows from the following Propositions D.7 and D.8 (whose proof is given later.). □

Proposition D.7. *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 D.8. $|\Pr[\mathsf{Hyb}_1 = 1] - \Pr[\mathsf{Hyb}_2 = 1]| \leq \mathsf{negl}(\lambda)$.

Proof of Proposition D.7. For the proof, we use Lemma D.9. 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 D.9. This contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma D.9. 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 $(\{\mathsf{pke.vk}_{i,x[i]}\}_{i\in[n]}, \{\mathsf{pke}.ct_{i,x[i]}\}_{i\in[n]}) \leftarrow \mathsf{PKE}.\mathcal{E}nc(\mathsf{pke.pk}_{i,x[i]}, m[i])$ 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 $\perp$. If $\mathcal{B}$ receives $\perp$, it outputs $\perp$ 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 $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\mathsf{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. By assumption, $|\Pr[\mathsf{Hyb}_0 = 1] - \Pr[\mathsf{Hyb}_1 = 1]|$ is non-negligible. Therefore, $|\Pr[1 \leftarrow \mathcal{B} \mid b = 0] - \Pr[1 \leftarrow \mathcal{B} \mid b = 1]|$ is also non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma D.9. □

Proof of Proposition D.8. It is obvious that the joint probability distribution that $\mathcal{A}_1$ receives $(\{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x, \{\mathsf{pke.sk}_{i,x[i]}\}_{i\in[n]}))$ in Hyb_1 is identical to the joint probability distribution that $\mathcal{A}_1$ receives $(\{\mathsf{pke}.ct_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}, (x^* \oplus m, \{\mathsf{pke.sk}_{i,x^*[i]\oplus m[i]}\}_{i\in[n]}))$ in Hyb_2. Hence, Proposition D.8 follows. □

We use the following lemma for the proof of Theorem D.6 and Theorem E.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 D.9. *Let s be some polynomial of the security parameter λ. Let $\Sigma := (\mathsf{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \mathsf{Vrfy})$ be a certified everlasting secure PKE scheme. Let us consider the following security experiment $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{multi\text{-}cert\text{-}ever}}(\lambda, b)$ 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 $(\mathsf{vk}_{i,f[i]\oplus 1}, ct_{i,f[i]\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_b[i])$ for all $i \in [s]$, and sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{ct_{i,f[i]\oplus 1}\}_{i\in[s]})$ 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,

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

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

Proof of Lemma D.9. 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
$$(\mathsf{vk}_{i,f[i]\oplus 1}, ct_{i,f[i]\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_1[i])$$
for $i \in [j]$ and
$$(\mathsf{vk}_{i,f[i]\oplus 1}, ct_{i,f[i]\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{pk}_{i,f[i]\oplus 1}, m_0[i])$$
for $i \in \{j+1, j+2, \cdots, s\}$, and sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{ct_{i,f[i]\oplus 1}\}_{i\in[s]})$ 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 $\Pr[\mathsf{Hyb}_0 = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda,0) = 1\right]$ and $\Pr[\mathsf{Hyb}_s = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{multi\text{-}cert\text{-}ever}}_{\Sigma,\mathcal{A}}(\lambda,1) = 1\right]$. Furthermore, we can show

$$\left|\Pr\left[\mathsf{Hyb}_j = 1\right] - \Pr\left[\mathsf{Hyb}_{j+1} = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

for each $j \in \{0,1,\cdots,s-1\}$. (Its proof is given below.) From these facts, we obtain Lemma D.9.

Let us show the remaining one. To show it, let us assume that $\left|\Pr\left[\mathsf{Hyb}_j = 1\right] - \Pr\left[\mathsf{Hyb}_{j+1} = 1\right]\right|$ 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 $(\mathsf{pk}_{j+1,\beta\oplus1},\mathsf{sk}_{j+1,\beta\oplus1}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
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
$$(\mathsf{vk}_{i,f[i]\oplus1}, ct_{i,f[i]\oplus1}) \leftarrow \mathcal{E}nc(\mathsf{pk}_{i,f[i]\oplus1}, m_1[i])$$
for $i \in [j]$ and
$$(\mathsf{vk}_{i,f[i]\oplus1}, ct_{i,f[i]\oplus1}) \leftarrow \mathcal{E}nc(\mathsf{pk}_{i,f[i]\oplus1}, m_0[i])$$
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 $(\mathsf{vk}_{j+1,f[j+1]\oplus1}, ct_{j+1,f[j+1]\oplus1}) \leftarrow \mathcal{E}nc(\mathsf{pk}_{j+1,f[j+1]\oplus1}, m_b[j+1])$ and sends $ct_{j+1,f[j+1]\oplus1}$ to $\mathcal{B}$.
9. $\mathcal{B}$ sends $(\{\mathsf{sk}_{i,f[i]}\}_{i\in[s]}, \{ct_{i,f[i]\oplus1}\}_{i\in[s]})$ 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 $\mathsf{sk}_{j+1,f[j+1]\oplus1}$ or $\bot$ from the challenger. If $\mathcal{B}$ receives $\bot$ from the challenger, it outputs $\bot$ and aborts.
12. $\mathcal{B}$ sends $\{\mathsf{sk}_{i,f[i]\oplus1}\}_{i\in[s]}$ 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 $\bot$.

Since $\mathsf{pk}_{j+1,\beta}$ and $\mathsf{pk}_{j+1,\beta\oplus1}$ are identically distributed, it holds that $\Pr[f[j+1] = \beta] = \Pr[f[j+1] = \beta\oplus1] = \frac{1}{2}$. If $b = 0$ and $f[j+1] = \beta\oplus1$, $\mathcal{B}$ simulates Hyb_j. Therefore, we have

$$\begin{aligned}\Pr[1 \leftarrow \mathcal{B} \mid b = 0] &= \Pr[1 \leftarrow \mathcal{B} \wedge f[j+1] = \beta\oplus1 \mid b = 0] \\ &= \Pr[1 \leftarrow \mathcal{B} \mid b = 0, f[j+1] = \beta\oplus1] \cdot \Pr[f[j+1] = \beta\oplus1] \\ &= \frac{1}{2}\Pr\left[\mathsf{Hyb}_j = 1\right].\end{aligned}$$

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

E Garbling Scheme with Certified Everlasting Deletion

In Appendix E.1, we define a garbling scheme with certified everlasting deletion. In Appendix E.2, we construct a certified everlasting secure garbling scheme from a certified everlasting secure SKE scheme.

E.1 Definition

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

Definition E.1 (Garbling Scheme with Certified Everlasting Deletion (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 garbling scheme with certified everlasting deletion is a tuple of algorithms $\Sigma=(\mathsf{Setup},\mathcal{G}arble,\mathcal{E}val,\mathcal{D}el,\mathsf{Vrfy})$ 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 $\mathcal{D}:=\mathcal{Q}^{\otimes s(\lambda)}$.*

$\mathsf{Setup}(1^\lambda)\rightarrow\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}}$**:** *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 $\alpha\in\{0,1\}$.*

$\mathcal{G}arble(1^\lambda,C,\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}})\rightarrow(\widetilde{C},\mathsf{vk})$**:** *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 $\mathsf{vk}\in\mathcal{VK}$.*

$\mathcal{E}val(\widetilde{C},\{L_{i,x_i}\}_{i\in[n]})\rightarrow y$**:** *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.*

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

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

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

Definition E.2 (Correctness for Garbling Scheme with Certified Everlasting Deletion). *There are two types of correctness, namely, evaluation correctness and verification correctness.*

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

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

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

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

Minimum requirements for correctness are evaluation correctness and verification correctness. However, we also require verification correctness with QOTP in this work because we need it for the construction of FE in Section 4.3.

Definition E.3 (Verification Correctness with QOTP). *There exists a negligible function* negl *and a PPT algorithm* $\mathsf{Recover}$ *such that for any* $\lambda \in \mathbb{N}$*,*

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

As security, we consider two definitions, Definition E.4 and Definition E.5 given below. The former is just the standard 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 E.4 (Selective Security for Garbling Scheme with Certified Everlasting Deletion). *Let* $\Sigma = (\mathsf{Setup}, \mathcal{Garble}, \mathcal{Eval}, \mathcal{Del}, \mathsf{Vrfy})$ *be a garbling scheme with certified everlasting deletion. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{sel\text{-}gbl}}_{\Sigma,\mathcal{A}}(1^\lambda, b)$ *against a QPT adversary* $\mathcal{A}$*. Let* $\mathcal{Sim}$ *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* $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Setup}(1^\lambda)$.
3. *If $b = 0$, the challenger computes* $(\widetilde{C}, \mathsf{vk}) \leftarrow \mathcal{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}})$*, and returns* $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ *to* $\mathcal{A}$*. If $b = 1$, the challenger computes* $\widetilde{C} \leftarrow \mathcal{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i\in[n]})$*, 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 $\mathcal{Sim}$ such that for any QPT adversary $\mathcal{A}$ it holds that

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

Definition E.5 (Selective Certified Everlasting Security for Garbling Scheme). *Let* $\Sigma = (\mathsf{Setup}, \mathcal{Garble}, \mathcal{Eval}, \mathcal{Del}, \mathsf{Vrfy})$ *be a garbling scheme with certified everlasting deletion. We consider the following security experiment* $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}sel\text{-}gbl}}_{\mathcal{A},\Sigma}(1^\lambda, b)$ *against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let $\mathcal{Sim}$ 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* $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Setup}(1^\lambda)$.
3. *If $b = 0$, the challenger computes* $(\widetilde{C}, \mathsf{vk}) \leftarrow \mathcal{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}})$*, and returns* $(\widetilde{C}, \{L_{i,x_i}\}_{i\in[n]})$ *to $\mathcal{A}_1$. If $b = 1$, the challenger computes* $(\widetilde{C}, \mathsf{vk}) \leftarrow \mathcal{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i\in[n]})$*, 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 selective certified everlasting secure if there exists a QPT simulator $\mathcal{Sim}$ such that for any QPT $\mathcal{A}_1$ and any unbounded $\mathcal{A}_2$ it holds that

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

E.2 Construction

In this section, we construct a certified everlasting secure garbling scheme from a certified everlasting secure SKE scheme (Definition C.1). Our construction is similar to Yao's construction of a standard garbling scheme [Yao86], but there are two important differences. First, we use a certified everlasting secure SKE scheme instead of a standard 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, $\mathsf{gate}_i := (g, w_a, w_b, w_c)$, 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 secure everlasting garbling scheme. We construct a certified everlasting secure garbling scheme $\Sigma_{\mathsf{cegc}} = (\mathsf{Setup}, \mathcal{G}arble, \mathcal{E}val, \mathcal{D}el, \mathsf{Vrfy})$ from a certified everlasting secure SKE scheme $\Sigma_{\mathsf{cesk}} = \mathsf{SKE}.(\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ (Definition C.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.

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

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

$\mathcal{G}arble(1^\lambda, C, \{L_{i,\sigma}\}_{i\in[n],\sigma\in\{0,1\}})$:

- 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

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

  where $\mathsf{gate}_i = (g, w_a, w_b, w_c)$ and $\mathcal{G}ate\mathcal{G}rbl$ is described in Fig 7.
- For each $i \in [m]$, set $\widetilde{d}_i := [(\mathsf{ske.sk}^0_{\mathsf{out}_i}, 0), (\mathsf{ske.sk}^1_{\mathsf{out}_i}, 1)]$.
- Output $\widetilde{C} := (\{\widetilde{g}_i\}_{i\in[q]}, \{\widetilde{d}_i\}_{i\in[m]})$ and $\mathsf{vk} := \{\mathsf{vk}_i\}_{i\in[q]}$.

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

- 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 $\mathsf{ske.sk}'_c \leftarrow \mathcal{G}ate\mathcal{E}val(\widetilde{g}_i, \mathsf{ske.sk}'_a, \mathsf{ske.sk}'_b)$ in the ascending order of the level, where $\mathcal{G}ate\mathcal{E}val$ is described in Fig 8. 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 $\mathsf{cert}_i \leftarrow \mathcal{G}ate\mathcal{D}el(\widetilde{g}_i)$, where $\mathcal{G}ate\mathcal{D}el$ is described in Fig 9.
- 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 10.
- 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 $\mathcal{GateGrbl}$

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. 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 $p_c^{\sigma_a,\sigma_b} \leftarrow \mathcal{K}$.
4. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b})$ and $(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^{g(\sigma_a,\sigma_b)})$.
5. 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 the 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 the permutated order of γ_i.

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

Figure 7: The description of $\mathcal{GateGrbl}$

Gate Evaluating Circuit $\mathcal{GateEval}$

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

Figure 8: The description of $\mathcal{GateEval}$

Gate Deletion Circuit $\mathcal{GateDel}$

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 $\mathsf{ske.cert}_a^{\sigma_a,\sigma_b} \leftarrow \mathsf{SKE}.\mathcal{Del}(\mathsf{ske}.ct_a^{\sigma_a,\sigma_b})$.
3. For each $\sigma_a, \sigma_b \in \{0,1\}$, compute $\mathsf{ske.cert}_b^{\sigma_a,\sigma_b} \leftarrow \mathsf{SKE}.\mathcal{Del}(\mathsf{ske}.ct_b^{\sigma_a,\sigma_b})$.
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 9: The description of $\mathcal{GateDel}$

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

Gate Verification Circuit 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 10: The description of GateVrfy

Security: The following two theorems hold.

Theorem E.6. *If Σ_{cesk} satisfies the IND-CPA security (Definition C.6), Σ_{cegc} satisfies the selective security (Definition E.4).*

Its proof is similar to that of Theorem E.7, and therefore we omit it.

Theorem E.7. *If Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition C.7), Σ_{cegc} satisfies the selective certified everlasting security (Definition E.5).*

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 $\widehat{\mathsf{gate}}_i = (\bot, w_a, w_b, w_c)$.

Proof of Theorem E.7. First, let us define a simulator $\mathcal{S}im$ as follows.

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

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 $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathcal{S}im.\mathcal{G}ate\mathcal{G}rbl(\widehat{\mathsf{gate}}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathcal{S}im.\mathcal{G}ate\mathcal{G}rbl$ is described in Fig 11 and $\widehat{\mathsf{gate}}_i = (\bot, w_a, w_b, w_c)$.
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) $\mathsf{InputDep}.\mathcal{S}im_j$ as follows.

The simulator $\mathsf{InputDep}.\mathcal{S}im_j(1^\lambda, C, x, \{L_{i,x_i}\}_{i\in[n]})$**:**

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 $\mathsf{ske.sk}_i^\sigma \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$.
4. For $i \in [j]$, compute $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{InputDep}.\mathcal{G}ate\mathcal{G}rbl(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{InputDep}\mathcal{G}ate\mathcal{G}rbl$ is described in Fig. 12 and $\mathsf{gate}_i = (g, w_a, w_b, w_c)$
5. For each $i \in \{j+1, j+2, \cdots, q\}$, compute $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathcal{G}ate\mathcal{G}rbl(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathcal{G}ate\mathcal{G}rbl$ is described in Fig 7 and $\mathsf{gate}_i = (g, w_a, w_b, w_c)$.

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

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 $(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b})$ and $(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^0)$.
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 11: The description of $\mathcal{Sim}.\mathcal{GateGrbl}$

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

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 $(\mathsf{ske.vk}_a^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_a^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a,\sigma_b})$ and $(\mathsf{ske.vk}_b^{\sigma_a,\sigma_b}, \mathsf{ske}.ct_b^{\sigma_a,\sigma_b}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a,\sigma_b} \oplus \mathsf{ske.sk}_c^{v(c)})$. 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 12: The description of $\mathsf{InputDep}.\mathcal{GateGrbl}$

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 $\mathsf{vk} := \{\mathsf{vk}_i\}_{i\in[q]}$.

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

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

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 $\{L_{i,\alpha}\}_{i\in[n],\alpha\in\{0,1\}} \leftarrow \mathsf{Smp}(1^\lambda)$.
3. The challenger computes $(\widetilde{C}, \mathsf{vk}) \leftarrow \mathsf{GC.InputDep}.\mathcal{Sim}_j(1^\lambda, C, x, \{L_{i,x_i}\}_{i\in[n]})$, 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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}sel\text{-}gbl}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}(1^\lambda, 0)$ by definition. Therefore, Theorem E.7 easily follows from the following Propositions E.8 and E.9 (whose proofs are given later). $\square$

Proposition E.8. *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 E.9. $\left|\Pr\left[\mathsf{Hyb}_q = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{cert\text{-}ever\text{-}sel\text{-}gbl}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}}(1^\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda)$.

Proof of Proposition E.8. For the proof, we use Lemma E.10 whose statement and proof are given later. 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 E.10, 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 E.10. 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 $\mathsf{Exp}^{\mathsf{parallel\text{-}cert}}_{\Sigma_{\mathsf{cesk}},\mathcal{B}} ever(\lambda, b)$ generates $\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$ and $\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1} \leftarrow \mathsf{SKE.KeyGen}(1^\lambda)$[24].

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 $\{L_{i,x_i}\}_{i\in[n]} := \{\mathsf{ske.sk}_i^{x_i}\}_{i\in[n]}$.

4. For each $i \in [j-1]$, $\mathcal{B}$ computes $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{InputDep}.\mathcal{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$, where $\mathsf{InputDep}.\mathcal{GateGrbl}$ is described in Fig 12 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 $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathsf{InputDep}.\mathcal{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^\sigma, \mathsf{ske.sk}_b^\sigma, \mathsf{ske.sk}_c^\sigma\}_{\sigma\in\{0,1\}})$.

5. $\mathcal{B}$ samples $p_\gamma^{v(\alpha),v(\beta)} \leftarrow \mathcal{K}$. $\mathcal{B}$ computes
$$(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \mathsf{ske}.ct_\alpha^{v(\alpha),v(\beta)}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)}, p_\gamma^{v(\alpha),v(\beta)}),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)}, \mathsf{ske}.ct_\beta^{v(\alpha),v(\beta)}) \leftarrow \mathsf{SKE}.\mathcal{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)}, p_\gamma^{v(\alpha),v(\beta)} \oplus \mathsf{ske.sk}_\gamma^{v(\gamma)}).$$

6. $\mathcal{B}$ sets
$$(x_0, y_0, z_0) := (\mathsf{ske.sk}_\gamma^{g(v(\alpha),v(\beta)\oplus 1)}, \mathsf{ske.sk}_\gamma^{g(v(\alpha)\oplus 1,v(\beta))}, \mathsf{ske.sk}_\gamma^{g(v(\alpha)\oplus 1,v(\beta)\oplus 1)}),$$
$$(x_1, y_1, z_1) := (\mathsf{ske.sk}_\gamma^{v(\gamma)}, \mathsf{ske.sk}_\gamma^{v(\gamma)}, \mathsf{ske.sk}_\gamma^{v(\gamma)}),$$
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
$$(\mathsf{ske.vk}_\alpha^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske}.ct_\alpha^{v(\alpha),v(\beta)\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)}, x),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha),v(\beta)\oplus 1}, \mathsf{ske}.ct_\beta^{v(\alpha),v(\beta)\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}, x \oplus x_b),$$
$$(\mathsf{ske.vk}_\alpha^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske}.ct_\alpha^{v(\alpha)\oplus 1,v(\beta)}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}, y),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha)\oplus 1,v(\beta)}, \mathsf{ske}.ct_\beta^{v(\alpha)\oplus 1,v(\beta)}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)}, y \oplus y_b),$$
$$(\mathsf{ske.vk}_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske}.ct_\alpha^{v(\alpha)\oplus 1,v(\beta)\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\alpha^{v(\alpha)\oplus 1}, z),$$
$$(\mathsf{ske.vk}_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1}, \mathsf{ske}.ct_\beta^{v(\alpha)\oplus 1,v(\beta)\oplus 1}) \leftarrow \mathcal{Enc}(\mathsf{ske.sk}_\beta^{v(\beta)\oplus 1}, z \oplus z_b),$$

[24] 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 $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathcal{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^{\sigma}, \mathsf{ske.sk}_b^{\sigma}, \mathsf{ske.sk}_c^{\sigma}\}_{\sigma\in\{0,1\}})$, 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 $(\mathsf{vk}_i, \widetilde{g}_i) \leftarrow \mathcal{GateGrbl}(\mathsf{gate}_i, \{\mathsf{ske.sk}_a^{\sigma}, \mathsf{ske.sk}_b^{\sigma}, \mathsf{ske.sk}_c^{\sigma}\}_{\sigma\in\{0,1\}})$.
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 $\bot$ 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 $\mathsf{GateVrfy}$ is described in Fig10. 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 $\bot$ to $\mathcal{A}_2$, and aborts.
13. $\mathcal{B}$ outputs the output of $\mathcal{A}_2$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr\left[\mathsf{Hyb}_{j-1} = 1\right]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr\left[\mathsf{Hyb}_j = 1\right]$. 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 E.10, it contradicts the certified everlasting IND-CPA security of Σ_{cesk} , which completes the proof. ☐

Proof of Proposition E.9. To show Proposition E.9, 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

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

and $\widetilde{d}_i$ is

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

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

$$(\mathsf{ske.vk}^{\sigma_a,\sigma_b}_a, \mathsf{ske}.ct^{\sigma_a,\sigma_b}_a) \leftarrow \mathsf{SKE}.\mathcal{E}nc(\mathsf{ske.sk}^{\sigma_a}_a, p^{\sigma_a,\sigma_b}_c),$$
$$(\mathsf{ske.vk}^{\sigma_a,\sigma_b}_b, \mathsf{ske}.ct^{\sigma_a,\sigma_b}_b) \leftarrow \mathsf{SKE}.\mathcal{E}nc(\mathsf{ske.sk}^{\sigma_b}_b, p^{\sigma_a,\sigma_b}_c \oplus \mathsf{ske.sk}^0_c),$$

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 $\mathsf{Exp}^{\mathsf{cert\text{-}ever}}_{\Sigma_{\mathsf{cegc}},\mathcal{A}} select(1^\lambda, 1)$. 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 $\mathsf{ske.sk}^{v(c)}_c \to v(c)$ or encrypting $\mathsf{ske.sk}^0_c$ and giving the programming mapping $\mathsf{ske.sk}^0_c \to C(x)_i$, which completes the proof. ☐

We use the following lemma for the proof of Proposition E.8. 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 E.10. *Let* $\Sigma := (\mathsf{KeyGen}, \mathcal{E}nc, \mathcal{D}ec, \mathcal{D}el, \mathsf{Vrfy})$ *be a certified everlasting secure 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* $(\mathsf{vk}^0_m, ct^0_m) \leftarrow \mathcal{E}nc(\mathsf{sk}'^0, m^*)$ *and* $(\mathsf{vk}^1_m, ct^1_m) \leftarrow \mathcal{E}nc(\mathsf{sk}, m \oplus m^*)$*, and sends* $\{\mathsf{vk}^\sigma_m, ct^\sigma_m\}_{\sigma\in\{0,1\}}$ *to* $\mathcal{A}_1$*.*
   - *If* $\beta = 1$*, the challenger generates* $m^* \leftarrow \mathcal{M}$*, computes* $(\mathsf{vk}^1_m, ct^1_m) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, m \oplus m^*)$ *and* $(\mathsf{vk}^0_m, ct^0_m) \leftarrow \mathcal{E}nc(\mathsf{sk}, m^*)$*, and sends* $\{\mathsf{vk}^\sigma_m, ct^\sigma_m\}_{\sigma\in\{0,1\}}$ *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*

$$(\mathsf{vk}^0_x, ct^0_x) \leftarrow \mathcal{E}nc(\mathsf{sk}^0, x), \quad (\mathsf{vk}^1_x, ct^1_x) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, x \oplus x_b)$$
$$(\mathsf{vk}^0_y, ct^0_y) \leftarrow \mathcal{E}nc(\mathsf{sk}'^0, y), \quad (\mathsf{vk}^1_y, ct^1_y) \leftarrow \mathcal{E}nc(\mathsf{sk}^1, y \oplus y_b)$$
$$(\mathsf{vk}^0_z, ct^0_z) \leftarrow \mathcal{E}nc(\mathsf{sk}'^0, z), \quad (\mathsf{vk}^1_z, ct^1_z) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, z \oplus z_b)$$

   and sends $\{ct^\sigma_x, ct^\sigma_y, ct^\sigma_z\}_{\sigma\in\{0,1\}}$ *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 $(\mathsf{vk}_m^0, ct_m^0) \leftarrow \mathcal{E}nc(\mathsf{sk}'^0, m^*)$ and $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}, m \oplus m^*)$, and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.*
- *If $\beta = 1$, the challenger generates $m^* \leftarrow \mathcal{M}$, computes $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, m \oplus m^*)$ and $(\mathsf{vk}_m^0, ct_m^0) \leftarrow \mathcal{E}nc(\mathsf{sk}, m^*)$, and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.*

$\mathcal{A}_1$ can repeat this process polynomially many times.

6. *$\mathcal{A}_1$ sends $\{\mathsf{cert}_x^\sigma, \mathsf{cert}_y^\sigma, \mathsf{cert}_z^\sigma\}_{\sigma \in \{0,1\}}$ 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,

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

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

Proof of Lemma E.10. We define the following hybrid experiment.

Hyb_1**:** This is identical to $\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0)$ 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 E.10 easily follows from the following Propositions E.11 to E.13 (whose proof is given later.). □

Proposition E.11. *If Σ is certified everlasting IND-CPA secure, it holds that*

$$\left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1\right] - \Pr[\mathsf{Hyb}_1 = 1]\right| \leq \mathsf{negl}(\lambda).$$

Proposition E.12. *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 E.13. *If Σ is certified everlasting IND-CPA secure, it holds that*

$$\left|\Pr[\mathsf{Hyb}_2 = 1] - \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 1) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Proof of Proposition E.11. We assume that $\left|\Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1\right] - \Pr[\mathsf{Hyb}_1(1) = 1]\right|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment of $\mathsf{Exp}_{\Sigma,\mathcal{B}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$. This contradicts the certified everlasting IND-CPA security of Σ. Let us describe how $\mathcal{B}$ works.

1. The challenger of $\mathsf{Exp}_{\Sigma,\mathcal{B}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$ 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 $(\mathsf{vk}_m^0, ct_m^0)$ from the challenger, computes $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}, m \oplus m^*)$, and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.

- If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, m \oplus m^*)$ and $(\mathsf{vk}_m^0, ct_m^0) \leftarrow \mathcal{E}nc(\mathsf{sk}, m^*)$ and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma\in\{0,1\}}$ 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

$$(\mathsf{vk}_x^0, ct_x^0) \leftarrow \mathcal{E}nc(\mathsf{sk}^0, x), (\mathsf{vk}_x^1, ct_x^1) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, x \oplus x_0),$$
$$(\mathsf{vk}_y^1, ct_y^1) \leftarrow \mathcal{E}nc(\mathsf{sk}^1, y \oplus y_0),$$
$$(\mathsf{vk}_z^1, ct_z^1) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, z \oplus z_0).$$

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 $(\mathsf{vk}_z^0, ct_z^0) \leftarrow \mathcal{E}nc(\mathsf{sk}'^0, m_b)$, and sends ct_z^0 to $\mathcal{B}$.

7. $\mathcal{B}$ sends an encryption query y to the challenger, and receives $(\mathsf{vk}_y^0, ct_y^0)$.

8. $\mathcal{B}$ sends $\{ct_x^\sigma, ct_y^\sigma, ct_z^\sigma\}_{\sigma\in\{0,1\}}$ 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 $(\mathsf{vk}_m^0, ct_m^0)$ from the challenger, computes $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}, m \oplus m^*)$, and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma\in\{0,1\}}$ to $\mathcal{A}_1$.
   - If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(\mathsf{vk}_m^1, ct_m^1) \leftarrow \mathcal{E}nc(\mathsf{sk}'^1, m \oplus m^*)$ and $(\mathsf{vk}_m^0, ct_m^0) \leftarrow \mathcal{E}nc(\mathsf{sk}, m^*)$ and sends $\{\mathsf{vk}_m^\sigma, ct_m^\sigma\}_{\sigma\in\{0,1\}}$ 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr\left[\mathsf{Exp}_{\Sigma,\mathcal{A}}^{\mathsf{parallel\text{-}cert\text{-}ever}}(\lambda, 0) = 1\right]$. 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\mathsf{Hyb}_1 = 1]$. 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

$$|\Pr[1 \leftarrow \mathcal{B} \mid b = 0] - \Pr[1 \leftarrow \mathcal{B} \mid b = 1]|$$

is non-negligible, which contradicts the certified everlasting IND-CPA security of Σ_{cesk}. ☐

Proof of Proposition E.12. The proof is very similar to that of Proposition E.11. Therefore we skip the proof. ☐

Proof of Proposition E.13. The proof is very similar to that of Proposition E.11. Therefore, we skip the proof. ☐