

Thesis

**Quantum Cryptographic Primitives
with Certified Everlasting Security**

Taiga Hiroka

January 22, 2025

Abstract

In cryptography, various cryptographic primitives rely on computational assumptions that certain problems are hard to solve with a current computing device. A drawback is that such assumptions could be broken in the future when revolutionary novel algorithms are found, or computing devices are drastically improved. One solution to the problem is to achieve information theoretical security. In information theoretical security, security is ensured regardless of the adversary's computing resource. However, many cryptographic primitives are known as impossible to achieve information-theoretical security even in the realm of quantum.

In this thesis, we study a novel compromise called "certified everlasting security". This security roughly ensures the following. In certified everlasting security, computational security is ensured as a default. Furthermore, the receiver can issue a deletion certificate proving that the encrypted data has been deleted. Once the validity of the certificate is verified, even if the receiver's computing power becomes unbounded, it cannot obtain the information of the original message. This security notion is weaker than information-theoretical security because the malicious receiver could refuse to issue a valid certificate. However, it is a still meaningful security concept because we can prevent a malicious receiver from refusing to issue a certificate by penalties. Interestingly, certified everlasting security is intrinsic to quantum. More precisely, if the encrypted data is classical strings, it would imply information-theoretical security, and thus certified everlasting security is impossible to achieve for several primitives. This is because if the encrypted data were a classical bit string, a malicious receiver could freely copy it. Therefore, the encrypted data must be secure against an unbounded malicious receiver at the point when the receiver obtains the encrypted data. The same argument does not apply to the quantum case because, due to the quantum no-cloning theorem, even a malicious receiver cannot copy quantum-encrypted data.

We construct several cryptographic primitives with certified everlasting security. In Chapter 3, we introduce and construct a certified everlasting version of zero-knowledge proof for **QMA**. In the primitive, it is a computational zero-knowledge proof for **QMA**, but the verifier issues a classical certificate showing that the verifier has deleted its quantum information. If the certificate is valid, even an unbounded malicious verifier can no longer learn anything beyond the validity of the statement. For the construction, we introduce a novel quantum cryptographic primitive called "commitment with statistical binding and certified everlasting hiding", which is independent of interest. In this primitive, the hiding property becomes statistical after the receiver issues a valid certificate indicating that the committed information has been deleted.

In Chapter 4, we introduce and construct a certified everlasting version of functional encryption(FE), compute-and-compare obfuscation, and predicate encryption(PE). These cryptographic protocols, originally computationally secure FE, compute-compare obfuscation, and PE, respectively. In addition, they allow for the issuance of a certificate that proves the deletion of ciphertexts. Once the deletion is confirmed, the security of these protocols becomes information-theoretical security. For the construction, we introduce several certified everlasting versions of cryptographic primitives such as secret-key encryption (SKE), public-key encryption, and garbling scheme.

Contents

List of Publications	6
1 Introduction	8
1.1 Background	8
1.2 Cryptographic Primitives	11
1.3 Summary of Contributions	15
1.4 Concurrent and independent works	17
1.5 Organization	18
2 Preliminaries	20
2.1 Notations	20
2.2 Cryptographic Tools	20
2.3 Quantum Computations	30
3 Certified Everlasting Zero-Knowledge Proof for QMA	32
3.1 Introduction	32
3.1.1 Background	32
3.1.2 Our Results	33
3.1.3 Technical Overview	35
3.2 Commitment with Certified Everlasting Hiding and Classical-Extractor-Based Binding . . .	36
3.2.1 Definition	37
3.2.2 Construction	40
3.3 Certified Everlasting Zero-Knowledge Proof for QMA	45
3.3.1 Definition	45
3.3.2 Construction of Three Round Protocol	47
3.3.3 Sequential Repetition for Certified Everlasting Zero-Knowledge Proof for QMA . . .	54
4 Certified Everlasting Secure Functional Encryption	57
4.1 Introduction	57
4.1.1 Background	57
4.1.2 Our Results	59
4.2 Technical Overview	60
4.2.1 Technical Overview: Collusion-Resistant FE	60
4.2.2 Technical Overview: Bounded Collusion-Resistant FE	63
4.2.3 Technical Overview: Compute-and-Compare Obfuscation	66

4.3	Collusion-Resistant Functional Encryption with Certified Everlasting Deletion	70
4.3.1	Definitions	70
4.3.2	Tools	71
4.3.3	Collusion-Resistant Construction	74
4.4	Bounded Collusion-Resistant Functional Encryption with Certified Everlasting Deletion . . .	83
4.4.1	Definitions	83
4.4.2	1-Bounded Construction with Non-Adaptive Security	86
4.4.3	1-Bounded Construction with Adaptive Security	90
4.4.4	q -Bounded Construction with Adaptive Security for NC^1 circuits	96
4.4.5	Discussion on q -Bounded Consturction for All Circuits	103
4.5	Compute-and-Compare Obfuscation with Certified Everlasting Deletion	104
4.5.1	Definition	104
4.5.2	Construction	105
4.6	Predicate Encryption with Certified Everlasting Deletion	111
4.6.1	Definition	111
4.6.2	Construction	113
5	Conclusion	120
A	Appendix	132
A.1	Omitted Proofs for Collusion-Resistant FE	132
A.2	Adaptively Secure Public-Slot PKFE	134
A.2.1	Building Blocks	135
A.2.2	Variants of Security Definitions	139
A.2.3	Adaptively Single-Key Single-Ciphertext Public-Slot SKFE Scheme	140
A.2.4	Adaptively Secure Public-Slot PKFE Scheme	144
A.3	Secret and Public Key Encryption with Certified Everlasting Deletion	154
A.3.1	Definition	155
A.3.2	SKE Scheme with QROM	159
A.3.3	SKE Scheme without QROM	160
A.3.4	PKE Scheme with QROM	162
A.3.5	PKE Scheme without QROM	163
A.4	Receiver Non-Committing Encryption with Certified Everlasting Deletion	165
A.4.1	Definition	165
A.4.2	Construction	167
A.5	Garbling Scheme with Certified Everlasting Deletion	173
A.5.1	Definition	173
A.5.2	Construction	175
	Acknowledgements	186

List of Publications

Papers related to the thesis

1. Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa, “Certified Everlasting Zero-Knowledge Proof for QMA ,” CRYPTO 2022, Part I, volume 13507 of LNCS, pages 239–268
2. Taiga Hiroka, Fuyuki Kitagawa, Tomoyuki Morimae, Ryo Nishimaki, Tapas Pal, and Takashi Yamakawa, “Certified Everlasting Secure Collusion-Resistant Functional Encryption, and More,” EUROCRYPT 2024, Part I, volume 14653 of LNCS, pages 434-456

Papers not included in the thesis

1. Taiga Hiroka, Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa, “Robust Combiners and Universal Constructions for Quantum Cryptography,” TCC 2024, Part II, volume 15365 of LNCS, pages 126-158
2. Taiga Hiroka and Tomoyuki Morimae, “Quantum Cryptography and Meta-Complexity,” arXiv:2410.01369.
3. Taiga Hiroka and Min-Hsiu Hsieh, “Computational Complexity of Learning Efficiently Generatable Pure States”, arXiv:2410.04373.

Chapter 1

Introduction

1.1 Background

General background of modern cryptography. Cryptography is the study of secure communication in the presence of adversarial behavior. One of the most fundamental cryptographic protocols is the one-time pad [Sha49]. This protocol enables secure communication between parties who share a secret key in advance (See Figure 1.1). When a sender wishes to send a message m to a receiver, he encrypts m using the pre-shared secret key sk by computing the ciphertext as $ct(m) = m + sk$, and then transmit $ct(m)$. Upon receiving the ciphertext $ct(m)$, the receiver can decrypt it by using the secret key sk to recover m . Even if adversaries intercept the communication and obtain the ciphertext $ct(m)$, they cannot deduce any information about m , regardless of their computing power because they do not know the secret key sk . However, a significant drawback of the one-time pad is that it requires the sender and receiver to share a secret key sk in advance, which is highly impractical. Therefore, it was an important problem of how to securely exchange messages without a pre-shared secret key.

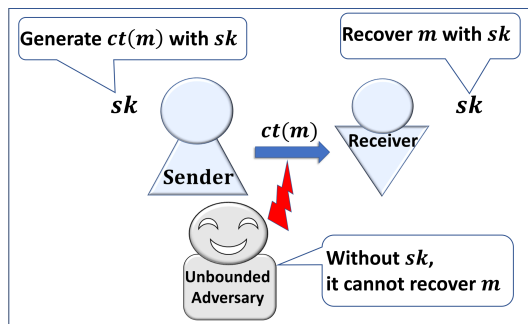


Figure 1.1: One-time pad

A revolutionary shift occurred with the introduction of public-key encryption (PKE) by Diffie and Hellman in 1976 [DH76]. In a PKE scheme, a receiver issues two different keys: a public key pk for encryption and a secret key sk for decryption (See Figure 1.2). The public key pk is shared openly, allowing anyone to encrypt their message m and generate the ciphertext $ct(m)$ of a message m . In contrast, the secret key sk is kept private. Using the secret key sk , the receiver can decrypt the ciphertext $ct(m)$ and recover the original message m . A fundamental feature of PKE is that adversaries cannot decrypt the ciphertext $ct(m)$ even if they know the public key pk . This allows anyone to encrypt a message with the public key pk , while only the

receiver possessing the corresponding secret key sk can decrypt it and recover the original message. As a result, PKE enables secure communication without prior key exchange, while keeping secrets secure from adversaries.

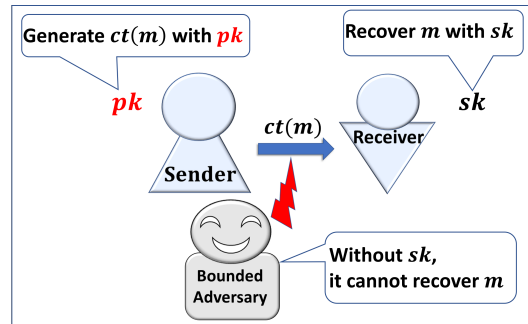


Figure 1.2: Public-Key Encryption

The first construction of PKE was introduced by Rivest, Shamir, and Adleman in 1978 [RSA78]. They constructed PKE based on the assumption that certain mathematical problems cannot be solved efficiently. As a consequence, the security of the scheme depends on the hardness of these underlying problems, which are considered infeasible to solve within a realistic amount of time with current computing devices. This type of security, which holds only when the adversaries' computing power is bounded, is called computational security. Computational security, however, is conditional: it holds only as long as the underlying problem remains hard. For example, if a groundbreaking algorithm were discovered that could efficiently solve the problem, or if computing devices were drastically improved, such as with quantum computers, the hardness assumptions could be broken. This would allow an adversary to break the encryption scheme. Therefore, while computational security can offer practical protection, it is inherently vulnerable to future advances in algorithms and computing devices.

While PKE is a cryptographic primitive developed for secure message transmission, modern cryptographic research has expanded the notion of PKE and introduced more practical and flexible cryptographic primitives with enhanced functionality such as multi-party computation [GMW87], attribute-based encryption [SW05], and functional encryption [BSW11]. Unfortunately, the security of almost all previous constructions of these primitives is also based on computational assumptions. This means that they could be broken by future technological advancements.

To remove the issue, we need to give constructions with information-theoretical security. In information-theoretical security, even if adversaries possess unlimited computational resources, the security of the system remains unaffected. As a consequence, it cannot be broken even if groundbreaking algorithms are discovered or computing devices are drastically improved. This robustness stems from the fact that information-theoretical security does not rely on the computational hardness of specific mathematical problems. Instead, it ensures that encrypted data inherently lacks sufficient information for any adversary to recover the original message. However, it is known that achieving information-theoretical security for PKE and many other cryptographic tasks is impossible [Sha49, DH76]. For these backgrounds, research on quantum cryptography, which leverages the unique properties of quantum mechanics, has been actively pursued.

Quantum Cryptography. Quantum cryptography is the study of how to apply quantum mechanics for secure communication in the presence of adversarial behavior. In quantum cryptography, we consider the scenario where each party possesses a quantum computer and can transmit quantum states to each other.

There are certain tasks that quantum cryptography provides information-theoretical security while classical cryptography cannot.

A key example is quantum key distribution (QKD) introduced by Bennett and Brassard [BB84] based on Wiesner’s earlier work [Wie83]. In QKD, a sender and a receiver can share a secret key by quantum communication (See Figure 1.3). An important aspect of QKD is that it provides information-theoretical security. Any attempt by even computationally unbounded adversaries to extract the shared secret key will disturb quantum states. In QKD, the sender and receiver can detect the disturbance, and hence they can notice if the secret key has been compromised by computationally unbounded adversaries. Once the secret key is securely shared, they can transmit messages with information-theoretical security by using the shared random secret key. In this way, without a pre-shared secret key, QKD enables us to transmit messages with information-theoretical security. Because, in classical cryptography, it is impossible to achieve the task with information-theoretical security, QKD thus stands as a unique example where quantum cryptography achieves a level of security unachievable by classical cryptography.

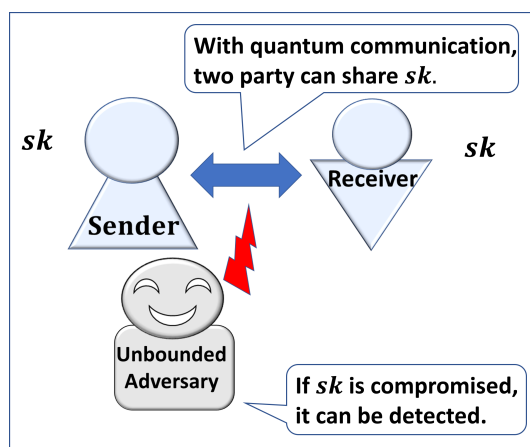


Figure 1.3: Quantum Key Distribution

Given the example, it is reasonable to hope that quantum cryptography could provide information-theoretical security for a variety of cryptographic protocols where classical cryptography fails. In classical cryptography, numerous cryptographic primitives—such as digital signatures [Rom90], commitment schemes [Nao90], multi-party computations [GMW87], attribute-based encryption [SW05], and functional encryption [BSW11]—are known as impossible to achieve information-theoretical security. If quantum cryptography could make this possible, it would be a remarkable breakthrough. However, it is unfortunate that, for many cryptographic protocols, information-theoretical security is shown as impossible even with quantum cryptography [May97, LC97, MW18, Aar18, LC19]. Consequently, it is now considered that information-theoretical security is not achievable for almost all cryptographic protocols, even in the quantum realm.

Although quantum cryptography cannot provide information theoretical security for almost all cryptographic primitives, it remains a rich field of study. This is because quantum mechanics enables us to construct several useful cryptographic protocols (e.g., quantum money [Wie83], unclonable encryption [Got04, BL20], and copy-protection [Aar09]) that are fundamentally impossible with classical cryptography. Among them, Broadbent and Islam [BI20] (based on [Unr14]) recently suggested a novel security notion, which we call certified everlasting security (See Figure 1.4). Certified everlasting security provides a practical compromise to information-theoretical security. In protocols with certified everlasting security, a sender encrypts a message

by generating quantum-encrypted data and sends it to a receiver. The receiver can then delete, issuing a deletion certificate to confirm this deletion. Once this certificate is verified, the original message becomes permanently unrecoverable even if the receiver later gains unbounded computational resources. Although this security notion is weaker than information-theoretical security (since a malicious receiver could refuse to issue a valid certificate), it remains highly useful. For example, by implementing penalties, it is possible to discourage a malicious receiver from withholding a valid certificate.

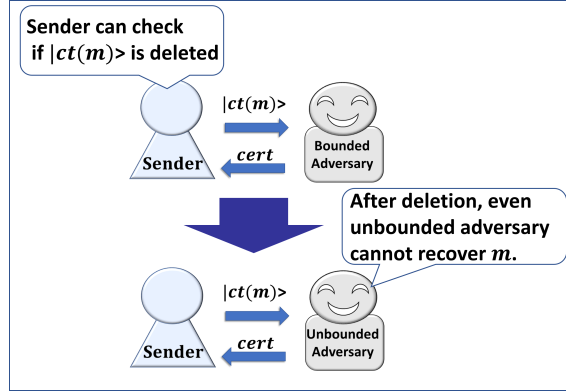


Figure 1.4: Certified Everlasting Security

Despite its practical advantages and theoretical appeal, Broadbent and Islam [BI20] left the formal definitions and constructions as open problems. In fact, achieving certified everlasting security is non-trivial, and at least without the help of quantum mechanics, it is impossible to achieve. If the encrypted data is a classical bit string, the adversary can freely clone it and hence the scheme must be secure against unbounded adversaries from the point it receives the data. This means that without the help of quantum mechanics, certified everlasting security is equivalent to information theoretical security, and hence it is impossible to achieve for many cryptographic primitives. On the other hand, this argument cannot be applied to the quantum case, because even a malicious adversary cannot clone quantum states due to the quantum no-cloning theorem. Therefore, there exists the possibility that we can achieve certified everlasting security for various cryptographic primitives with the help of quantum mechanics. For these backgrounds, this thesis investigates the following fascinating question:

Which cryptographic primitives can achieve certified everlasting security with the help of quantum mechanics?

1.2 Cryptographic Primitives

In this thesis, we formalize certified everlasting security for several cryptographic primitives and show the constructions. Before explaining the contributions of this thesis in more detail, we review the notion of cryptographic primitives and their quantum generalizations.

Public-Key Encryption. Diffie and Hellman [DH76] introduced the concept of public-key encryption (PKE). In a PKE scheme, a key pair consisting of a public key pk and a secret key sk is generated. Anyone with access to the public key pk can encrypt a message M , producing a ciphertext $ct(M)$. The receiver, in possession of the corresponding secret key sk , can decrypt the ciphertext to recover the original message M .

Informally, the security of PKE requires that, without knowledge of the secret key sk , any adversary cannot extract any meaningful information about M from $ct(M)$.

To precisely capture a security requirement, modern cryptography rigorously formalizes security in a mathematical way without ambiguity. Goldwasser and Micali [GM82] laid the groundwork for this by introducing semantic security. Informally, a semantically secure PKE scheme ensures that any information that can be efficiently computed from $ct(M)$ could be simulated just as efficiently without access to $ct(M)$. In other words, the ciphertext $ct(M)$ reveals no information about the message M because any potential knowledge that could be extracted from it can be simulated without seeing the ciphertext. It is important to emphasize that multiple, often incomparable, security notions exist for cryptographic primitives. In the context of PKE, beyond semantic security, other security notions have been explored, such as indistinguishability under chosen plaintext attacks (IND-CPA) [GM82] and indistinguishability under chosen ciphertext attacks (IND-CCA) [RS92]. In some cases, these security notions are proven to be equivalent, while in others, one is strictly stronger than the other.

A series of works [RSA78, Rab79, GM82] provided the constructions of PKE schemes, primarily based on the hardness of factoring-related problems. These constructions rely on the assumption that certain mathematical problems, such as integer factorization, are computationally intractable. However, the security of these schemes is broken if these assumptions are proven false. For instance, Shor [Sho94] introduced a quantum algorithm that efficiently solves the factoring problem, a task previously believed to be infeasible for classical computers. As a result, once large-scale quantum computers become practical, factoring-based PKE schemes will no longer be secure. This potential vulnerability has driven the cryptographic community to explore PKE schemes that can withstand quantum attacks. Regev [Reg05] proposed a PKE scheme based on the hardness of the Learning with Errors (LWE) problem, which is currently believed to be resistant even to quantum algorithms. This has made LWE-based encryption a promising candidate for post-quantum cryptography.

Attribute-Based Encryption and Predicate Encryption. Attribute-Based Encryption (ABE) and Predicate Encryption (PE) are advanced encryption that extend the traditional PKE scheme by introducing attribute-based access control mechanisms. In both ABE and PE, the ability to decrypt a ciphertext is governed not solely by a secret key, but also by a combination of attributes associated with both the ciphertext and the decryption key. More precisely, each ciphertext $ct_x(M)$ is encrypted with respect to an attribute x in addition to the plaintext message M , while each secret key sk_f is tied to a predicate function f . The receiver with sk_f can successfully decrypt the ciphertext $ct_x(M)$ and recover the message M if and only if the predicate $f(x) = 1$ holds. This mechanism provides fine-grained access control, enabling encryption policies that depend on specific attributes of the recipient. The security of ABE requires that the plaintext M is not revealed to unauthorized parties. More formally, an adversary in possession of a set of secret keys $\{sk_{f_i}\}_{i \in [n]}$ is not able to learn anything about the plaintext M unless at least one of their predicate functions f_i evaluates to 1 for the given attribute x . PE additionally requires that the information of attribute x is not revealed. This ensures that even with multiple keys, unauthorized access to the message remains computationally infeasible, unless the attributes meet the required predicates. ABE and PE are particularly useful in scenarios where access to encrypted data must be restricted based on complex policies. It generalizes several cryptographic primitives by enabling more expressive and flexible access control models compared to traditional encryption schemes.

The concept of ABE was proposed by Sahai and Waters [SW05]. Several constructions of ABE have been proposed with different flavors in terms of efficiency, security guarantees, and diversifying security assumptions [GPSW06, BSW07, GGH⁺13b, GVW13, BGG⁺14]. Gorbunov, Vaikuntanathan, and Wee [GVW15b] gave a construction of PE from the LWE assumption. Wichs and Zirdelis [WZ17], and Goyal,

Koppula, and Waters [GKW17] independently introduce lockable obfuscation, which can be instantiated by the LWE assumption, and show how to compile ABE to PE by using lockable obfuscation.

Functional Encryption. Functional encryption (FE) is an extension of ABE and PE. In the primitive, one can generate a secret key sk_f of a function f and a ciphertext $ct(M)$ of a message M . A receiver with a ciphertext $ct(M)$ and a secret key sk_f can obtain $f(M)$. On the other hand, it cannot obtain any information beyond $f(M)$. The concept of FE is proposed by Boneh, Sahai and Waters [BSW11]. Garg et al [GGH⁺13a] constructed the first (selectively secure) FE scheme for all circuits based on the existence of indistinguishability obfuscation (iO) and OWFs. Waters [Wat15] and Ananth et al. [ABSV15] proposed fully secure FE schemes for all circuits based on the existence of iO and OWFs. Although these schemes are fully secure, their constructions rely on the existence of iO, which is a relatively strong assumption. For this reason, several works study whether we can construct FE from weaker computational assumptions by restricting functionality or security requirements. The sequence of works [GVW12, AR17, AV19] construct bounded collusion FE from standard PKE. In the bounded collusion FE, the number of secret keys that a receiver obtains is determined in advance. If a receiver receives more keys than the predetermined number, the security is no longer ensured and the receiver may obtain information of M from the ciphertext $ct(M)$. The sequence of works [ALS16, ABDP15] study inner product FE from specific mathematical assumptions such as LWE. In the inner product FE, functionality is limited, and a receiver with a ciphertext $ct(M)$ of a message M and a secret key sk_X associated with X can obtain inner product $\langle X, M \rangle$, but cannot obtain any other information.

Commitments. A commitment scheme is a fundamental cryptographic primitive that allows a sender to "commit" to a chosen message while keeping it hidden from the receiver, with the ability to later "reveal" the committed value. More precisely, in commitment schemes, the sender first generates a commitment $c(m)$ for a message m and a decommitment d , and sends $c(m)$ to the receiver. In the reveal phase, the sender provides d and m , and the receiver can verify that the original committed value is m by using $c(m)$, d and m . Commitment schemes require hiding and binding properties. Hiding property ensures that the commitment $c(m)$ should not reveal any information about the message m to the receiver before the decommitment d is provided. This ensures that the receiver cannot learn the message before the sender chooses to reveal it. Binding property ensures that once the sender has committed to a message m by sending $c(m)$, it should be infeasible for the sender to change their mind and successfully convince the receiver that they had committed to a different message m^* . This prevents the sender from altering the committed message after the commitment phase.

We cannot achieve information theoretical hiding and information theoretical binding at the same time. As a consequence, if the binding property is information theoretical security, then the hiding property must be computational security, and vice versa. Impagliazzo and Luby [IL89] showed that commitment schemes imply the existence of one-way functions (OWFs). Naor [Nao91] showed how to construct a commitment scheme that is computationally hiding and information-theoretically binding using pseudorandom generators, which are equivalent to OWFs. On the other hand, Haitner and Reingold [HR07] showed how to construct a scheme that is information-theoretically hiding but computationally binding from OWFs.

Zero-Knowledge Proof. Zero-knowledge proof is a cryptographic protocol that allows a prover to convince a verifier that a given mathematical statement is true, without revealing any information beyond the validity of the statement itself. In a zero-knowledge proof, there are two security requirements. The first is *soundness*, which ensures that if the statement is false, no prover can convince the verifier of its truth, except with

negligible probability. The second is *zero-knowledge*, which guarantees that the verifier learns nothing beyond the fact that the statement is true.

It is known that we can construct a zero-knowledge protocol with computational zero-knowledge and computational soundness under specific cryptographic assumptions. However, achieving both *information-theoretic* soundness and *information-theoretic* zero-knowledge simultaneously is impossible unless the polynomial hierarchy collapses [For87]. It is also conjectured that even quantum protocols cannot achieve both properties simultaneously [MW18]. In fact, previous constructions of zero-knowledge proof for **NP** guarantee that at least one of these properties holds only against computationally bounded adversaries. The first zero-knowledge proof for **NP** was introduced by Goldreich, Micali, and Wigderson [GMW87], based on the existence of commitment schemes. Goldreich and Kahan [GK96] later improved upon this by constructing constant-round zero-knowledge proofs for **NP**.

Several works [VDG98, DFS04] point out a technical challenge in proving the security of zero-knowledge protocols against quantum adversaries due to the difficulty of applying classical *rewinding* techniques, which are commonly used to simulate interactions between the prover and verifier. Watrous [Wat09] gave the first construction of a zero-knowledge protocol for **NP** secure against quantum adversaries. More recent works, such as [CCY21, LMS22], have further developed techniques to analyze and prove the security of zero-knowledge protocols in more general quantum settings. Beyond classical complexity classes, zero-knowledge proofs for quantum complexity classes have been explored. Broadbent, Ji, Song, and Watrous [BJSW16] constructed the first zero-knowledge proof for **QMA**, the quantum analogue of **NP**. Building on this, Broadbent and Grilo [BG20] introduced a simplified and elegant construction of zero-knowledge proofs for **QMA** using the concept of local simulatability [GSY19], which provides a powerful framework for proving the security of quantum protocols.

Quantum Cryptographic Primitives. Beyond classical cryptographic primitives, numerous works have explored quantum generalizations of cryptographic primitives, which enable cryptographic tasks that are fundamentally impossible in the classical setting. These quantum cryptographic primitives typically consider the situation, where at least one participating entity possesses a quantum computer, and they leverage quantum mechanical properties to achieve enhanced security and functionality. For example, in a quantum generalization of PKE, the key pair comprising a public key pk , a secret key sk , and the ciphertext $ct(M)$ of a message M may consist of quantum states. In such protocols, at least one of the algorithms involved in key generation, encryption, or decryption is implemented by a quantum algorithm. This extension opens up new functionalities and security guarantees that are unachievable within the classical framework. Broadly, current applications of quantum cryptographic primitives can be categorized into two motivations.

The first motivation is the realization of security properties impossible in classical cryptography. Certified everlasting security, which is a primary focus of this thesis, serves as one such example. Another notable example is unclonability, a uniquely quantum feature that offers advantages not available in classical cryptography. A series of works [Got04, BL20, AK21, AKL⁺22] have investigated ciphertext unclonability, which ensures that a quantum ciphertext $ct(M)$ cannot be cloned to produce multiple copies that encrypt the same message M . More formally, if Alice receives a quantum ciphertext $ct(M)$ for a uniformly random M , and produces two quantum states ρ_B and ρ_C , Bob and Charlie, who receive ρ_B and ρ_C along with the secret key, must independently guess the message M without further communication. The unclonability property ensures that no pair of Alice, Bob, and Charlie can correctly guess M under this scenario. Based on Gottesman [Got04], Broadbent and Lord [BL20] formalized this notion and showed how to construct a one-time quantum secret-key encryption (SKE) scheme with ciphertext unclonability without relying on any assumptions. Ananth and Kaleoglu [AK21] later constructed quantum PKE with ciphertext unclonability using

standard PKE and one-time quantum SKE with this unclonability property. Other works [AKL⁺22, AKL23] have explored stronger forms of ciphertext unclonability. It’s important to note that achieving ciphertext unclonability is unique to quantum encryption schemes because if $\text{ct}(M)$ were a classical string, Alice could trivially copy $\text{ct}(M)$ and provide it to both Bob and Charlie, who could then easily decrypt the message M using the secret key. Hence, ciphertext unclonability is a uniquely quantum feature.

For the second reason, there exists a potential that we can construct cryptographic primitives based on weaker assumptions. As explained so far, in the current constructions of cryptographic primitives, their security relies on the computational assumption that a certain mathematical problem is hard to solve with a current computer. Hence, reducing security to more reliable assumptions is crucial in cryptography. In the classical case, the existence of one-way functions (OWFs), which are easy to compute but hard to invert, is one of the most reliable assumptions in cryptography. Many useful primitives such as secret-key encryption, digital signature, and commitments can be constructed from OWFs, and the existence of OWFs is implied by almost all classical cryptographic primitives such as PKE, FE, and commitment schemes. Recent works show the possibility that weaker assumptions are sufficient to construct useful cryptographic primitives in the realm of quantum. Ji, Liu, and Song [JLS18] introduce a notion of pseudorandom quantum state generators (PRSG), which can be seen as a quantum generalization of pseudorandom generators (PRGs), and show that it can be constructed from OWFs. Kretchmer [Kre21] shows that there exists an oracle $\mathcal{O}$ such that PRSGs exists, but $\text{BQP}^{\mathcal{O}} = \text{QMA}^{\mathcal{O}}$ and hence OWFs does not exist. Morimae and Yamakawa [MY22] and Ananth, Qian and Yuen [AQY22] independently concurrently construct quantum commitment from PRSGs. It is shown that several cryptographic primitives such as oblivious transfer and multi-party computation are constructed from quantum commitments [GLSV21, BCKM21, Yan22, BCQ23]. It is conjectured that EFI may exist relative to a random oracle and any classical oracle [LMW24].

1.3 Summary of Contributions

As reviewed in the previous section, there has been a wide variety of cryptographic primitives, and recently, quantum variants of them has been introduced until now. However, the security of almost all protocols is a computational one, and hence its security will be broken in the future when novel algorithms are found or computing devices are drastically improved. What is worse, information theoretical security is shown as impossible to achieve for many cryptographic primitives.

In this thesis, we formalize certified everlasting secure versions for many cryptographic primitives, which can be seen as a compromise of information theoretical security. Then, we show how to achieve certified everlasting security for those primitives. Our contributions are summarized as follows:

Certified Everlasting Zero-Knowledge Proof for QMA. In the Chapter 3, we formalize and construct the certified everlasting version of zero-knowledge proof for **QMA**. Our contributions are summarized as follows:

1. We formalize and construct a certified everlasting version of a commitment scheme. In this new commitment scheme, binding is statistical but hiding is computational. Furthermore, the hiding property becomes information-theoretical security once the receiver has issued a valid certificate that shows that the receiver has deleted the committed information.

We construct a certified everlasting secure version of a commitment scheme using a commitment scheme with statistical binding and computational hiding. We prove its security in the quantum random oracle model [BDF⁺11].

2. We formalize a certified everlasting version of zero-knowledge proof for **QMA**. In this new zero-knowledge proof for **QMA**, the computational zero-knowledge property is satisfied. Furthermore, the verifier can issue a classical certificate that shows that the verifier has deleted its information. If the certificate is valid, even unbounded malicious verifier can no longer learn anything beyond the validity of the statement. For the construction, we use a certified everlasting secure version of a commitment scheme, which we newly introduced.

Let us explain the comparison between our contributions and previous works. There are several constructions of zero-knowledge proof for **QMA** [BSW16, BG20]. However, all previous constructions do not achieve information-theoretical zero-knowledge properties when the soundness is information-theoretical one. Indeed, it is believed that we cannot construct a zero-knowledge proof for **NP** (hence **QMA**), where the protocol satisfies information theoretical soundness and zero-knowledge property at the same time [For87, MW18]. Therefore, if the soundness is information-theoretical security, then the zero-knowledge property becomes the computational one. Hence, the adversary could obtain information beyond the validity of the statement if novel algorithms are found or computing devices are drastically improved.

On the other hand, in our certified everlasting zero-knowledge proof for **QMA**, we can check if the verifier deletes encrypted information. Once its deletion is confirmed, even if the verifier's computing resource becomes unbounded, the malicious verifier can no longer learn anything beyond the validity of the statement.

Certified Everlasting Functional Encryption. In the Chapter 4, we formalize and construct the certified everlasting versions of several cryptographic primitives including, PKE, FE, compute-and-compare obfuscation, and so on. Our contributions are summarized as follows:

1. We formally define certified everlasting secure versions of basic cryptographic primitives and construct them. We use those primitives for constructing a certified everlasting secure version of FE. Our contribution is summarized as follows:
 - We introduce a certified everlasting secure version of SKE. We give two constructions of certified everlasting secure SKE from standard SKE. 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.
 - We introduce a certified everlasting secure version of PKE. We give two constructions of certified everlasting secure PKE with the same properties of the SKE constructions above from standard PKE.
 - We introduce a certified everlasting secure version of receiver non-committing encryption (RNCE). We give a construction of certified everlasting secure RNCE from certified everlasting PKE.
 - We introduce a certified everlasting secure version of a garbling scheme. We give a construction of certified everlasting secure garbling scheme for **P/poly** from certified everlasting SKE.
2. We formally define a certified everlasting secure version of adaptively secure bounded collusion resistant FE for **NC**¹, and construct it from a certified everlasting secure version of SKE, PKE, RNCE and a garbling scheme. In the FE, the adversary can obtain many copies of secret keys whenever it wants while the number of secret keys that it can obtain is determined in advance. The adversary with secret keys $\{sk_{f_i}\}_{i \in [n]}$ and a quantum ciphertext $ct(m)$ can issue a deletion certificate of $ct(M)$. Once a

deletion certificate is issued, the adversary cannot obtain information more than $\{f_i(M)\}_{i \in [n]}$ even if its computing power becomes unbounded.

3. We formally define a certified everlasting secure version of adaptively secure collusion-resistant FE for **P/poly** and construct it from indistinguishability obfuscation (iO) and one-way functions (OWFs). Different from bounded collusion-resistant FE, the adversary can obtain an arbitrary number of secret keys $\{\text{sk}_{f_i}\}_{i \in [n]}$ whenever the adversary wants. The adversary with $\{\text{sk}_{f_i}\}_{i \in [n]}$ and a quantum ciphertext $\text{ct}(M)$ can issue a deletion certificate of the quantum ciphertext $\text{ct}(M)$. Once a deletion certificate is issued the adversary cannot obtain information more than $\{f_i(M)\}_{i \in [n]}$ even if its computing power becomes unbounded.
4. We construct certified everlasting secure compute-and-compare obfuscation from standard FHE and standard compute-and-compare obfuscation. 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.

Let us explain the comparison between our contributions and previous works. So far, there are several constructions of PKE, FE, and so on, respectively. All of the previous constructions are not secure against computationally unbounded adversaries, and it is shown that we cannot achieve information theoretical security for these primitives even in the quantum regime. Compared to the previous works, in our constructions, we can verify whether the malicious receiver deletes encrypted information correctly or not. Once its deletion is confirmed, even if the malicious receiver's computing resource becomes unbounded, the receiver cannot obtain any additional information.

1.4 Concurrent and independent works

In Chapter 4, we study certified everlasting secure versions of several cryptographic primitives. This result has several concurrent and independent works. Let us explain the relationship between them.

On Bounded Collusion Resistant Functional Encryption. Results in Section 4.4 were obtained concurrently and independently with the results of [BK23]. In Section 4.4, we introduce and construct certified everlasting secure versions of bounded collusion-resistant functional encryption, and SKE, PKE, and garbling schemes. Bartusek and Khurana [BK23] concurrently and independently obtained similar results. 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 4.2.2.) 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 $f(m)$.

On Unbounded Collusion Resistant Functional Encryption Results in Sections 4.3 and 4.5 rely on [BK23] while our certified everlasting secure bounded collusion-resistant FE (and its building block SKE, PKE, garbling, and RNCE) schemes are concurrent and independent work.¹ In Sections 4.3 and 4.5, we study unbounded collusion-resistant FE, compute-and-compare obfuscation, and PE. Those three schemes were added after the paper by Bartusek and Khurana was made public. Their work does not consider FE, compute-and-compare obfuscation, and PE, and it is unclear how to apply their compiler to those primitives. 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 we cannot recover $P(a)$ from $P(a \oplus r)$ and r . 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.

On Concurrent work of Unbounded Collusion Resistant FE Bartusek, Garg, Goyal, Khurana, Malavolta, Raizes, and Roberts [BGK⁺24] concurrently and independently obtained adaptively certified everlasting secure collusion-resistant FE for 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 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.

1.5 Organization

This thesis is organized as follows: In Chapter 2, we review cryptographic primitives, which we will use in this thesis.

In Chapter 3, we study a certified everlasting version of zero-knowledge proof for **QMA**. Section 3.1 gives an introduction and a technical overview of our results. Section 3.2 studies a certified everlasting secure version of a commitment scheme. In Section 3.2.1, we formalize a certified everlasting secure version of a commitment scheme. In Section 3.2.2, we give the construction and prove its security. Section 3.3 studies a certified everlasting secure version of zero-knowledge proof for **QMA**. In Section 3.3.1, we

¹This is because after [BK23], we add new results and update the original paper [HMNY22a]. Therefore, the content in [HMNY22a] is concurrent and independent work of the work by [BK23].

formalize a certified everlasting secure version of zero-knowledge proof for **QMA**. In Section 3.3.2, we give a construction, and prove its security.

In Chapter 4, we study a certified everlasting version of functional encryption, compute-and-compare obfuscation, and predicate encryption. Section 4.1 gives an introduction of results. In Section 4.2, we give a technical overview of our results. In Section 4.3, we study a collusion-resistant functional encryption with certified everlasting security. Section 4.3.1 provides its definition, Section 4.3.2 gives tool used for our construction, and Section 4.3.3 presents our construction and security proof. In Section 4.4, we study a bounded collusion-resistant functional encryption with certified everlasting security. Definition 4.4.1 provides its definition, Section 4.4.2 presents 1-bounded construction with non-adaptive security, Section 4.4.3 improves it to adaptive secure one, and Section 4.4.4 improves it to q -bounded secure one. In Section 4.5, we study a compute-and-compare obfuscation with certified everlasting security. Section 4.5.1 provides its definition, and Section 4.5.2 presents construction. In Section 4.6, we study predicate encryption with certified everlasting security. Section 4.6.1 provides its definition, and Section 4.6.2 provides construction.

Finally, in Chapter 5, we conclude this thesis.

Chapter 2

Preliminaries

2.1 Notations

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

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 $\{1, 2, \dots, N\}$. 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 constant c , there exists $\lambda_0 \in \mathbb{N}$ such that for any $\lambda > \lambda_0$, $f(\lambda) < \lambda^{-c}$. We write $f(\lambda) \leq \text{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function.

2.2 Cryptographic Tools

In cryptography, cryptographic primitives are formalized in a mathematical way. In this section, we review the definitions of cryptographic tools used in this paper.

Public Key Encryption (PKE).

Definition 2.2.1 (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 $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ 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 $\mathcal{SK} := \{0, 1\}^{r(\lambda)}$.

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

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

$\text{Dec}(\text{sk}, \text{ct}) \rightarrow m' \text{ or } \perp$: The decryption algorithm takes as input sk and ct , and outputs a plaintext m' or $\perp$.

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

Definition 2.2.2 (Decryption Correctness for PKE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,*

$$\Pr \left[\text{Dec}(\text{sk}, \text{ct}) \neq m \mid \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda) \\ \text{ct} \leftarrow \text{Enc}(\text{pk}, m) \end{array} \right] \leq \text{negl}(\lambda).$$

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

Definition 2.2.3 (OW-CPA Security for PKE). *Let ℓ be a polynomial of the security parameter λ . Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a PKE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ow-cpa}}(\lambda)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger computes $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *The challenger samples $(m^1, \dots, m^\ell) \leftarrow \mathcal{M}^\ell$, computes $\text{ct}^i \leftarrow \text{Enc}(\text{pk}, m^i)$ for all $i \in [\ell]$ and sends $\{\text{ct}^i\}_{i \in [\ell]}$ 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

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

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

Definition 2.2.4 (IND-CPA Security for PKE). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a PKE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger generates $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$, 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 $\text{ct} \leftarrow \text{Enc}(\text{pk}, m_b)$, 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

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

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.

Non-interactive commitment.

Definition 2.2.5 (Non-Interactive Commitment (Syntax)). Let λ be the security parameter and let p, q and r be some polynomials. A (classical) non-interactive commitment scheme consists of a single PPT algorithm Commit with plaintext space $\mathcal{M} := \{0, 1\}^{p(\lambda)}$, randomness space $\{0, 1\}^{q(\lambda)}$ and commitment space $\mathcal{C} := \{0, 1\}^{r(\lambda)}$ satisfying two properties:

Perfect binding: For every $(r_0, r_1) \in \{0, 1\}^{q(\lambda)} \times \{0, 1\}^{q(\lambda)}$ and $(m, m') \in \mathcal{M}^2$ such that $m \neq m'$, we have that $\text{Commit}(m; r_0) \neq \text{Commit}(m'; r_1)$, where $(\text{Commit}(m; r_0), \text{Commit}(m'; r_1)) \in \mathcal{C}^2$.

Unpredictability: Let $\Sigma := \text{Commit}$. For any QPT adversary $\mathcal{A}$, we define the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{unpre}}(\lambda)$.

1. The challenger chooses $R \leftarrow \mathcal{M}$ and $R' \leftarrow \{0, 1\}^{q(\lambda)}$, computes $\text{com} \leftarrow \text{Commit}(R; R')$, and sends com to $\mathcal{A}$.
2. $\mathcal{A}$ outputs R^* . The output of the experiment is 1 if $R^* = R$. Otherwise, the output of the experiment is 0.

We say that the commitment is unpredictable if for any QPT adversary $\mathcal{A}$, it holds that

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

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

Remark 2.2.6. The unpredictability is a weaker version of computational hiding. We define unpredictability instead of computational hiding since this suffices for our purpose.

A non-interactive commitment scheme that satisfies the above definition exists assuming the existence of injective one-way functions or perfectly correct public key encryption [LS19]. Alternatively, we can also instantiate it based on random oracles.

Secret Key Encryption (SKE).

Definition 2.2.7 (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 $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ with plaintext space $\mathcal{M} := \{0, 1\}^n$, ciphertext space $\mathcal{C} := \{0, 1\}^{p(\lambda)}$, and secret key space $\mathcal{SK} := \{0, 1\}^{q(\lambda)}$.

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

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

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

We require that a SKE scheme satisfies correctness defined below.

Definition 2.2.8 (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 $m \in \mathcal{M}$,*

$$\Pr \left[\text{Dec}(\text{sk}, \text{ct}) \neq m \mid \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ \text{ct} \leftarrow \text{Enc}(\text{sk}, m) \end{array} \right] \leq \text{negl}(\lambda).$$

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

$$\Pr \left[\text{Dec}(\text{sk}_2, \text{ct}) \neq \perp \mid \begin{array}{l} \text{sk}_2, \text{sk}_1 \leftarrow \text{KeyGen}(1^\lambda) \\ \text{ct} \leftarrow \text{Enc}(\text{sk}_1, m) \end{array} \right] \leq \text{negl}(\lambda).$$

Remark 2.2.9. 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 A.5.2. 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 2.2.10 (OW-CPA Security for SKE). *Let ℓ be a polynomial of the security parameter λ . Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a SKE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ow-cpa}}(\lambda)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *$\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\text{ct} \leftarrow \text{Enc}(\text{sk}, m)$ and returns ct to $\mathcal{A}$. $\mathcal{A}$ can repeat this process polynomially many times.*
3. *The challenger samples $(m^1, \dots, m^\ell) \leftarrow \mathcal{M}^\ell$, computes $\text{ct}^i \leftarrow \text{Enc}(\text{sk}, m^i)$ for all $i \in [\ell]$ and sends $\{\text{ct}^i\}_{i \in [\ell]}$ to $\mathcal{A}$.*
4. *$\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\text{ct} \leftarrow \text{Enc}(\text{sk}, m)$ 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

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

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

Definition 2.2.11 (IND-CPA Security for SKE). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a SKE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *$\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $\text{ct} \leftarrow \text{Enc}(\text{sk}, m)$ 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 $ct \leftarrow \text{Enc}(\text{sk}, m_b)$ and sends ct to $\mathcal{A}$.
5. $\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $ct \leftarrow \text{Enc}(\text{sk}, m)$ 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

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

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

Definition 2.2.12 (Ciphertext Pseudorandomness for SKE). Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a SKE scheme whose ciphertext space is $\{0, 1\}^\ell$. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ct-pr}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.

1. The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.
2. $\mathcal{A}$ sends an encryption query m_i to the challenger. If $b = 0$, the challenger computes $\text{ct}_i \leftarrow \text{Enc}(\text{sk}, m_i)$ and returns ct_i to $\mathcal{A}$. If $b = 1$, the challenger chooses $\text{ct}_i \leftarrow \{0, 1\}^\ell$ 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

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

Theorem 2.2.13. If OWFs exist, there exists an SKE scheme that is ciphertext pseudorandom.

Pseudorandom generators.

Definition 2.2.14 (Pseudorandom Generator). A pseudorandom generator (PRG) $\text{PRG} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{\lambda + \ell(\lambda)}$ 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

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

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

Theorem 2.2.15 ([HILL99]). If there exists a OWF, there exists a PRG.

Pseudorandom Functions.

Definition 2.2.16 (Pseudorandom Function). Let $\{F_K : \{0, 1\}^{\ell_1} \rightarrow \{0, 1\}^{\ell_2} \mid K \in \{0, 1\}^\lambda\}$ 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 $\text{negl}(\cdot)$ such that it holds that

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

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

Theorem 2.2.17 ([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.

Attribute-Based Encryption.

Definition 2.2.18 (KP-ABE (Syntax)). A key-policy ABE (KP-ABE) scheme is a tuple of PPT algorithms (Setup, KeyGen, Enc, Dec) with a class of predicates $\mathcal{P}_n$ (represented as class of boolean circuits with n input bits), and a message space $\mathcal{M}$.

Setup(1^λ): The setup algorithm takes as input a security parameter 1^λ and outputs a public key pk and a master secret key msk .

KeyGen(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 .

Enc(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 .

Dec(sk_P, ct): The decryption algorithm takes as input a secret key sk_P and a ciphertext ct , and outputs a classical message m' or $\perp$.

Definition 2.2.19 (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 $P(a) = 1$,

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

Definition 2.2.20 (IND-CPA Security of KP-ABE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a KP-ABE scheme for a class predicates $\mathcal{P}_n$ and message space $\mathcal{M}$. We consider the following experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$.

1. The challenger computes $(\text{pk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$ 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 $\text{sk}_{P_i} \leftarrow \text{KeyGen}(\text{msk}, P_i)$. $\mathcal{A}$ 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 $ct_b \leftarrow \text{Enc}(\text{pk}, a, m_b)$. 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

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

We can define similar experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-ind}}(\lambda, b)$ 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

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

Theorem 2.2.21 ([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 2.2.22 (Leveled Fully Homomorphic Encryption). A leveled FHE is a tuple of PPT algorithms $(\text{KeyGen}, \text{Enc}, \text{Eval}, \text{Dec})$ 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 .

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 .

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

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

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

Definition 2.2.23 (Compactness). A classical FHE is compact if there exists a fixed polynomial bound $B(\cdot)$ so that, for all $\lambda \in \mathbb{N}$, any $C \in \mathcal{C}$, and plaintext $x \in \{0, 1\}^n$, it holds that

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

Definition 2.2.24 (Correctness). An FHE scheme is said to be correct for $\mathcal{C}$ if for any $\lambda, n \in \mathbb{N}, C \in \mathcal{C}, x = (x_1, \dots, x_n) \in \{0, 1\}^n$,

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

Definition 2.2.25 (Security of FHE). An FHE scheme $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ 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:

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

Theorem 2.2.26 ([BV14, GSW13]). If the LWE assumption holds, there exists leveled FHE.

Indistinguishability Obfuscation

Definition 2.2.27 (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 $\Pr[\forall x, C_0(x) = C_1(x) \wedge |C_0| = |C_1| \mid (C_0, C_1, \text{aux}) \leftarrow \text{Sampler}(1^\lambda)] > 1 - \text{negl}(\lambda)$, then we have

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

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 2.2.28 (Compute-and-Compare Circuits). A compute-and-compare circuit $\text{CC}[P, \text{lock}, m]$ is of the form

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

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 2.2.29 (Compute-and-Compare Obfuscation). A PPT algorithm CC.Obf is a secure obfuscator for the family of distributions $D = \{D_{\text{param}}\}_{\text{param}}$ 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

$$\Pr[\forall x, \tilde{P}(x) = \text{CC}[P, \text{lock}, m](x) \mid \tilde{P} \leftarrow \text{CC.Obf}(1^\lambda, P, \text{lock}, m)] \geq 1 - \text{negl}(\lambda).$$

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

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

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

Theorem 2.2.30 ([GKW17, WZ17]). *If the LWE assumption holds, there exists compute-and-compare obfuscation for all families of distributions $D = \{D_{\text{param}}\}$, where each D_{param} outputs uniformly random lock value lock independent of P and aux .*

Lemma 2.2.31 (Difference Lemma [Sho04]). *Let A, B, F be events defined in some probability distribution, and suppose $\Pr[A \wedge \bar{F}] = \Pr[B \wedge \bar{F}]$. Then $|\Pr[A] - \Pr[B]| \leq \Pr[F]$.*

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

Definition 2.2.32 (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 $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ 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 $\mathcal{D} := \{0, 1\}^{r(\lambda)}$.*

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

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

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

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

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

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

Definition 2.2.33 (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 $m \in \mathcal{M}$,*

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

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

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

We additionally require verification correctness with QOTP in this work. This is because we need it for the construction of FE in Section 4.4.3. This notion means that even if we encrypt a ciphertext with quantum one-time pad (QOTP), we can run the original deletion algorithm 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 2.2.34 (Verification Correctness with QOTP). *There exists a negligible function negl and a PPT algorithm Recover such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr \left[\text{Vrfy}(\text{sk}, \text{cert}^*) = \perp \mid \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ \text{ct} \leftarrow \text{Enc}(\text{sk}, m) \\ a, b \leftarrow \{0, 1\}^{p(\lambda)} \\ \widetilde{\text{cert}} \leftarrow \text{Del}(Z^b X^a \text{ct} X^a Z^b) \\ \text{cert}^* \leftarrow \text{Recover}(a, b, \widetilde{\text{cert}}) \end{array} \right] \leq \text{negl}(\lambda).$$

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

Definition 2.2.35 (Certified Deletion Security for One-Time SKE with Certified Deletion). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a one-time SKE scheme with certified deletion. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{otsk-cert-del}}(\lambda, b)$ against an unbounded adversary $\mathcal{A}$.*

1. *The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *$\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.*
3. *The challenger computes $\text{ct} \leftarrow \text{Enc}(\text{sk}, m_b)$ and sends ct to $\mathcal{A}$.*
4. *$\mathcal{A}$ sends cert to the challenger.*
5. *The challenger computes $\text{Vrfy}(\text{sk}, \text{cert})$. If the output is $\perp$, the challenger sends $\perp$ 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

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

Theorem 2.2.36 ([BI20]). *There exists one-time SKE with certified deletion that satisfies Definitions 2.2.32 to 2.2.35 exists unconditionally.*

2.3 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 $A^{x_1} \otimes A^{x_2} \otimes \cdots \otimes A^{x_n}$. The trace distance between two states ρ and σ is given by $\frac{1}{2}\|\rho - \sigma\|_{\text{tr}}$, where $\|A\|_{\text{tr}} := \text{tr} \sqrt{A^\dagger A}$ is the trace norm. If $\frac{1}{2}\|\rho - \sigma\|_{\text{tr}} \leq \epsilon$, we say that ρ and σ are ϵ -close. If $\epsilon \leq \text{negl}(\lambda)$, then we say that ρ and σ are statistically indistinguishable.

Lemma 2.3.1 (Quantum Rewinding Lemma [Wat09]). *Let Q be a quantum circuit that acts on an n -qubit state $|\psi\rangle$ and an m -qubit auxiliary state $|0^m\rangle$. Let $p(\psi) := ||(\langle 0| \otimes I)Q(|\psi\rangle \otimes |0^m\rangle)||^2$ and $|\phi(\psi)\rangle := \frac{1}{\sqrt{p(\psi)}}((\langle 0| \otimes I)Q(|\psi\rangle \otimes |0^m\rangle))$. Let $p_0, q \in (0, 1)$ and $\epsilon \in (0, \frac{1}{2})$ such that $|p(\psi) - q| < \epsilon$, $p_0(1 - p_0) < q(1 - q)$, and $p_0 < p(\psi)$. Then there is a quantum circuit R of size at most $O\left(\frac{\log(\frac{1}{\epsilon})\text{size}(Q)}{p_0(1 - p_0)}\right)$ such that on input $|\psi\rangle$, R computes a quantum state $\rho(\psi)$ that satisfies $\langle \phi(\psi)|\rho(\psi)|\phi(\psi)\rangle \geq 1 - 16\epsilon \frac{\log^2(\frac{1}{\epsilon})}{p_0^2(1 - p_0)^2}$.*

Quantum Random Oracle. We use the quantum random oracle model (QROM) [BDF⁺11] to construct SKE and PKE with certified everlasting deletion in Appendices A.3.2 and A.3.4, 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 2.3.2 (One-Way to Hiding Lemma [AHU19]). *Let $S \subseteq \mathcal{X}$ be a random subset of $\mathcal{X}$. Let $G, H : \mathcal{X} \rightarrow \mathcal{Y}$ 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 $|\Pr[\mathcal{A}^G(z) = 1] - \Pr[\mathcal{A}^H(z) = 1]| \leq 2q\sqrt{\Pr[\mathcal{B}^H(z) \in S]}$.*

Quantum Teleportation. We use quantum teleportation to prove that our construction of the FE scheme in Section 4.4.3 satisfies adaptive security.

Lemma 2.3.3 (Quantum Teleportation). *Suppose that we have N Bell pairs between registers A and B , i.e., $\frac{1}{\sqrt{2^N}} \sum_{s \in \{0,1\}^N} |s\rangle_A \otimes |s\rangle_B$, and let ρ be an arbitrary N -qubit quantum state in register C . Suppose that we measure j -th qubits of C and A in the Bell basis and let $(x_j, z_j) \in \{0, 1\} \times \{0, 1\}$ be the measurement outcome for all $j \in [N]$. Let $x := x_1 || x_2 || \cdots || x_N$ and $z := z_1 || z_2 || \cdots || z_N$. Then (x, z) is uniformly distributed over $\{0, 1\}^N \times \{0, 1\}^N$. Moreover, conditioned on the measurement outcome (x, z) , the resulting state in B is $X^x Z^z \rho Z^z X^x$.*

CSS code. We explain basics of CSS codes. CSS codes are used only in the constructions of SKE and PKE with certified everlasting deletion (Appendix A.3.3 and Appendix A.3.5), 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 ¹ 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, $\bmod C$ 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 $x \in \{0, 1\}^q$.

QMA and k-SimQMA

Definition 2.3.4 (QMA). We say that a promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in **QMA** if there exist a polynomial p , a QPT algorithm V , and $0 \leq \beta < \alpha \leq 1$ with $\alpha - \beta \geq \frac{1}{\text{poly}(|x|)}$ such that

Completeness: For any $x \in A_{\text{yes}}$, there exists a quantum state w of $p(|x|)$ -qubit (called a witness) such that

$$\Pr[V(x, w) = \top] \geq \alpha.$$

Soundness: For any $x \in A_{\text{no}}$ and any quantum state w of $p(|x|)$ -qubit,

$$\Pr[V(x, w) = \top] \leq \beta.$$

For any $x \in A_{\text{yes}}$, $R_A(x)$ is the (possibly infinite) set of all quantum states w such that $\Pr[V(x, w) = \top] \geq \frac{2}{3}$.

A complexity class of **k-SimQMA** is introduced, and proven to be equal to **QMA** in [BG20].

Definition 2.3.5 (k-SimQMA [BG20]). A promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is in **k-SimQMA** with soundness $\beta(|x|) \leq 1 - \frac{1}{\text{poly}(|x|)}$, if there exist polynomials m and n such that given $x \in A_{\text{yes}}$, there is an efficient deterministic algorithm that computes $m(|x|)$ k -qubit POVMs $\{\Pi_1, I - \Pi_1\}, \dots, \{\Pi_{m(|x|)}, I - \Pi_{m(|x|)}\}$ such that:

Simulatable completeness: If $x \in A_{\text{yes}}$, there exists an $n(|x|)$ -qubit state ρ_{hist} , which we call a simulatable witness, such that for all $c \in [m]$, $\text{Tr}(\Pi_c \rho_{\text{hist}}) \geq 1 - \text{negl}(|x|)$, and there exists a set of k -qubit density matrices $\{\rho_{\text{sim}}^{x, S}\}_{S \subseteq [n(|x|)], |S|=k}$ that can be computed in polynomial time from x and ρ_{hist} such that $\|\text{Tr}_{\bar{S}}(\rho_{\text{hist}}) - \rho_{\text{sim}}^{x, S}\|_{\text{tr}} \leq \text{negl}(|x|)$.

Soundness: If $x \in A_{\text{no}}$, for any $n(|x|)$ -qubit state ρ , $\frac{1}{m} \sum_{c \in [m]} \text{Tr}(\Pi_c \rho) \leq \beta(|x|)$.

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

Chapter 3

Certified Everlasting Zero-Knowledge Proof for QMA

3.1 Introduction

In this chapter, we study a certified everlasting secure version of zero-knowledge for **QMA**. Several works show how to construct a zero-knowledge proof for **QMA**. However, in the previous constructions, the zero-knowledge property is computational security if the soundness satisfies the information-theoretical one. Therefore, the adversary could obtain the information of the statement in the future when the underlying computational assumptions are found to be wrong or computing devices are drastically improved. In this chapter, we formalize a certified everlasting version of zero-knowledge property and show how to construct a zero-knowledge proof for **QMA** with certified everlasting zero-knowledge and information-theoretical soundness.

In Section 3.1.1, we give the background in more detail. In Section 3.1.2, we explain our results. In Section 3.1.3, we give a technical overview of our certified everlasting zero-knowledge proof for **QMA**. In Section 3.2, we formalize a certified everlasting version of quantum bit commitments and show how to construct it from standard classical bit commitments. In Section 3.3, we formalize a certified everlasting version of zero-knowledge proof for **QMA**, and show how to construct it from a certified everlasting version of quantum bit commitments. In the following, we explain the background of these results in detail.

3.1.1 Background

Zero-knowledge [GMR89] is a foundational concept in cryptography and computer science, which informally states that the verifier learns nothing beyond the validity of the statement. This concept has been extensively studied in classical cryptography, and recent developments have expanded its scope to quantum cryptography. In known constructions of classical zero-knowledge protocols for **NP**, either the zero-knowledge property or soundness is guaranteed only against computationally bounded adversaries. In fact, achieving both statistical zero-knowledge and statistical soundness simultaneously with a classical verifier for **NP** is impossible unless the polynomial-time hierarchy collapses [For87]. It is also conjectured to be impossible in the case of quantum verifiers [MW18].

Broadbent and Islam [BI20] suggested a novel approach to address this issue by achieving "everlasting zero-knowledge" through the use of quantum encryption with certified deletion. Everlasting security, as defined by Unruh [Unr13], ensures that a protocol remains secure as long as the adversary runs in polynomial

time during the protocol execution. Quantum encryption with certified deletion, a primitive introduced by Broadbent and Islam [BI20], allows a classical message to be encrypted into a quantum ciphertext. The receiver can generate a classical certificate that proves they have deleted the quantum ciphertext, rendering it impossible to decrypt the message even if the secret key is later provided. Broadbent and Islam's key insight was to apply this certified deletion security to quantum commitment, encrypting the first message in a standard Σ -protocol. Once the verifier generates a deletion certificate for all commitments not opened by the verifier's challenge, even an unbounded verifier cannot access the unopened commitments' values. However, the formal definition and construction of such a protocol were left open for future work.

There are several challenges in realizing this idea. First, the quantum encryption with certified deletion, as originally proposed, cannot be directly applied to a Σ -protocol because their scheme does not have any binding property. Their ciphertext consists of a classical and a quantum component. The classical part is $m \oplus u \oplus H(r)$, where m is the plaintext, and u and r are random bit strings with H as a hash function. The quantum part is a random BB84 state, with the computational basis states encoding r . The decryption key consists of u and the positions of the computational basis states encoding r , making it non-binding since modifying u can yield a different message. To address this, we need to extend quantum encryption with certified deletion to include a statistical binding property.

Second, defining a meaningful notion of "everlasting zero-knowledge proof" is non-trivial. In fact, everlasting zero-knowledge proofs for **QMA** or even for **NP**, in the sense defined by Unruh [Unr13], are unlikely to exist.¹ The definition of quantum statistical zero-knowledge [Wat09, MW18] requires a simulator that can statistically reproduce the view of a quantum polynomial-time malicious verifier. Therefore, everlasting zero-knowledge, as per Unruh's definition, is equivalent to quantum statistical zero-knowledge, and it is believed that such proofs for **NP** do not exist [MW18]. For example, Menda and Watrous [MW18] constructed an oracle showing that quantum statistical zero-knowledge proofs for even a subclass of **NP** do not exist.

However, this argument does not apply to what we call *certified everlasting zero-knowledge*, where the verifier can issue a classical certificate proving that it has deleted its information. Once this certificate is generated, even an unbounded malicious verifier cannot learn anything beyond the validity of the statement. This is because certified everlasting zero-knowledge does not imply statistical zero-knowledge, as it does not provide security against a verifier that refuses to provide a valid certificate of deletion. This leads to the following question:

*Is it possible to define and construct a certified everlasting zero-knowledge proof for **QMA**?*

3.1.2 Our Results

In this work, we define and construct a certified everlasting zero-knowledge proof for **QMA**. Our approach involves four main steps:

1. We introduce a novel quantum cryptographic primitive, termed *commitment with statistical binding and certified everlasting hiding*. In this primitive, the binding property is statistical, while the hiding property is computational. However, once the receiver provides a valid certificate confirming the deletion of the committed information, the hiding property becomes statistical.

¹It is possible to construct everlasting zero-knowledge *arguments*, which only require computational soundness. In fact, any statistical zero-knowledge argument is an everlasting zero-knowledge argument. While computational soundness may be sufficient for practical purposes, it is theoretically intriguing to explore forms of everlasting zero-knowledge that do not compromise soundness, as we do in this paper.

2. Using secret-key quantum encryption with certified deletion as a building block, we construct this commitment scheme in a black-box manner. The security of our construction is established in the quantum random oracle model [BDF⁺11].
3. We define a new zero-knowledge proof called the *certified everlasting zero-knowledge proof for QMA*. This proof satisfies computational zero-knowledge for **QMA** and includes an additional feature: after the verifier issues a classical certificate confirming the deletion of its information, even an unbounded malicious verifier cannot extract any information beyond the validity of the statement.
4. Finally, we integrate the new commitment scheme into the quantum Σ -protocol for **QMA** proposed by Broadbent and Grilo [BG20], resulting in the construction of a certified everlasting zero-knowledge proof for **QMA**.

We make the following three remarks regarding our results:

First, while the primary contribution of this work is the definition and construction of a certified everlasting zero-knowledge proof for **QMA**, the proposed commitment scheme with statistical binding and certified everlasting hiding is of independent interest. This scheme holds promise for various applications beyond zero-knowledge. Importantly, it is well-known that both binding and hiding properties cannot simultaneously be statistical, even in the quantum setting [LC97, May97], underscoring the significance of our compromise.

Second, this work marks the first cryptographic application of symmetric-key quantum encryption with certified deletion. Although the concept of certified deletion is appealing, it initially lacked concrete cryptographic applications [BI20]. This is partly because scenarios in which the receiver gains access to the private key at a later stage are not usual in cryptography. Hiroka et al. [HMNY21] extended the symmetric-key scheme of Broadbent and Islam [BI20] to public-key encryption, attribute-based encryption, and publicly verifiable schemes, broadening its applicability. However, their extensions rely on computational security, unlike the information-theoretically secure symmetric-key scheme of [BI20]. In contrast, our results provide the first cryptographic applications leveraging information-theoretic security. Notably, the setup of the symmetric-key scheme in [BI20], where the receiver initially lacks the private key, aligns naturally with the Σ -protocol framework, as the verifier (receiver) does not possess the decryption key for the prover's (sender's) initial message.

Lastly, certified everlasting zero-knowledge and certified everlasting hiding appear unattainable in the classical world, as malicious adversaries can always duplicate their information. In fact, certified everlasting zero-knowledge against classical verifiers would imply honest-verifier statistical zero-knowledge, given that an honest verifier operates within polynomial time.² Moreover, it is known that $\mathbf{HVSZK} = \mathbf{SZK}$, where **HVSZK** and **SZK** represent the classes of languages with honest-verifier and general statistical zero-knowledge proofs, respectively [GSV98]. Therefore, the existence of certified everlasting zero-knowledge proofs for **NP** with classical verification would imply $\mathbf{NP} \subseteq \mathbf{HVSZK} = \mathbf{SZK}$, collapsing the polynomial-time hierarchy [For87]. While this argument pertains to protocols in the standard model, no honest-verifier statistical zero-knowledge proofs for **NP** have been constructed in the random oracle model either. Thus, our results contribute to the expanding repertoire of quantum cryptographic primitives that are feasible only in the quantum domain.

²This reasoning does not extend to quantum verifiers, as honest-verifier quantum statistical zero-knowledge [Wat02] requires the simulator to reproduce the verifier's internal state at *any* point during the protocol execution. In contrast, certified everlasting zero-knowledge ensures security only after a valid deletion certificate is issued.

3.1.3 Technical Overview

Certified Everlasting Zero-Knowledge. As discussed in Section 3.1.1, achieving everlasting zero-knowledge proofs for NP and QMA appears infeasible, even when quantum verifiers are allowed. To overcome this limitation, we introduce *certified everlasting zero-knowledge*, a relaxed notion inspired by the framework of quantum encryption with certified deletion by Broadbent and Islam [BI20]. Certified everlasting zero-knowledge guarantees that the proof remains secure against polynomial-time malicious verifiers that issue a valid certificate proving the deletion of sensitive information. (The formal definition is provided in Section 3.3.1.) In contrast to traditional everlasting zero-knowledge, this notion does not ensure security against verifiers who fail to provide such a certificate. Nevertheless, the concept is meaningful, as the prover can impose penalties on verifiers that refuse to comply.

Quantum Commitment with Certified Everlasting Hiding. Our construction of certified everlasting zero-knowledge proofs builds on the ideas outlined by Broadbent and Islam [BI20], which suggest using a commitment scheme with certified deletion in a Σ -protocol. While their work did not provide a concrete construction or formal definition of such commitments, a straightforward approach might consider using their encryption with certified deletion as a commitment scheme. Specifically, one could treat a ciphertext ct as a commitment. However, this fails to satisfy the binding property since different secret keys can decrypt the same ciphertext into different messages.

One potential solution combines encryption with a classical commitment to the secret key. Let Enc denote the encryption algorithm from [BI20], sk the secret key, and Commit a classical commitment scheme with statistical binding and computational hiding. We can define the commitment as:

$$(ct = \text{Enc}(sk, m), com = \text{Commit}(sk)),$$

where m is the committed message. This resolves the binding issue because $\text{Commit}(sk)$ ensures that sk is uniquely determined. However, this construction does not achieve the required *certified everlasting hiding*, which ensures that the hiding property persists after deletion certificates are issued, even against computationally unbounded receivers. Specifically, the security of certified deletion for Enc relies on sk being information-theoretically hidden prior to deletion. In this construction, sk is statistically determined by $\text{Commit}(sk)$, making it impossible to "delete" sk during security reductions.

To address this issue, we consider the following modified construction:

$$(ct = \text{Enc}(sk, m), com = \text{Commit}(R), H(R) \oplus sk),$$

where R is a random string, H is a hash function modeled as a random oracle, and Commit is a classical commitment scheme. In this approach, $H(R)$ masks sk , and the computational hiding property of Commit ensures that R cannot be efficiently extracted before deletion. Therefore, sk remains hidden, enabling us to reduce the certified everlasting hiding property to the certified deletion security of Enc . This argument is formalized using the one-way-to-hiding lemma [Unr15, AHU19].

Quantum commitments generally lack classical binding properties, as a malicious sender could create a superposition of commitments to multiple messages. To address this, we introduce *classical-extractor-based binding*, which guarantees that an unbounded-time extractor can determine the committed message before decommitment. This property is critical for ensuring soundness in our certified everlasting zero-knowledge proofs. Detailed proofs are provided in Section 3.2.2.

Certified Everlasting Zero-Knowledge Proof for QMA. Once we have a commitment scheme with certified everlasting hiding, constructing certified everlasting zero-knowledge proofs follows naturally from the approach outlined in [BI20]. Although they focused on NP, we observe that their idea extends naturally to QMA because the "quantum version" of a Σ -protocol for QMA, called the Ξ -protocol was introduced by Broadbent and Grilo [BG20]. We sketch the construction below for clarity. Let $A = (A_{\text{yes}}, A_{\text{no}})$ be a promise problem in QMA. Broadbent and Grilo [BG20] showed that for any $x \in A_{\text{yes}}$ and its corresponding witness w , it is possible to generate (in quantum polynomial time) a state known as the local simulatable history state ρ_{hist} from w , which satisfies two key properties (see Definition 2.3.5 for details):

(LS1) Verification can be performed by measuring five randomly chosen qubits of ρ_{hist} .

(LS2) The classical description of any five-qubit reduced density matrix of ρ_{hist} can be computed in classical polynomial time.

With these properties, the quantum Σ -protocol from [BG20] is constructed as follows:

1. **Commitment phase:** The prover randomly selects $x, z \in \{0, 1\}^n$ and sends $(X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes \text{com}(x, z)$ to the verifier, where $X^x Z^z := \prod_{i=1}^n X_i^{x_i} Z_i^{z_i}$, n is the number of qubits in ρ_{hist} , and $\text{com}(x, z)$ is a classical commitment to (x, z) .
2. **Challenge phase:** The verifier randomly chooses a subset $S \subset [n]$ of size $|S| = 5$ and sends it to the prover.
3. **Response phase:** The prover opens the commitment for $\{x_i, z_i\}_{i \in S}$.
4. **Verification phase:** The verifier applies $\prod_{i \in S} X_i^{x_i} Z_i^{z_i}$ to the state and measures the qubits in S .

Correctness and soundness follow from property (LS1), while zero-knowledge is guaranteed by property (LS2). If the classical commitment scheme used in this protocol has statistical binding and computational hiding, the protocol becomes a computational zero-knowledge proof for QMA. However, an unbounded malicious verifier could potentially open the commitment for $\{x_i, z_i\}_{i \in [n] \setminus S}$, thereby obtaining the entire ρ_{hist} , which compromises the zero-knowledge property if more than five qubits of ρ_{hist} are revealed.

To construct a certified everlasting zero-knowledge proof for QMA, we build on the quantum Σ -protocol by using a commitment scheme with certified everlasting hiding and statistical binding for the commitment of (x, z) . In this modified protocol, if the verifier issues a valid deletion certificate for the commitment of $\{x_i, z_i\}_{i \in [n] \setminus S}$, even an unbounded malicious verifier cannot recover $\{x_i, z_i\}_{i \in [n] \setminus S}$, and therefore can only access the five qubits of ρ_{hist} . This ensures the certified everlasting zero-knowledge property. The proof of statistical soundness relies on classical-extractor-based binding, similar to [BG20]. Classical-extractor-based binding allows us to extract the committed message with an unbounded-time extractor before the prover decommits. In particular, we can extract the committed (x, z) from $\text{com}(x, z)$ before the challenge phase. Since extraction occurs before the challenge S is issued, the extracted values do not depend on S , allowing us to reduce the soundness of the scheme to the original QMA promise problem A .

The details of the construction are provided in Section 3.3.2.

3.2 Commitment with Certified Everlasting Hiding and Classical-Extractor-Based Binding

In this section, we define and construct commitment with certified everlasting hiding and statistical binding. We adopt a non-standard syntax for the verification algorithm and a slightly involved definition for the binding,

which we call the classical-extractor-based binding, that are tailored to our construction. This is because they are convenient for our construction of certified everlasting zero-knowledge proof for **QMA** given in Section 3.3.

3.2.1 Definition

Definition 3.2.1 (Commitment with Certified Everlasting Hiding and Classical-Extractor-Based Binding (Syntax)). Let λ be the security parameter and let p, q, r, s and t be some polynomials. Commitment with certified everlasting hiding and classical-extractor-based binding consists of a tuple of polynomial-time algorithms $(\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$ with message space $\mathcal{M} := \{0, 1\}^n$, commitment space $\mathcal{C} := \mathcal{Q}^{\otimes p(\lambda)} \times \{0, 1\}^{q(\lambda)}$, decommitment space $\mathcal{D} := \{0, 1\}^{r(\lambda)}$, key space $\mathcal{K} := \{0, 1\}^{s(\lambda)}$ and deletion certificate space $\mathcal{E} := \{0, 1\}^{t(\lambda)}$.

$\text{Commit}(1^\lambda, m) \rightarrow (com, d, ck)$: The commitment algorithm takes as input a security parameter 1^λ and a message $m \in \mathcal{M}$, and outputs a commitment $com \in \mathcal{C}$, a decommitment $d := (d_1, d_2) \in \mathcal{D}$ and a key $ck \in \mathcal{K}$. Note that com consists of a quantum state $\psi \in \mathcal{Q}^{\otimes p(\lambda)}$ and a classical bit string $f \in \{0, 1\}^{q(\lambda)}$.

$\text{Vrfy}(com, d) \rightarrow m' \text{ or } \perp$: The verification algorithm consists of two algorithms, Vrfy_1 and Vrfy_2 . It parses $d = (d_1, d_2)$. Vrfy_1 takes f and (d_1, d_2) as input, and outputs $\top$ to indicate acceptance or $\perp$. Vrfy_2 takes com and d_1 as input, and outputs m' . If the output of Vrfy_1 is $\perp$, then the output of Vrfy is $\perp$. Otherwise the output of Vrfy is m' .

$\text{Del}(com) \rightarrow \text{cert}$: The deletion algorithm takes com as input, and outputs a certificate $\text{cert} \in \mathcal{E}$.

$\text{Cert}(\text{cert}, ck) \rightarrow \top \text{ or } \perp$: The certification algorithm takes cert and ck as input, and outputs $\top$ to indicate acceptance or $\perp$.

Definition 3.2.2 (Correctness). There are two types of correctness, namely, decommitment correctness and deletion correctness.

Decommitment correctness: There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \{0, 1\}^n$,

$$\Pr \left[m \leftarrow \text{Vrfy}(com, d) \mid (com, d, ck) \leftarrow \text{Commit}(1^\lambda, m) \right] \geq 1 - \text{negl}(\lambda).$$

Deletion correctness: There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \{0, 1\}^n$,

$$\Pr \left[\top \leftarrow \text{Cert}(\text{cert}, ck) \mid (com, d, ck) \leftarrow \text{Commit}(1^\lambda, m), \text{cert} \leftarrow \text{Del}(com) \right] \geq 1 - \text{negl}(\lambda).$$

Definition 3.2.3 (Classical-Extractor-Based Binding). There exists an unbounded-time deterministic algorithm Ext that takes $f \in \{0, 1\}^{q(\lambda)}$ of com as input, and outputs $d_1^* \leftarrow \text{Ext}(f)$ such that for any com , any $d_1 \neq d_1^*$, and any d_2 , $\Pr[\text{Vrfy}(com, d = (d_1, d_2)) = \perp] = 1$.

Definition 3.2.4 (Computational Hiding). Let $\Sigma := (\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$. Let us consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{c-hide}}(\lambda, b)$ against any QPT adversary $\mathcal{A}$.

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

2. The challenger computes $(com, d, ck) \leftarrow \text{Commit}(1^\lambda, m_b)$, and sends com to $\mathcal{A}$.
3. $\mathcal{A}$ outputs $b' \in \{0, 1\}$.
4. The output of the experiment is b' .

Computational hiding means that the following is satisfied for any QPT $\mathcal{A}$.

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

Definition 3.2.5 (Certified Everlasting Hiding). Let $\Sigma := (\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$. Let us consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ever-hide}}(\lambda, b)$ against $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consisting of any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.

1. $\mathcal{A}_1$ generates $(m_0, m_1) \in \mathcal{M}^2$ and sends it to the challenger.
2. The challenger computes $(com, d, ck) \leftarrow \text{Commit}(1^\lambda, m_b)$, and sends com to $\mathcal{A}_1$.
3. At some point, $\mathcal{A}_1$ sends cert to the challenger, and sends its internal state to $\mathcal{A}_2$.
4. The challenger computes $\text{Cert}(\text{cert}, ck)$. If the output is $\top$, then the challenger outputs $\top$, and sends (d, ck) to $\mathcal{A}_2$. Else, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$.
5. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
6. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

We say that it is certified everlasting hiding if the following is satisfied for any $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

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

Remark 3.2.6. We remark that it would be incorrect to formulate a definition that conditions on cert being accepted (see discussion in [Unr15].) We also remark that certified everlasting hiding does not imply computational hiding since it does not require anything if the adversary does not send a valid certificate.

The following lemma will be used in the construction of the certified everlasting zero-knowledge proof for QMA in Section 3.3. It is shown with the standard hybrid argument. It is also easy to see that a similar lemma holds for computational hiding.

Lemma 3.2.7. Let $\Sigma := (\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$ and $\mathcal{M} = \{0, 1\}$. Let us consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda, b)$ against $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ consisting of any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.

1. $\mathcal{A}_1$ generates $(m^0, m^1) \in \{0, 1\}^n \times \{0, 1\}^n$ and sends it to the challenger.
2. The challenger computes

$$(com_i(m_i^b), d_i(m_i^b), ck_i(m_i^b)) \leftarrow \text{Commit}(1^\lambda, m_i^b)$$

for each $i \in [n]$, and sends $\{com_i(m_i^b)\}_{i \in [n]}$ to $\mathcal{A}_1$. Here, m_i^b is the i -th bit of m^b .

3. At some point, $\mathcal{A}_1$ sends $\{\text{cert}_i\}_{i \in [n]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
4. The challenger computes $\text{Cert}(\text{cert}_i, \text{ck}_i(m_i^b))$ for each $i \in [n]$. If the output is $\top$ for all $i \in [n]$, then the challenger outputs $\top$, and sends $\{d_i(m_i^b), \text{ck}_i(m_i^b)\}_{i \in [n]}$ to $\mathcal{A}_2$. Else, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$.
5. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
6. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

If Σ is certified everlasting hiding,

$$\begin{aligned} & \text{Adv}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda) \\ &:= \left| \Pr \left[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda, 0) = 1 \right] - \Pr \left[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda, 1) = 1 \right] \right| \leq \text{negl}(\lambda) \end{aligned}$$

for any $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

Proof of Lemma 3.2.7. Let us consider the following hybrids for $j \in \{0, 1, \dots, n\}$.

Hyb_j:

1. $\mathcal{A}_1$ generates $(m^0, m^1) \in \{0, 1\}^n \times \{0, 1\}^n$ and sends it to the challenger.
2. The challenger computes

$$(\text{com}_i(m_i^1), d_i(m_i^1), \text{ck}_i(m_i^1)) \leftarrow \text{Commit}(1^\lambda, m_i^1)$$

for $i \in [j]$ and

$$(\text{com}_i(m_i^0), d_i(m_i^0), \text{ck}_i(m_i^0)) \leftarrow \text{Commit}(1^\lambda, m_i^0)$$

for each $i \in \{j+1, \dots, n\}$, and sends $\{\text{com}_i(m_i^1)\}_{i \in [j]}$ and $\{\text{com}_i(m_i^0)\}_{i \in \{j+1, \dots, n\}}$ to $\mathcal{A}_1$. Here, m_i^b is the i -th bit of m^b .

3. At some point, $\mathcal{A}_1$ sends $\{\text{cert}_i\}_{i \in [n]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
4. The challenger computes $\text{Cert}(\text{cert}_i, \text{ck}_i(m_i^1))$ for each $i \in [j]$ and $\text{Cert}(\text{cert}_i, \text{ck}_i(m_i^0))$ for each $i \in \{j+1, \dots, n\}$. If the outputs are $\top$ for all $i \in [n]$, then the challenger outputs $\top$, and sends $\{d_i(m_i^1), \text{ck}_i(m_i^1)\}_{i \in [j]}$ and $\{d_i(m_i^0), \text{ck}_i(m_i^0)\}_{i \in \{j+1, \dots, n\}}$ to $\mathcal{A}_2$. Else, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$.
5. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
6. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

It is clear that $\text{Hyb}_0 = \text{Exp}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda, 0)$ and $\text{Hyb}_n = \text{Exp}_{\Sigma, \mathcal{A}}^{\text{bit-ever-hide}}(\lambda, 1)$. Furthermore, we can show

$$\left| \Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1] \right| \leq \text{negl}(\lambda)$$

for each $j \in \{0, 1, \dots, n-1\}$. (Its proof is given below.) From these facts, we obtain Lemma 3.2.7.

Let us show the remaining one. To show it, let us assume that $\left| \Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1] \right|$ is non-negligible. Then, we can construct an adversary $\mathcal{B}$ that can break the certified everlasting hiding of Σ_{ccd} as follows.

1. $\mathcal{B}$ receives (m^0, m^1) from $\mathcal{A}_1$, and computes

$$(com_i(m_i^1), d_i(m_i^1), ck_i(m_i^1)) \leftarrow \text{Commit}(1^\lambda, m_i^1)$$

for $i \in [j]$ and

$$(com_i(m_i^0), d_i(m_i^0), ck_i(m_i^0)) \leftarrow \text{Commit}(1^\lambda, m_i^0)$$

for $i \in \{j+2, \dots, n\}$.

2. $\mathcal{B}$ sends (m_{j+1}^0, m_{j+1}^1) to the challenger of $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{B}}^{\text{ever-hide}}(\lambda, b')$, and receives $com_{j+1}(m_{j+1}^{b'})$ from the challenger.
3. $\mathcal{B}$ sends $\{com_i(m_i^1)\}_{i \in [j]}$, $com_{j+1}(m_{j+1}^{b'})$, and $\{com_i(m_i^0)\}_{i \in \{j+2, \dots, n\}}$, to $\mathcal{A}_1$.
4. $\mathcal{A}_1$ sends $\{\text{cert}_i\}_{i \in [n]}$ to $\mathcal{B}$, and sends its internal state to $\mathcal{A}_2$.
5. $\mathcal{B}$ sends cert_{j+1} to the challenger of $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{B}}^{\text{ever-hide}}(\lambda, b')$, and receives $(d_{j+1}(m_{j+1}^{b'}), ck_{j+1}(m_{j+1}^{b'}))$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.
6. $\mathcal{B}$ sends all d_i and ck_i to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b'' .
8. $\mathcal{B}$ computes Cert for all cert_i , and outputs b'' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[\mathcal{B} \rightarrow 1 \mid b' = 0] = \Pr[\text{Hyb}_j = 1]$ and $\Pr[\mathcal{B} \rightarrow 1 \mid b' = 1] = \Pr[\text{Hyb}_{j+1} = 1]$. By assumption, $|\Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1]|$ is non-negligible, and therefore $|\Pr[\mathcal{B} \rightarrow 1 \mid b' = 0] - \Pr[\mathcal{B} \rightarrow 1 \mid b' = 1]|$ is non-negligible, which contradict the certified everlasting hiding of Σ_{ccd} . $\square$

3.2.2 Construction

Let λ be the security parameter, and let p, q, r, s, t and u be some polynomials. We construct commitment with certified everlasting hiding and classical-extractor-based binding, $\Sigma_{\text{ccd}} = (\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$, with message space $\mathcal{M} = \{0, 1\}^n$, commitment space $\mathcal{C} = \mathcal{Q}^{\otimes p(\lambda)} \times \{0, 1\}^{q(\lambda)} \times \{0, 1\}^{r(\lambda)}$, decommitment space $\mathcal{D} = \{0, 1\}^{s(\lambda)} \times \{0, 1\}^{t(\lambda)}$, key space $\mathcal{K} = \{0, 1\}^{r(\lambda)}$ and deletion certificate space $\mathcal{E} = \{0, 1\}^{u(\lambda)}$ from the following primitives:

- Secret-key encryption with certified deletion, $\Sigma_{\text{skcd}} = \text{SKE}(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$, with plaintext space $\mathcal{M} = \{0, 1\}^n$, ciphertext space $\mathcal{C} = \mathcal{Q}^{\otimes p(\lambda)}$, key space $\mathcal{K} = \{0, 1\}^{r(\lambda)}$, and deletion certificate space $\mathcal{E} = \{0, 1\}^{u(\lambda)}$.
- Classical non-interactive commitment, $\Sigma_{\text{com}} = \text{Classical.Commit}$, with plaintext space $\{0, 1\}^{s(\lambda)}$, randomness space $\{0, 1\}^{t(\lambda)}$, and commitment space $\{0, 1\}^{q(\lambda)}$.
- A hash function H from $\{0, 1\}^{s(\lambda)}$ to $\{0, 1\}^{r(\lambda)}$ modeled as a quantumly-accessible random oracle.

The construction is as follows.

$\text{Commit}(1^\lambda, m)$:

- Generate $\text{ske.sk} \leftarrow \text{SKE.KeyGen}(1^\lambda)$, $R \leftarrow \{0,1\}^{s(\lambda)}$, $R' \leftarrow \{0,1\}^{t(\lambda)}$, and a hash function H from $\{0,1\}^{s(\lambda)}$ to $\{0,1\}^{r(\lambda)}$.
- Compute $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, m)$, $f \leftarrow \text{Classical.Commit}(R; R')$, and $h := H(R) \oplus \text{ske.sk}$.
- Output $\text{com} := (\text{ske.ct}, f, h)$, $d_1 := R$, $d_2 := R'$, and $\text{ck} := \text{ske.sk}$.

$\text{Vrfy}_1(\text{com}, d_1, d_2)$:

- Parse $\text{com} = (\text{ske.ct}, f, h)$, $d_1 = R$, and $d_2 = R'$.
- Output $\top$ if $f = \text{Classical.Commit}(R; R')$, and output $\perp$ otherwise.

$\text{Vrfy}_2(\text{com}, d_1)$:

- Parse $\text{com} = (\text{ske.ct}, f, h)$ and $d_1 = R$.
- Compute $\text{ske.sk}' := H(R) \oplus h$.
- Output $m' \leftarrow \text{SKE.Dec}(\text{ske.sk}', \text{ske.ct})$.

$\text{Del}(\text{com})$:

- Parse $\text{com} = (\text{ske.ct}, f, h)$.
- Compute $\text{ske.cert} \leftarrow \text{SKE.Del}(\text{ske.ct})$.
- Output $\text{cert} := \text{ske.cert}$.

$\text{Cert}(\text{cert}, \text{ck})$:

- Parse $\text{cert} = \text{ske.cert}$ and $\text{ck} = \text{ske.sk}$.
- Output $\top / \perp \leftarrow \text{SKE.Vrfy}(\text{ske.sk}, \text{ske.cert})$.

Correctness. The decommitment and deletion correctness easily follow from the correctness of Σ_{skcd} .

Security. We prove the following three theorems.

Theorem 3.2.8. *If Σ_{com} is perfect binding, then Σ_{ccd} is classical-extractor-based binding.*

Theorem 3.2.9. *If Σ_{com} is unpredictable and Σ_{skcd} is OT-CD secure, then Σ_{ccd} is certified everlasting hiding.*

Theorem 3.2.10. *If Σ_{com} is unpredictable and Σ_{skcd} is OT-CD secure, then Σ_{ccd} is computationally hiding.*

Proof of Theorem 3.2.8. Due to the perfect binding of $\Sigma_{\text{com}} = \text{Classical.Commit}$, there exists a unique d_1^* such that $f = \text{Classical.Commit}(d_1^*; d_2)$ for a given f . Let Ext be the algorithm that finds such d_1^* and outputs it. (If there is no such d_1^* , then Ext outputs $\perp$.) Then, for any $\text{com} = (\text{ske.ct}, f, h)$, any $d_1 \neq d_1^*$, and any d_2 ,

$$\Pr[\text{Vrfy}_1(\text{com}, d_1, d_2) = \top] \geq \Pr[f \neq \text{Classical.Commit}(d_1, d_2)] = 1,$$

which completes the proof. $\square$

Proof of Theorem 3.2.9. For clarity, we describe how the experiment works against an adversary $\mathcal{A} := (\mathcal{A}_1, \mathcal{A}_2)$ consisting of any QPT adversary $\mathcal{A}_1$ and any quantum unbounded time adversary $\mathcal{A}_2$.

$\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{A}}^{\text{ever-hide}}(\lambda, b)$: This is the original experiment.

1. A uniformly random function H from $\{0, 1\}^{s(\lambda)}$ to $\{0, 1\}^{r(\lambda)}$ is chosen. $\mathcal{A}_1$ and $\mathcal{A}_2$ can make arbitrarily many quantum queries to H at any time in the experiment.
2. $\mathcal{A}_1$ chooses $(m_0, m_1) \leftarrow \mathcal{M}^2$, and sends (m_0, m_1) to the challenger.
3. The challenger generates $\text{ske.sk} \leftarrow \text{SKE.KeyGen}(1^\lambda)$, $R \leftarrow \{0, 1\}^{s(\lambda)}$ and $R' \leftarrow \{0, 1\}^{t(\lambda)}$. The challenger computes $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, m_b)$, $f := \text{Classical.Commit}(R; R')$ and $h := H(R) \oplus \text{ske.sk}$, and sends $(\text{ske.ct}, f, h)$ to $\mathcal{A}_1$.
4. $\mathcal{A}_1$ sends ske.cert to the challenger and sends its internal state to $\mathcal{A}_2$.
5. If $\top \leftarrow \text{SKE.Vrfy}(\text{ske.sk}, \text{ske.cert})$, the challenger outputs $\top$ and sends $(R, R', \text{ske.sk})$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$ and sends $\perp$ 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 $\perp$.

What we have to prove is that

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

We define the following sequence of hybrids.

$\text{Hyb}_1(b)$: This is identical to $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{A}}^{\text{ever-hide}}(\lambda, b)$ except that the oracle given to $\mathcal{A}_1$ is replaced with $H_{R \rightarrow H'}$ which is H reprogrammed according to H' on an input R where H' is another independent uniformly random function. More formally, $H_{R \rightarrow H'}$ is defined by

$$H_{R \rightarrow H'}(R^*) := \begin{cases} H(R^*) & (R^* \neq R) \\ H'(R^*) & (R^* = R). \end{cases}$$

We note that the challenger still uses H to generate h , and the oracle which $\mathcal{A}_2$ uses is still H similarly to the original experiment.

$\text{Hyb}_2(b)$: This is identical to $\text{Hyb}_1(b)$ except for the following three points. First, the challenger generates h uniformly at random. Second, the oracle given to $\mathcal{A}_1$ is replaced with H' which is an independent uniformly random function. Third, the oracle given to $\mathcal{A}_2$ is replaced with $H'_{R \rightarrow h \oplus \text{ske.sk}}$ which is H' reprogrammed to $h \oplus \text{ske.sk}$ on an input R . More formally, $H'_{R \rightarrow h \oplus \text{ske.sk}}$ is defined by

$$H'_{R \rightarrow h \oplus \text{ske.sk}}(R^*) := \begin{cases} H'(R^*) & (R^* \neq R) \\ h \oplus \text{ske.sk} & (R^* = R). \end{cases}$$

Proposition 3.2.11. *If Σ_{com} is unpredictable, then*

$$\left| \Pr \left[\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{A}}^{\text{ever-hide}}(\lambda, b) = 1 \right] - \Pr[\text{Hyb}_1(b) = 1] \right| \leq \text{negl}(\lambda).$$

Proof. We assume that $\left| \Pr \left[\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{A}}^{\text{ever-hide}}(\lambda, b) = 1 \right] - \Pr[\text{Hyb}_1(b) = 1] \right|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the unpredictability of Σ_{com} . For notational simplicity, we denote $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{A}}^{\text{ever-hide}}(\lambda, b)$ by $\text{Hyb}_0(b)$. We consider an algorithm $\tilde{\mathcal{A}}$ that works as follows. $\tilde{\mathcal{A}}$ is given an oracle $\mathcal{O}$, which is either H or $H_{R \rightarrow H'}$, and an input z that consists of R and the whole truth table of H , where $R \leftarrow \{0, 1\}^{s(\lambda)}$, and H and H' are uniformly random functions. $\tilde{\mathcal{A}}$ runs $\text{Hyb}_0(b)$ except that it uses its own oracle $\mathcal{O}$ to simulate $\mathcal{A}_1$'s random oracle queries. On the other hand, $\tilde{\mathcal{A}}$ uses H to simulate h and $\mathcal{A}_2$'s random oracle queries regardless of $\mathcal{O}$, which is possible because the truth table of H is included in the input z . By definition, we have

$$\Pr[\text{Hyb}_0(b) = 1] = \Pr[\tilde{\mathcal{A}}^H(R, H) = 1]$$

and

$$\Pr[\text{Hyb}_1(b) = 1] = \Pr[\tilde{\mathcal{A}}^{H_{R \rightarrow H'}}(R, H) = 1]$$

where H in the input means the truth table of H . We apply the one-way to hiding lemma (Lemma 2.3.2) to the above $\tilde{\mathcal{A}}$. Note that $\tilde{\mathcal{A}}$ is inefficient, but the one-way to hiding lemma is applicable to inefficient algorithms. Then if we let $\tilde{\mathcal{B}}$ be the algorithm that measures uniformly chosen query of $\tilde{\mathcal{A}}$, we have

$$\left| \Pr[\tilde{\mathcal{A}}^H(R, H) = 1] - \Pr[\tilde{\mathcal{A}}^{H_{R \rightarrow H'}}(R, H) = 1] \right| \leq 2q \sqrt{\Pr[\tilde{\mathcal{B}}^{H_{R \rightarrow H'}}(R, H) = R]}.$$

By the assumption, the LHS is non-negligible, and thus $\Pr[\tilde{\mathcal{B}}^{H_{R \rightarrow H'}}(R, H) = R]$ is non-negligible.

Let $\tilde{\mathcal{B}}'$ be the algorithm that is the same as $\tilde{\mathcal{B}}$ except that it does not take the truth table of H as input, and sets h to be uniformly random string instead of setting $h := H(R) \oplus \text{ske.sk}$. Then we have

$$\Pr[\tilde{\mathcal{B}}^{H_{R \rightarrow H'}}(R, H) = R] = \Pr[\tilde{\mathcal{B}}'^{H_{R \rightarrow H'}}(R) = R].$$

The reason is as follows: First, $\tilde{\mathcal{B}}$ uses the truth table of H only for generating $h := H(R) \oplus \text{ske.sk}$, because it halts before $\tilde{\mathcal{B}}$ simulates $\mathcal{A}_2$. Second, the oracle $H_{R \rightarrow H'}$ reveals no information about $H(R)$, and thus h can be independently and uniformly random.

Moreover, for any fixed R , when H and H' are uniformly random, $H_{R \rightarrow H'}$ is also a uniformly random function, and therefore we have

$$\Pr[\tilde{\mathcal{B}}'^{H_{R \rightarrow H'}}(R) = R] = \Pr[\tilde{\mathcal{B}}'^H(R) = R].$$

Since $\Pr[\tilde{\mathcal{B}}^{H_{R \rightarrow H'}}(R, H) = R]$ is non-negligible, $\Pr[\tilde{\mathcal{B}}'^H(R) = R]$ is also non-negligible. Recall that $\tilde{\mathcal{B}}'^H$ is an algorithm that simulates $\text{Hyb}_0(b)$ with the modification that h is set to be uniformly random and measures randomly chosen $\mathcal{A}_1$'s query. Then it is straightforward to construct an adversary $\mathcal{B}$ that breaks the unpredictability of Σ_{com} by using $\tilde{\mathcal{B}}'$. For clarity, let us give the description of $\mathcal{B}$ as follows.

$\mathcal{B}$ is given $\text{Classical.Commit}(R; R')$ from the challenger of $\text{Exp}_{\Sigma_{\text{com}}, \mathcal{B}}^{\text{unpre}}(\lambda)$. $\mathcal{B}$ chooses $i \leftarrow [q]$ and runs $\text{Hyb}_1(b)$ until $\mathcal{A}_1$ makes i -th random oracle query or $\mathcal{A}_1$ sends the internal state to $\mathcal{A}_2$, where $\mathcal{B}$ embeds the problem instance $\text{Classical.Commit}(R; R')$ into those sent to $\mathcal{A}_1$ instead of generating it by itself. $\mathcal{B}$ measures the i -th random oracle query by $\mathcal{A}_1$, and outputs the measurement outcome. Note that $\mathcal{B}$ can efficiently simulate the random oracle H by Zhandry's compressed oracle technique [Zha19]. It is clear that the probability that $\mathcal{B}$ outputs R is exactly $\Pr[\tilde{\mathcal{B}}'^H(R) = R]$, which is non-negligible. This contradicts the unpredictability of Σ_{com} . Therefore $|\Pr[\text{Hyb}_0(b) = 1] - \Pr[\text{Hyb}_1(b) = 1]|$ is negligible. $\square$

Proposition 3.2.12. $\Pr[\text{Hyb}_1(b) = 1] = \Pr[\text{Hyb}_2(b) = 1]$.

Proof. First, let us remind the difference between $\text{Hyb}_1(b)$ and $\text{Hyb}_2(b)$. In $\text{Hyb}_1(b)$, $\mathcal{A}_1$ receives $h = \text{ske.sk} \oplus H(R)$. Moreover, $\mathcal{A}_1$ can access to the random oracle $H_{R \rightarrow H'}$, and $\mathcal{A}_2$ can access to the random oracle H . On the other hand, in $\text{Hyb}_2(b)$, $\mathcal{A}_1$ receives uniformly random h . Moreover, $\mathcal{A}_1$ can access to the random oracle H' instead of $H_{R \rightarrow H'}$, and $\mathcal{A}_2$ can access to the random oracle $H'_{R \rightarrow h \oplus \text{ske.sk}}$ instead of H .

Let $\Pr[(h, H_{R \rightarrow H'}, H) = (r, G, G') \mid \text{Hyb}_1(b)]$ be the probability that the adversary $\mathcal{A}_1$ in $\text{Hyb}_1(b)$ receives a classical bit string r as h , random oracle which $\mathcal{A}_1$ can access to is G , and random oracle which $\mathcal{A}_2$ can access to is G' . Similarly, let us define $\Pr[(h, H', H'_{R \rightarrow h \oplus \text{ske.sk}}) = (r, G, G') \mid \text{Hyb}_2(b)]$ for $\text{Hyb}_2(b)$. What we have to show is that the following equation holds for any (r, G, G')

$$\Pr[(h, H_{R \rightarrow H'}, H) = (r, G, G') \mid \text{Hyb}_1(b)] = \Pr[(h, H', H'_{R \rightarrow h \oplus \text{ske.sk}}) = (r, G, G') \mid \text{Hyb}_2(b)].$$

Since $h = \text{ske.sk} \oplus H(R)$ in $\text{Hyb}_1(b)$, H is a uniformly random function, and h in $\text{Hyb}_2(b)$ is uniformly generated,

$$\Pr[h = r \mid \text{Hyb}_1(b)] = \Pr[h = r \mid \text{Hyb}_2(b)]$$

holds for any r .

For any classical bit string r and any random oracle G , we have

$$\Pr[H_{R \rightarrow H'} = G \mid h = r, \text{Hyb}_1(b)] = \Pr[H' = G \mid h = r, \text{Hyb}_2(b)].$$

This is shown as follows. First, in $\text{Hyb}_1(b)$, from the construction of $H_{R \rightarrow H'}$, $H_{R \rightarrow H'}(R)$ is independent from h for any $R \in \{0, 1\}^{s(\lambda)}$. Furthermore, since H and H' is random oracle, $H_{R \rightarrow H'}(R)$ is uniformly random for any $R \in \{0, 1\}^{s(\lambda)}$. Second, in $\text{Hyb}_2(b)$, from the construction of H' , $H'(R)$ is independent from h for any $R \in \{0, 1\}^{s(\lambda)}$. Furthermore, since H' is random oracle, $H'(R)$ is uniformly random for any $R \in \{0, 1\}^{s(\lambda)}$. Therefore, we have the above equation.

For any classical bit string r and any random oracles G and G' , we have

$$\Pr[H = G' \mid (h, H_{R \rightarrow H'}) = (r, G), \text{Hyb}_1(b)] = \Pr[H'_{R \rightarrow h \oplus \text{ske.sk}} = G' \mid (h, H') = (r, G), \text{Hyb}_2(b)].$$

This can be shown as follows. First, in $\text{Hyb}_1(b)$, we obtain $H(R) = r \oplus \text{ske.sk}$, because $h := \text{ske.sk} \oplus H(R)$ and $h = r$. Furthermore, from the definition of $H_{R \rightarrow H'}$, we obtain $H(R^*) = G(R^*)$ for $R^* \neq R$. Second, in $\text{Hyb}_2(b)$, from the definition of $H'_{R \rightarrow h \oplus \text{ske.sk}}$, we have $H'_{R \rightarrow h \oplus \text{ske.sk}}(R) = r \oplus \text{ske.sk}$ and $H'_{R \rightarrow h \oplus \text{ske.sk}}(R^*) = G(R^*)$ for $R^* \neq R$.

From all the above discussions, we have

$$\Pr[(h, H_{R \rightarrow H'}, H) = (r, G, G') \mid \text{Hyb}_1(b)] = \Pr[(h, H', H'_{R \rightarrow h \oplus \text{ske.sk}}) = (r, G, G') \mid \text{Hyb}_2(b)].$$

□

Proposition 3.2.13. If Σ_{skcd} is OT-CD secure, then

$$|\Pr[\text{Hyb}_2(1) = 1] - \Pr[\text{Hyb}_2(0) = 1]| \leq \text{negl}(\lambda).$$

Proof. To show this, we assume that $|\Pr[\text{Hyb}_2(1) = 1] - \Pr[\text{Hyb}_2(0) = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the OT-CD security of Σ_{skcd} .

$\mathcal{B}$ plays the experiment $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$ for some $b' \in \{0, 1\}$. First, $\mathcal{B}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$. $\mathcal{B}$ receives ske.ct from the challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$. $\mathcal{B}$ generates $R \leftarrow \{0, 1\}^{s(\lambda)}$, $R' \leftarrow \{0, 1\}^{t(\lambda)}$ and $h \leftarrow \{0, 1\}^{r(\lambda)}$, and computes $f := \text{Classical.Commit}(R; R')$. $\mathcal{B}$ sends $(\text{ske.ct}, f, h)$ to $\mathcal{A}_1$. $\mathcal{B}$ simulates the random oracle H' given to $\mathcal{A}_1$ by itself. At some point, $\mathcal{A}_1$ sends ske.cert to $\mathcal{B}$, and sends the internal state to $\mathcal{A}_2$. $\mathcal{B}$ passes ske.cert to the challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$.

The challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$ runs $\text{SKE.Vrfy}(\text{ske.sk}, \text{ske.cert}) \rightarrow \top / \perp$. If it is $\top$, the challenger sends ske.sk to $\mathcal{B}$. In that case, $\mathcal{B}$ outputs $\top$, and sends $(R, R', \text{ske.sk})$ to $\mathcal{A}_2$. We denote this event by $\text{Reveal}_{\text{sk}}(b')$. $\mathcal{B}$ simulates $\mathcal{A}_2$, and outputs the output of $\mathcal{A}_2$. On the other hand, if $\text{SKE.Vrfy}(\text{ske.sk}, \text{ske.cert}) \rightarrow \perp$, then the challenger sends $\perp$ to $\mathcal{B}$. In that case, $\mathcal{B}$ outputs $\perp$ and aborts. Note that $\mathcal{B}$ can simulate the random oracle $H'_{R \rightarrow h \oplus \text{ske.sk}}$ given to $\mathcal{A}_2$ when $\mathcal{B}$ does not abort, because $\mathcal{B}$ receives ske.sk from the challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$ when $\mathcal{B}$ does not abort.

Now we have

$$\begin{aligned}
& \text{Adv}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda) \\
&:= \left| \Pr \left[\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b') = 1 \mid b' = 0 \right] - \Pr \left[\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b') = 1 \mid b' = 1 \right] \right| \\
&= \left| \Pr [\mathcal{B} = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 0] - \Pr [\mathcal{B} = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 1] \right| \\
&= \left| \Pr [\mathcal{A}_2 = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 0] - \Pr [\mathcal{A}_2 = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 1] \right| \\
&= \left| \Pr [\text{Hyb}_2(b') = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 0] - \Pr [\text{Hyb}_2(b') = 1 \wedge \text{Reveal}_{\text{sk}}(b') \mid b' = 1] \right| \\
&= \left| \Pr [\text{Hyb}_2(b') = 1 \mid b' = 0] - \Pr [\text{Hyb}_2(b') = 1 \mid b' = 1] \right| \\
&= \left| \Pr [\text{Hyb}_2(0) = 1] - \Pr [\text{Hyb}_2(1) = 1] \right|.
\end{aligned}$$

In the second equation, we have used the fact that $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b) = 1$ if and only if $\mathcal{B} = 1$ and the challenger of $\text{Exp}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda, b')$ outputs $\top$. In the third equation, we have used the fact that the output of $\mathcal{B}$ is equal to the output of $\mathcal{A}_2$ conditioned that $\text{Reveal}_{\text{sk}}(b')$ occurs. In the fourth equation, we have used the fact that $\mathcal{B}$ simulates the challenger of $\text{Hyb}_2(b)$ when $\text{Reveal}_{\text{sk}}(b)$ occurs. In the fifth equation, we have used the fact that $\text{Hyb}_2(b) = 1$ only when $\text{Reveal}_{\text{sk}}(b)$ occurs. Since $|\Pr[\text{Hyb}_2(0) = 1] - \Pr[\text{Hyb}_2(1) = 1]|$ is non-negligible, $\text{Adv}_{\Sigma_{\text{skcd}}, \mathcal{B}}^{\text{otsk-cert-del}}(\lambda)$ is non-negligible. This contradicts the OT-CD security of Σ_{skcd} . $\square$

By Propositions 3.2.11 to 3.2.13, we immediately obtain Theorem 3.2.9. $\square$

3.3 Certified Everlasting Zero-Knowledge Proof for QMA

In this section, we define and construct the certified everlasting zero-knowledge proof for **QMA**. In Section 3.3.1, we define the certified everlasting zero-knowledge proof for **QMA**. We then construct a three round protocol with completeness-soundness gap $\frac{1}{\text{poly}(\lambda)}$ in Section 3.3.2, and finally amplify the gap to $1 - \text{negl}(\lambda)$ with the sequential repetition in Section 3.3.3.

3.3.1 Definition

We first define a quantum interactive protocol. Usually, in zero-knowledge proofs or arguments, we do not consider prover's output. However, in this paper, we also consider prover's output, because we are interested in the certified everlasting zero-knowledge. Furthermore, in this paper, we consider only an interactive proof, which means that a malicious prover is unbounded.

Definition 3.3.1 (Quantum Interactive Protocol). A quantum interactive protocol is modeled as an interaction between QPT machines $\mathcal{P}$ referred as a prover and $\mathcal{V}$ referred as a verifier. We denote by $\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x)$ an execution of the protocol where x is a common input, x_P is $\mathcal{P}$'s private input, and x_V is $\mathcal{V}$'s private input. We denote by $\text{OUT}_{\mathcal{V}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x)$ the final output of $\mathcal{V}$ in the execution. An honest verifier's output is $\top$ indicating acceptance or $\perp$ indicating rejection, and a malicious verifier's output is an arbitrary quantum state. We denote by $\text{OUT}_{\mathcal{P}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x)$ the final output of $\mathcal{P}$ in the execution. An honest prover's output is $\top$ indicating acceptance or $\perp$ indicating rejection. We also define $\text{OUT}'_{\mathcal{P}, \mathcal{V}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x)$ by

$$\begin{aligned} & \text{OUT}'_{\mathcal{P}, \mathcal{V}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x) \\ & := \begin{cases} (\top, \text{OUT}_{\mathcal{V}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x)) & (\text{OUT}_{\mathcal{P}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x) = \top) \\ (\perp, \perp) & (\text{OUT}_{\mathcal{P}}\langle \mathcal{P}(x_P), \mathcal{V}(x_V) \rangle(x) \neq \top). \end{cases} \end{aligned}$$

We next define a computational zero-knowledge proof for **QMA**, which is the standard definition.

Definition 3.3.2 (Computational Zero-Knowledge Proof for QMA). A c -complete s -sound computational zero-knowledge proof for a **QMA** promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is a quantum interactive protocol between a QPT prover $\mathcal{P}$ and a QPT verifier $\mathcal{V}$ that satisfies the followings:

c -completeness: For any $x \in A_{\text{yes}}$ and any $w \in R_A(x)$,

$$\Pr[\text{Out}_{\mathcal{V}}\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V} \rangle(x) = \top] \geq c$$

for some polynomial k .

s -soundness: For any $x \in A_{\text{no}}$ and any unbounded-time prover $\mathcal{P}^*$,

$$\Pr[\text{Out}_{\mathcal{V}}\langle \mathcal{P}^*, \mathcal{V} \rangle(x) = \top] \leq s.$$

Computational zero-knowledge: There exists a QPT algorithm $\mathcal{S}$ such that

$$\text{OUT}_{\mathcal{V}^*}\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x) \approx_c \mathcal{S}(x, \mathcal{V}^*, \cdot)$$

for any QPT malicious verifier $\mathcal{V}^*$, any $x \in A_{\text{yes}} \cap \{0, 1\}^\lambda$, any $w \in R_A(x)$, and some polynomial k . Note that $\text{OUT}_{\mathcal{V}^*}\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x)$ and $\mathcal{S}(x, \mathcal{V}^*, \cdot)$ are quantum channels that map any quantum state ξ to quantum states $\text{OUT}_{\mathcal{V}^*}\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$ and $\mathcal{S}(x, \mathcal{V}^*, \xi)$, respectively.

We just call it a computational zero-knowledge proof if it satisfies $(1 - \text{negl}(|x|))$ -completeness, $\text{negl}(|x|)$ -soundness, and computational zero-knowledge.

We finally define a certified everlasting zero-knowledge proof for **QMA**, which is the main target of this paper.

Definition 3.3.3 (Certified Everlasting Zero-Knowledge Proof for QMA). A certified everlasting zero-knowledge proof for a **QMA** promise problem $A = (A_{\text{yes}}, A_{\text{no}})$ is a computational zero-knowledge proof for A (Definition 3.3.2) that additionally satisfies the followings:

Prover's completeness: $\Pr[\text{OUT}_{\mathcal{P}}\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V} \rangle(x) = \top] \geq 1 - \text{negl}(\lambda)$ for any $x \in A_{\text{yes}} \cap \{0, 1\}^\lambda$ and any $w \in R_A(x)$.

Certified everlasting zero-knowledge: *There exists a QPT algorithm $\mathcal{S}$ such that*

$$\text{OUT}'_{\mathcal{P}, \mathcal{V}^*} \langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x) \approx_s \mathcal{S}(x, \mathcal{V}^*, \cdot)$$

for any QPT malicious verifier $\mathcal{V}^$, any $x \in A_{\text{yes}} \cap \{0, 1\}^\lambda$, any $w \in R_A(x)$, and some polynomial k . Note that $\text{OUT}'_{\mathcal{P}, \mathcal{V}^*} \langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x)$ and $\mathcal{S}(x, \mathcal{V}^*, \cdot)$ are quantum channels that map any quantum state ξ to quantum states $\text{OUT}'_{\mathcal{P}, \mathcal{V}^*} \langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$ and $\mathcal{S}(x, \mathcal{V}^*, \xi)$, respectively.*

Remark 3.3.4. We remark that certified everlasting zero-knowledge does not imply computational zero-knowledge since it does not require anything if the prover does not output $\top$.

3.3.2 Construction of Three Round Protocol

In this section, we construct a three round protocol with completeness-soundness gap $\frac{1}{\text{poly}(\lambda)}$. In the next section, we will amplify its completeness-soundness gap by the sequential repetition.

In the following, $n, m, \Pi_c, \rho_{\text{hist}}$, and $\rho_{\text{Sim}}^{x, S}$ are given in Definition 2.3.5. Let $S_c \subseteq [n]$ be the set of qubits on which Π_c acts non-trivially. The three round protocol Σ_{ecd} is constructed from commitment with certified everlasting hiding and classical-extractor-based binding, $\Sigma_{\text{ccd}} = (\text{Commit}, \text{Vrfy}, \text{Del}, \text{Cert})$.

The first action by the prover (commitment phase):

- Generate $x, z \leftarrow \{0, 1\}^n$.
- Compute

$$\begin{aligned} (\text{com}_i(x_i), d_i(x_i), \text{ck}_i(x_i)) &\leftarrow \text{Commit}(1^\lambda, x_i) \\ (\text{com}_i(z_i), d_i(z_i), \text{ck}_i(z_i)) &\leftarrow \text{Commit}(1^\lambda, z_i) \end{aligned}$$

for all $i \in [n]$.

- Generate a simulatable witness ρ_{hist} for the instance x and generate $X^x Z^z \rho_{\text{hist}} Z^z X^x$.
- Send the first message (commitment), $m_1 := (X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes \text{com}(x) \otimes \text{com}(z)$, to the verifier, where $\text{com}(x) := \bigotimes_{i=1}^n \text{com}_i(x_i)$ and $\text{com}(z) := \bigotimes_{i=1}^n \text{com}_i(z_i)$.

The second action by the verifier (challenge phase):

- Generate $c \leftarrow [m]$.
- Compute $\text{cert}_i(x_i) \leftarrow \text{Del}(\text{com}_i(x_i))$ and $\text{cert}_i(z_i) \leftarrow \text{Del}(\text{com}_i(z_i))$ for all $i \in \bar{S}_c$.
- Send the second message (challenge), $m_2 := (c, \{\text{cert}_i(x_i), \text{cert}_i(z_i)\}_{i \in \bar{S}_c})$, to the prover.

The third action by the prover (reply phase):

- Send the third message (reply), $m_3 := \{d_i(x_i), d_i(z_i)\}_{i \in S_c}$, to the verifier.
- Output $\top$ if $\top \leftarrow \text{Cert}(\text{cert}_i(x_i), \text{ck}_i(x_i))$ and $\top \leftarrow \text{Cert}(\text{cert}_i(z_i), \text{ck}_i(z_i))$ for all $i \in \bar{S}_c$, and output $\perp$ otherwise.

The fourth action by the verifier (verification phase):

- Compute $x'_i \leftarrow \text{Vrfy}(\text{com}_i(x_i), d_i(x_i))$ and $z'_i \leftarrow \text{Vrfy}(\text{com}_i(z_i), d_i(z_i))$ for all $i \in S_c$. If $x'_i = \perp$ or $z'_i = \perp$ for at least one $i \in S_c$, output $\perp$ and abort.

- Apply $X_i^{x'_i} Z_i^{z'_i}$ on the i -th qubit of $X^x Z^z \rho_{\text{hist}} Z^z X^x$ for each $i \in S_c$, and perform the POVM measurement $\{\Pi_c, I - \Pi_c\}$ on the state.
- Output $\top$ if the result Π_c is obtained, and output $\perp$ otherwise.

Theorem 3.3.5. $\Sigma_{\Xi\text{cd}}$ is a certified everlasting zero-knowledge proof for **QMA** with $(1 - \text{negl}(\lambda))$ -completeness and $(1 - \frac{1}{\text{poly}(\lambda)})$ -soundness.

This is shown from the following Lemmata 3.3.6 to 3.3.9.

Lemma 3.3.6. $\Sigma_{\Xi\text{cd}}$ satisfies the $(1 - \text{negl}(\lambda))$ -completeness and prover's completeness.

Lemma 3.3.7. If Σ_{ccd} is classical-extractor-based binding, then $\Sigma_{\Xi\text{cd}}$ satisfies $(1 - \frac{1}{\text{poly}(\lambda)})$ -soundness.

Lemma 3.3.8. If Σ_{ccd} is certified everlasting hiding and computational hiding, then $\Sigma_{\Xi\text{cd}}$ satisfies certified everlasting zero-knowledge.

Lemma 3.3.9. If Σ_{ccd} is computational hiding, then $\Sigma_{\Xi\text{cd}}$ satisfies computational zero-knowledge.

Proof of Lemma 3.3.6. It is clear from the definition of k -**SimQMA** (Definition 2.3.5) and the correctness of Σ_{ccd} . $\square$

Proof of Lemma 3.3.7. Let us show the soundness by analyzing the case for $x \in A_{\text{no}}$. The prover sends the first message to the verifier. The first message consists of three registers, RS , RCX , and RCZ . The register RCX further consists of n registers $\{RCX_i\}_{i \in [n]}$. The register RCZ also consists of n registers $\{RCZ_i\}_{i \in [n]}$. If the prover is honest, RS contains $X^x Z^z \rho_{\text{hist}} Z^z X^x$, RCX_i contains $\text{com}_i(x_i)$, and RCZ_i contains $\text{com}_i(z_i)$. Let $\text{com}'_{i,x}$ and $\text{com}'_{i,z}$ be the (reduced) states of the registers RCX_i and RCZ_i , respectively. Let $f'_{i,x}$ and $f'_{i,z}$ be classical parts of $\text{com}'_{i,x}$ and $\text{com}'_{i,z}$, respectively.

The verifier generates $c \leftarrow [m]$, and issues the deletion certificate. The verifier sends c and the deletion certificate to the prover. The verifier then receives $\{d_1^{x,i}, d_2^{x,i}, d_1^{z,i}, d_2^{z,i}\}_{i \in S_c}$ from the prover. For each $i \in [n]$, let us define $d_1^{*,x,i}$ and $d_1^{*,z,i}$ by $d_1^{*,x,i} \leftarrow \text{Ext}(f'_{i,x})$ and $d_1^{*,z,i} \leftarrow \text{Ext}(f'_{i,z})$, respectively. Note that each $d_1^{*,x,i}$ and $d_1^{*,z,i}$ is independent of c , because $\text{com}'_{i,x}$ and $\text{com}'_{i,z}$ are sent to the verifier before the verifier chooses c .

We have only to consider the case when $d_1^{x,i} = d_1^{*,x,i}$ and $d_1^{z,i} = d_1^{*,z,i}$ for all $i \in S_c$, because of the following reason: Due to the classical-extractor-based binding of Σ_{ccd} , $\text{Vrfy}(\text{com}'_{i,x}, (d_1^{x,i}, d_2^{x,i})) = \perp$ for any $d_1^{x,i} \neq d_1^{*,x,i}$ and any $d_2^{x,i}$. Similarly, $\text{Vrfy}(\text{com}'_{i,z}, (d_1^{z,i}, d_2^{z,i})) = \perp$ for any $d_1^{z,i} \neq d_1^{*,z,i}$ and any $d_2^{z,i}$. Therefore, the prover who wants to make the verifier accept has to send $d_1^{x,i} = d_1^{*,x,i}$ and $d_1^{z,i} = d_1^{*,z,i}$ for all $i \in S_c$.

Let us define

$$p(x, z) := \Pr \left[\bigwedge_{i \in [n]} \left(\text{Vrfy}_2(\text{com}'_{i,x}, d_1^{*,x,i}) \rightarrow x_i \wedge \text{Vrfy}_2(\text{com}'_{i,z}, d_1^{*,z,i}) \rightarrow z_i \right) \right].$$

Note that $p(x, z)$ is independent of c , because $\{\text{com}'_{i,x}, \text{com}'_{i,z}\}_{i \in [n]}$ and $\{d_1^{*,x,i}, d_1^{*,z,i}\}_{i \in [n]}$ are independent of

c. Let ψ be the (reduced) state of the register RS . The verifier's acceptance probability is

$$\begin{aligned}
& \frac{1}{m} \sum_{c \in [m]} \sum_{x, z \in \{0,1\}^n} p(x, z) \text{Tr} \left[\Pi_c \left(\prod_{i \in S_c} Z_i^{z_i} X_i^{x_i} \right) \psi \left(\prod_{i \in S_c} X_i^{x_i} Z_i^{z_i} \right) \right] \\
&= \frac{1}{m} \sum_{c \in [m]} \sum_{x, z \in \{0,1\}^n} p(x, z) \text{Tr} \left[\Pi_c \left(\prod_{i \in [n]} Z_i^{z_i} X_i^{x_i} \right) \psi \left(\prod_{i \in [n]} X_i^{x_i} Z_i^{z_i} \right) \right] \\
&= \frac{1}{m} \sum_{c \in [m]} \text{Tr} \left[\Pi_c \sum_{x, z \in \{0,1\}^n} p(x, z) \left(\prod_{i \in [n]} Z_i^{z_i} X_i^{x_i} \right) \psi \left(\prod_{i \in [n]} X_i^{x_i} Z_i^{z_i} \right) \right] \\
&\leq 1 - \frac{1}{\text{poly}(\lambda)},
\end{aligned}$$

where the last inequality comes from Definition 2.3.5. This completes the proof. $\square$

Proof of Lemma 3.3.8. Let us show certified everlasting zero-knowledge. For a subset $S_c \subseteq [n]$ and $x, z \in \{0,1\}^n$, let us define $x^{S_c} := (x_1^{S_c}, x_2^{S_c}, \dots, x_n^{S_c})$ and $z^{S_c} := (z_1^{S_c}, z_2^{S_c}, \dots, z_n^{S_c})$, where $x_i^{S_c} = x_i$ and $z_i^{S_c} = z_i$ for $i \in S_c$, and $x_i^{S_c} = z_i^{S_c} = 0$ for $i \notin S_c$.

For clarity, we describe how the interactive algorithm $\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$ runs against a QPT verifier $\mathcal{V}^*$ with an input ξ , where w is the witness and x is the instance.

$\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$:

1. $\mathcal{P}$ generates $x, z \leftarrow \{0,1\}^n$, and computes

$$\begin{aligned}
(\text{com}_i(x_i), d_i(x_i), \text{ck}_i(x_i)) &\leftarrow \text{Commit}(1^\lambda, x_i) \\
(\text{com}_i(z_i), d_i(z_i), \text{ck}_i(z_i)) &\leftarrow \text{Commit}(1^\lambda, z_i)
\end{aligned}$$

for all $i \in [n]$. $\mathcal{P}$ sends $m_1 := (X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes \text{com}(x) \otimes \text{com}(z)$ to $\mathcal{V}^*$.

2. $\mathcal{V}^*$ appends ξ to the received state, and runs a QPT circuit V_1^* on it to obtain $(c, \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$. $\mathcal{V}^*$ sends $m_2 := (c, \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$ to $\mathcal{P}$.
3. $\mathcal{P}$ sends $m_3 := \{d_i(x_i), d_i(z_i)\}_{i \in S_c}$ to $\mathcal{V}^*$.
4. $\mathcal{V}^*$ appends m_3 to its state, and runs a QPT circuit V_2^* on it. $\mathcal{V}^*$ outputs its state ξ' .
5. $\mathcal{P}$ computes $\text{Cert}(\text{cert}'_{i,x}, \text{ck}_i(x_i))$ and $\text{Cert}(\text{cert}'_{i,z}, \text{ck}_i(z_i))$ for all $i \in \bar{S}_c$. If all outputs are $\top$, then $\mathcal{P}$ outputs $\top$. Otherwise, $\mathcal{P}$ outputs $\perp$.

Next, let us define a simulator $\mathcal{S}^{(1)}$ as follows.

The simulator $\mathcal{S}^{(1)}(x, \mathcal{V}^*, \xi)$:

1. Pick $c \leftarrow [m]$ and $x, z \leftarrow \{0,1\}^n$. Compute

$$\begin{aligned}
(\text{com}_i(x_i^{S_c}), d_i(x_i^{S_c}), \text{ck}_i(x_i^{S_c})) &\leftarrow \text{Commit}(1^\lambda, x_i^{S_c}) \\
(\text{com}_i(z_i^{S_c}), d_i(z_i^{S_c}), \text{ck}_i(z_i^{S_c})) &\leftarrow \text{Commit}(1^\lambda, z_i^{S_c})
\end{aligned}$$

for all $i \in [n]$.

2. Generate $(X^x Z^z \sigma(c) Z^z X^x) \otimes \text{com}(x^{S_c}) \otimes \text{com}(z^{S_c}) \otimes \xi$, where $\sigma(c) := \rho_{\text{sim}}^{x, S_c} \otimes \left(\prod_{i \in \bar{S}_c} |0\rangle\langle 0|_i \right)$. Run V_1^* on the state to obtain $(c', \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$.
3. If $c' \neq c$, abort and output a fixed state η and the flag state fail.
4. Append $\{d_i(x_i^{S_c}), d_i(z_i^{S_c})\}_{i \in S_c}$ to its quantum state, and run V_2^* on the state to obtain ξ' .
5. Compute $\text{Cert}(\text{cert}'_{i,x}, \text{ck}_i(x_i^{S_c}))$ and $\text{Cert}(\text{cert}'_{i,z}, \text{ck}_i(z_i^{S_c}))$ for all $i \in \bar{S}_c$. If all outputs are $\top$, then output the state $(\top, \xi')$. Otherwise, output $(\perp, \perp)$. Also output the flag state success.

Let us also define other two simulators, $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$, as follows.

The simulator $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \xi)$: It is the same as $\mathcal{S}^{(1)}$ except that $\sigma(c)$ is replaced with ρ_{hist} .

The simulator $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \xi)$: $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ is the channel that postselects the output of $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ on the non-aborting state. More precisely, if we write $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \rho_{\text{in}}) = p\rho_{\text{out}} \otimes \text{success} + (1-p)\eta \otimes \text{fail}$, where p is the non-aborting probability, $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \rho_{\text{in}}) = \rho_{\text{out}}$.

Lemma 3.3.8 is shown from the following Propositions 3.3.10 to 3.3.12 (whose proofs will be given later) and quantum rewinding lemma (Lemma 2.3.1), which is used to reduce the probability that $\mathcal{S}^{(1)}$ aborts to $\text{negl}(\lambda)$. In fact, from Proposition 3.3.10 and Lemma 2.3.1, there exists a quantum circuit $\mathcal{S}^{(0)}$ of size at most $O(m \text{poly}(n) \text{size}(\mathcal{S}^{(1)}))$ such that the probability that $\mathcal{S}^{(0)}$ aborts is $\text{negl}(\lambda)$, and the output quantum states of $\mathcal{S}^{(0)}$ and $\mathcal{S}^{(1)}$ are $\text{negl}(\lambda)$ -close when they do not abort. From Propositions 3.3.11 and 3.3.12, $\mathcal{S}^{(0)}$ is $\text{negl}(\lambda)$ -close to the real protocol, which completes the proof. $\square$

Proposition 3.3.10. *If Σ_{ccd} is computationally hiding, then the probability that $\mathcal{S}^{(1)}$ does not abort is $\frac{1}{m} \pm \text{negl}(\lambda)$.*

Proposition 3.3.11. *$\mathcal{S}^{(1)}(x, \mathcal{V}^*, \cdot) \approx_s \mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ for any $x \in A_{\text{yes}} \cap \{0,1\}^\lambda$ and any $w \in R_A(x)$.*

Proposition 3.3.12. *If Σ_{ccd} is certified everlasting hiding, $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot) \approx_s \text{OUT}'_{\mathcal{P}, \mathcal{V}^*}(\mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot))(x)$.*

Proof of Proposition 3.3.10. We prove the proposition by contradiction. Let p be the probability that $\mathcal{S}^{(1)}$ does not abort. Assume that the probability p satisfies $|p - \frac{1}{m}| \geq \frac{1}{q(\lambda)}$ for a polynomial q . Then, we can construct an adversary $\mathcal{B}$ that breaks the computational hiding of Σ_{ccd} . Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ generates $c \leftarrow [m]$ and $x, z \leftarrow \{0,1\}^n$.
2. $\mathcal{B}$ sends $m_0 := \{x_i, z_i\}_{i \in S_c}$ and $m_1 := 0^{10}$ to the challenger. $\mathcal{B}$ receives commitments from the challenger which is either $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in S_c}$ or $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in S_c}$.
3. $\mathcal{B}$ generates $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in \bar{S}_c}$.
4. $\mathcal{B}$ generates $X^x Z^z \sigma(c) Z^z X^x$. $\mathcal{B}$ appends commitments and ξ to the quantum state in the ascending order. If the commitments for $i \in S_c$ are $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in S_c}$, $\mathcal{B}$ obtains $X^x Z^z \sigma(c) Z^z X^x \otimes \text{com}(x^{S_c}) \otimes \text{com}(z^{S_c}) \otimes \xi$. If the commitments for $i \in S_c$ are $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in S_c}$, $\mathcal{B}$ obtains $X^x Z^z \sigma(c) Z^z X^x \otimes \text{com}(0^n) \otimes \text{com}(0^n) \otimes \xi$.

5. $\mathcal{B}$ runs V_1^* on it to obtain $(c', \{\text{cert}'_{i,x'}, \text{cert}'_{i,z}\}_{i \in \bar{S}_{c'}})$. $\mathcal{B}$ outputs 0 when $c \neq c'$. $\mathcal{B}$ outputs 1 when $c = c'$.

When $\mathcal{B}$ receives $\{com_i(x_i), com_i(z_i)\}_{i \in S_c}$ from the challenger, it outputs 1 with probability p since it simulates $\mathcal{S}^{(1)}$. When $\mathcal{B}$ receives $\{com_i(0), com_i(0)\}_{i \in S_c}$ from the challenger, on the other hand, it outputs 1 with probability $\frac{1}{m}$, because $(X^x Z^z \sigma(c) Z^z X^x) \otimes com(0^n) \otimes com(0^n) \otimes \xi$ is independent of c . (Note that $\sigma(c)$ is one-time padded by x, z .) Therefore if there exists some polynomial q such that $|p - \frac{1}{m}| \geq \frac{1}{q(\lambda)}$, $\mathcal{B}$ can break the computational hiding of Σ_{ccd} from (the computational hiding version of) Lemma 3.2.7. $\square$

Proof of Proposition 3.3.11. It is clear from the local simulatability (Definition 2.3.5) and the definition of x^{S_c} and z^{S_c} (all $x_i^{S_c}$ and $z_i^{S_c}$ are 0 except for those in $i \in S_c$). $\square$

Proof of Proposition 3.3.12. We prove the proposition by contradiction. We construct an adversary $\mathcal{B}$ that breaks the security of the certified everlasting hiding of Σ_{ccd} by assuming the existence of a distinguisher $\mathcal{D}$ that distinguishes two states δ_0 and δ_1 ,

$$\begin{aligned}\delta_0 &:= (\text{OUT}'_{\mathcal{P}, \mathcal{V}^*} \langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle (x) \otimes I) \sigma \\ \delta_1 &:= (\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot) \otimes I) \sigma,\end{aligned}$$

with a certain state σ . Let us describe how $\mathcal{B}$ works.

1. $\mathcal{B}$ generates $c \leftarrow [m]$ and $x, z \leftarrow \{0, 1\}^n$.
2. $\mathcal{B}$ sends $m_0 := \{x_i, z_i\}_{i \in \bar{S}_c}$ and $m_1 := 0^{2n-10}$ to the challenger of $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{B}}^{\text{bit-ever hide}}(\lambda, b)$. $\mathcal{B}$ receives commitments from the challenger which is either $\{com_i(x_i), com_i(z_i)\}_{i \in \bar{S}_c}$ or $\{com_i(0), com_i(0)\}_{i \in \bar{S}_c}$.
3. $\mathcal{B}$ computes

$$\begin{aligned}(com_i(x_i), d_i(x_i), ck_i(x_i)) &\leftarrow \text{Commit}(1^\lambda, x_i) \\ (com_i(z_i), d_i(z_i), ck_i(z_i)) &\leftarrow \text{Commit}(1^\lambda, z_i)\end{aligned}$$

for $i \in S_c$ by itself.

4. $\mathcal{B}$ generates $X^x Z^z \rho_{\text{hist}} Z^z X^x$. $\mathcal{B}$ appends commitments and σ to the quantum state. If the commitments for $i \in \bar{S}_c$ are $\{com_i(x_i), com_i(z_i)\}_{i \in \bar{S}_c}$, $\mathcal{B}$ obtains $(X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes com(x) \otimes com(z) \otimes \sigma$. If the commitments for $i \in \bar{S}_c$ are $\{com_i(0), com_i(0)\}_{i \in \bar{S}_c}$, $\mathcal{B}$ obtains $(X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes com(x^{S_c}) \otimes com(z^{S_c}) \otimes \sigma$.
5. $\mathcal{B}$ runs V_1^* on it to obtain $(c', \{\text{cert}'_{i,x'}, \text{cert}'_{i,z}\}_{i \in \bar{S}_{c'}})$. $\mathcal{B}$ aborts when $c \neq c'$.
6. $\mathcal{B}$ appends $\{d_i(x_i), d_i(z_i)\}_{i \in S_c}$ to the post-measurement state and runs V_2^* on it to obtain σ' .
7. $\mathcal{B}$ sends $\{\text{cert}'_{i,x'}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c}$ to the challenger of $\text{Exp}_{\Sigma_{\text{ccd}}, \mathcal{B}}^{\text{bit-ever hide}}(\lambda, b)$, and receives $\perp$ or $\{d_i(x_i), d_i(z_i)\}_{i \in \bar{S}_c}$ and $\{ck_i(x_i), ck_i(z_i)\}_{i \in \bar{S}_c}$ from the challenger.
8. $\mathcal{B}$ passes $(\perp, \perp)$ to $\mathcal{D}$ if $\mathcal{B}$ receives $\perp$ from the challenger, and passes $(\top, \sigma')$ to $\mathcal{D}$ otherwise.
9. When $\mathcal{D}$ outputs b , $\mathcal{B}$ outputs b .

When $\mathcal{B}$ receives $\{com_i(x_i), com_i(z_i)\}_{i \in \bar{S}_c}$ from the challenger and it does not abort, it simulates $\text{OUT}'_{\mathcal{P}, \mathcal{V}^*}(\mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot))(x)$. Because $(X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes com(x) \otimes com(z) \otimes \sigma$ is independent of c , the probability that $\mathcal{B}$ does not abort is $\frac{1}{m}$. Therefore, $\mathcal{B}$ can simulate $\text{OUT}'_{\mathcal{P}, \mathcal{V}^*}(\mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot))(x)$ with probability $\frac{1}{m}$.

When $\mathcal{B}$ receives $\{com_i(0), com_i(0)\}_{i \in \bar{S}_c}$ from the challenger and it does not abort, it simulates $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$. The probability that $\mathcal{B}$ does not abort is $\frac{1}{m} \pm \text{negl}(\lambda)$ from Propositions 3.3.10 and 3.3.11. Therefore, $\mathcal{B}$ can simulate $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ with probability $\frac{1}{m} \pm \text{negl}(\lambda)$.

Therefore, if there exists a distinguisher $\mathcal{D}$ that distinguishes δ_0 and δ_1 , $\mathcal{B}$ can distinguish $\{com_i(x_i), com_i(z_i)\}_{i \in \bar{S}_c}$ from $\{com_i(0), com_i(0)\}_{i \in \bar{S}_c}$. From Lemma 3.2.7, this contradicts the certified everlasting hiding of Σ_{ccd} . $\square$

Proof of Lemma 3.3.9. Computational zero-knowledge can be proven similarly to [BG20, Lemma 5.3] because our protocol is identical to theirs if we ignore the deletion certificates, which are irrelevant to the computational zero-knowledge property. For the convenience of readers, we provide a proof below.

This proof is similar to the proof of Lemma 3.3.8. For a subset $S_c \subseteq [n]$ and $x, z \in \{0, 1\}^n$, let us define $x^{S_c} := (x_1^{S_c}, x_2^{S_c}, \dots, x_n^{S_c})$ and $z^{S_c} := (z_1^{S_c}, z_2^{S_c}, \dots, z_n^{S_c})$, where $x_i^{S_c} = x_i$ and $z_i^{S_c} = z_i$ for $i \in S_c$, and $x_i^{S_c} = z_i^{S_c} = 0$ for $i \notin S_c$.

For clarity, we describe how the interactive algorithm $\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$ runs against a QPT verifier $\mathcal{V}^*$ with an input ξ , where w is the witness and x is the instance.

$\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\xi) \rangle(x)$:

1. $\mathcal{P}$ generates $x, z \leftarrow \{0, 1\}^n$, and computes

$$(com_i(x_i), d_i(x_i), ck_i(x_i)) \leftarrow \text{Commit}(1^\lambda, x_i)$$

$$(com_i(z_i), d_i(z_i), ck_i(z_i)) \leftarrow \text{Commit}(1^\lambda, z_i)$$

for all $i \in [n]$. $\mathcal{P}$ sends $m_1 := (X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes com(x) \otimes com(z)$ to $\mathcal{V}^*$.

2. $\mathcal{V}^*$ appends ξ to the received state, and runs a QPT circuit V_1^* on it to obtain $(c, \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$. $\mathcal{V}^*$ sends $m_2 := (c, \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$ to $\mathcal{P}$.
3. $\mathcal{P}$ sends $m_3 := \{d_i(x_i), d_i(z_i)\}_{i \in S_c}$ to $\mathcal{V}^*$.
4. $\mathcal{V}^*$ appends m_3 to its state, and runs a QPT circuit V_2^* on it. $\mathcal{V}^*$ outputs its state ξ' .

Next, let us define a simulator $\mathcal{S}^{(1)}$ as follows.

The simulator $\mathcal{S}^{(1)}(x, \mathcal{V}^*, \xi)$:

1. Pick $c \leftarrow [m]$ and $x, z \leftarrow \{0, 1\}^n$. Compute

$$(com_i(x_i^{S_c}), d_i(x_i^{S_c}), ck_i(x_i^{S_c})) \leftarrow \text{Commit}(1^\lambda, x_i^{S_c})$$

$$(com_i(z_i^{S_c}), d_i(z_i^{S_c}), ck_i(z_i^{S_c})) \leftarrow \text{Commit}(1^\lambda, z_i^{S_c})$$

for all $i \in [n]$.

2. Generate $(X^x Z^z \sigma(c) Z^z X^x) \otimes com(x^{S_c}) \otimes com(z^{S_c}) \otimes \xi$, where $\sigma(c) := \rho_{\text{sim}}^{x, S_c} \otimes \left(\prod_{i \in \bar{S}_c} |0\rangle\langle 0|_i \right)$. Run V_1^* on the state to obtain $(c', \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_c})$.

3. If $c' \neq c$, abort and output a fixed state η and the flag state fail.
4. Append $\{d_i(x_i^{S_c}), d_i(z_i^{S_c})\}_{i \in S_c}$ to its quantum state, and run V_2^* on the state. $\mathcal{S}$ outputs the output state and the flag state success.

Let us also define other two simulators $\mathcal{S}^{(2)}$ and $\mathcal{S}^{(3)}$ as follows.

The modified simulator $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \xi)$: It is the same as $\mathcal{S}^{(1)}$ except that $\sigma(c)$ is replaced with ρ_{hist} .

The simulator $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \xi)$: $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ is the channel that postselects the output of $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ on the non-aborting state. More precisely, if we write $\mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \rho_{\text{in}}) = p\rho_{\text{out}} \otimes \text{success} + (1-p)\eta \otimes \text{fail}$, where p is the non-aborting probability, $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \rho_{\text{in}}) = \rho_{\text{out}}$.

Lemma 3.3.9 is shown from the following Propositions 3.3.13 to 3.3.15 (whose proofs will be given later) and quantum rewinding lemma (Lemma 2.3.1), which is used to reduce the probability that $\mathcal{S}^{(1)}$ aborts to $\text{negl}(\lambda)$. In fact, from Proposition 3.3.13 and Lemma 2.3.1, there exists a quantum circuit $\mathcal{S}^{(0)}$ of size at most $O(m \text{poly}(n) \text{size}(\mathcal{S}^{(1)}))$ such that the probability $\mathcal{S}^{(0)}$ aborts is $\text{negl}(\lambda)$, and the output quantum states of $\mathcal{S}^{(0)}$ and $\mathcal{S}^{(1)}$ are $\text{negl}(\lambda)$ -close when they do not abort. From Propositions 3.3.14 and 3.3.15, $\mathcal{S}^{(0)}$ is $\text{negl}(\lambda)$ -close to the run of the real protocol, which completes the proof. $\square$

Proposition 3.3.13. *If Σ_{ccd} is computational hiding, then the probability that $\mathcal{S}^{(1)}$ does not abort is $\frac{1}{m} \pm \text{negl}(\lambda)$.*

Proposition 3.3.14. $\mathcal{S}^{(1)}(x, \mathcal{V}^*, \cdot) \approx_s \mathcal{S}^{(2)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ for any $x \in A_{\text{yes}} \cap \{0, 1\}^\lambda$ and any $w \in R_A(x)$.

Proposition 3.3.15. *If Σ_{ccd} is computational hiding, $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot) \approx_c \text{OUT}_{\mathcal{V}^*}(\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x))$.*

Proof of Proposition 3.3.13. This proof is the same as the proof of Proposition 3.3.10. $\square$

Proof of Proposition 3.3.14. This proof is the same as the proof of Proposition 3.3.11. $\square$

Proof of Proposition 3.3.15. We prove the proposition by contradiction. We construct an adversary $\mathcal{B}$ that breaks the security of the computationally hiding of Σ_{ccd} by assuming the existence of a distinguisher $\mathcal{D}$ that distinguishes two states δ_0 and δ_1 ,

$$\begin{aligned}\delta_0 &:= (\text{OUT}_{\mathcal{V}^*}(\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x)) \otimes I) \sigma \\ \delta_1 &:= (\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot) \otimes I) \sigma,\end{aligned}$$

with a certain state σ . Let us describe how $\mathcal{B}$ works.

1. $\mathcal{B}$ generates $c \leftarrow [m]$ and $x, z \leftarrow \{0, 1\}^n$.
2. $\mathcal{B}$ sends $m_0 := \{x_i, z_i\}_{i \in \overline{S}_c}$ and $m_1 := 0^{2n-10}$ to the challenger. $\mathcal{B}$ receives commitments from the challenger which is either $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in \overline{S}_c}$ or $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in \overline{S}_c}$.

3. $\mathcal{B}$ computes

$$\begin{aligned} (\text{com}_i(x_i), d_i(x_i), \text{ck}_i(x_i)) &\leftarrow \text{Commit}(1^\lambda, x_i) \\ (\text{com}_i(z_i), d_i(z_i), \text{ck}_i(z_i)) &\leftarrow \text{Commit}(1^\lambda, z_i) \end{aligned}$$

for $i \in S_c$ by itself.

4. $\mathcal{B}$ generates $X^x Z^z \rho_{\text{hist}} Z^z X^x$. $\mathcal{B}$ appends commitments and σ to the quantum state. If the commitments for $i \in \bar{S}_c$ are $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in \bar{S}_c}$, $\mathcal{B}$ obtains $X^x Z^z \rho_{\text{hist}} Z^z X^x \otimes \text{com}(x) \otimes \text{com}(z) \otimes \sigma$. If the commitments for $i \in \bar{S}_c$ are $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in \bar{S}_c}$, $\mathcal{B}$ obtains $X^x Z^z \rho_{\text{hist}} Z^z X^x \otimes \text{com}(x^{S_c}) \otimes \text{com}(z^{S_c}) \otimes \sigma$.
5. $\mathcal{B}$ runs V_1^* on it to obtain $(c', \{\text{cert}'_{i,x}, \text{cert}'_{i,z}\}_{i \in \bar{S}_{c'}})$. $\mathcal{B}$ aborts when $c \neq c'$.
6. $\mathcal{B}$ appends $\{d_i(x_i), d_i(z_i)\}_{i \in S_c}$ to the post-measurement state and runs V_2^* on it.
7. $\mathcal{B}$ passes the output state to $\mathcal{D}$.
8. When $\mathcal{D}$ outputs b , $\mathcal{B}$ outputs b .

When $\mathcal{B}$ receives $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in \bar{S}_c}$ from the challenger and it does not abort, it simulates $\text{OUT}_{\mathcal{V}^*}(\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x))$. Because $(X^x Z^z \rho_{\text{hist}} Z^z X^x) \otimes \text{com}(x) \otimes \text{com}(z) \otimes \sigma$ is independent of c , the probability that $\mathcal{B}$ does not abort is $\frac{1}{m}$. Therefore, $\mathcal{B}$ can simulate $\text{OUT}_{\mathcal{V}^*}(\langle \mathcal{P}(w^{\otimes k(|x|)}), \mathcal{V}^*(\cdot) \rangle(x))$ with probability $\frac{1}{m}$.

When $\mathcal{B}$ receives $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in \bar{S}_c}$ from the challenger and it does not abort, it simulates $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$. The probability that $\mathcal{B}$ does not abort is $\frac{1}{m} \pm \text{negl}(\lambda)$ from Propositions 3.3.13 and 3.3.14. Therefore, $\mathcal{B}$ can simulate $\mathcal{S}^{(3)}(x, w^{\otimes k(|x|)}, \mathcal{V}^*, \cdot)$ with probability $\frac{1}{m} \pm \text{negl}(\lambda)$.

Therefore, if there exists the distinguisher $\mathcal{D}$ that distinguishes δ_0 and δ_1 , $\mathcal{B}$ can distinguish $\{\text{com}_i(x_i), \text{com}_i(z_i)\}_{i \in \bar{S}_c}$ from $\{\text{com}_i(0), \text{com}_i(0)\}_{i \in \bar{S}_c}$. From (the computational hiding version of) Lemma 3.2.7, this contradicts the computational hiding of Σ_{ccd} . $\square$

3.3.3 Sequential Repetition for Certified Everlasting Zero-Knowledge Proof for QMA

In this section, we amplify the completeness-soundness gap of the three-round protocol constructed in the previous section by sequential repetition.

Theorem 3.3.16. *Let Σ_{ecd} be a certified everlasting zero-knowledge proof for a QMA promise problem A with $(1 - \text{negl}(\lambda))$ -completeness and $(1 - \frac{1}{\text{poly}(\lambda)})$ -soundness. For any polynomial $N = \text{poly}(\lambda)$, let Σ_{ecd}^N be the N -sequential repetition of Σ_{ecd} . That is, $\mathcal{P}$ and $\mathcal{V}$ in Σ_{ecd}^N run Σ_{ecd} sequentially N times. Let $\mathcal{P}_j$ and $\mathcal{V}_j$ be the prover and the verifier in the j -th run of Σ_{ecd} , respectively. $\mathcal{P}$ in Σ_{ecd}^N outputs $\top$ if $\mathcal{P}_j$ outputs $\top$ for all $j \in [N]$, and outputs $\perp$ otherwise. $\mathcal{V}$ in Σ_{ecd}^N outputs $\top$ if $\mathcal{V}_j$ outputs $\top$ for all $j \in [N]$, and outputs $\perp$ otherwise. Σ_{ecd}^N is a certified everlasting zero-knowledge proof for A with $(1 - \text{negl}(\lambda))$ -completeness and $\text{negl}(\lambda)$ -soundness.*

Proof of Theorem 3.3.16. It is easy to show that Σ_{ecd}^N satisfies $(1 - \text{negl}(\lambda))$ -completeness and $\text{negl}(\lambda)$ -soundness. Moreover, as proven in [GO94], the sequential repetition of a computational zero-knowledge proof preserves the computational zero-knowledge property. Let us show that Σ_{ecd}^N satisfies certified everlasting zero-knowledge. For clarity, we describe how $\langle \mathcal{P}(w^{\otimes Nk(|x|)}), \mathcal{V}^*(\xi_1) \rangle(x)$ runs against any QPT verifier $\mathcal{V}^*$ with an input ξ_1 , where w is a witness and x is the instance.

$\langle \mathcal{P}(w^{\otimes Nk(|x|)}), \mathcal{V}^*(\xi_1) \rangle(x)$:

1. For $1 \leq j \leq N$, $\mathcal{V}^*$ and $\mathcal{P}$ run $\langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x)$ sequentially to get the outputs

$$\xi_{j+1} := \text{OUT}_{\mathcal{V}_j^*} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x)$$

and

$$\text{OUT}_{\mathcal{P}_j} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x) = \top / \perp,$$

respectively.

2. $\mathcal{V}^*$ outputs ξ_{N+1} .
3. $\mathcal{P}$ outputs $\top$ if $\text{OUT}_{\mathcal{P}_j} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle = \top$ for all $j \in [N]$, and outputs $\perp$ otherwise.

Since Σ_{ECD} satisfies the certified everlasting zero-knowledge property, for each $j \in [N]$ and any $\mathcal{V}_j^*$ there exists a QPT algorithm (a simulator) $\mathcal{S}_j(x, \mathcal{V}_j^*, \cdot)$ such that the following holds for any x and w .

$$\text{OUT}'_{\mathcal{P}_j, \mathcal{V}_j^*} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\cdot) \rangle(x) \approx_s \mathcal{S}_j(x, \mathcal{V}_j^*, \cdot).$$

We show that for any $\mathcal{V}^*$ there exists a QPT algorithm (a simulator) $\mathcal{S}(x, \mathcal{V}^*, \cdot)$ such that the following holds for any x and w .

$$\text{OUT}'_{\mathcal{P}, \mathcal{V}^*} \langle \mathcal{P}(w^{\otimes Nk(|x|)}), \mathcal{V}^*(\cdot) \rangle(x) \approx_s \mathcal{S}(x, \mathcal{V}^*, \cdot).$$

Let us define the simulator $\mathcal{S}$ as follows.

The simulator $\mathcal{S}(x, \mathcal{V}^*, \xi_1)$:

1. For $1 \leq j \leq N$, $\mathcal{S}$ runs $\mathcal{S}_j(x, \mathcal{V}_j^*, \cdot)$ on ξ_j to get $\mathcal{S}_j(x, \mathcal{V}_j^*, \xi_j) = (\perp, \perp) / (\top, \xi_{j+1})$ sequentially. If $\mathcal{S}_j(x, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$, then $\xi_{j+1} := \perp$ for each $j \in [N]$.
2. $\mathcal{S}$ outputs $(\perp, \perp)$ if $\mathcal{S}_j(x, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$ for some $j \in [N]$, and outputs $(\top, \xi_{N+1})$ otherwise.

We define the sequence of hybrids $\text{Hybi}(\xi_1)$ as follows.

$\text{Hybi}(\xi_1)$:

1. For $1 \leq j \leq i$, $\mathcal{V}^*$ and $\mathcal{P}$ run $\langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x)$ sequentially to get the outputs

$$\xi_{j+1} := \text{OUT}_{\mathcal{V}_j^*} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x)$$

and

$$\text{OUT}_{\mathcal{P}_j} \langle \mathcal{P}_j(w^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(x) = \top / \perp,$$

respectively.

2. For $i+1 \leq j \leq N$, $\mathcal{S}$ runs $\mathcal{S}_j(x, \mathcal{V}_j^*, \cdot)$ on ξ_j to get $\mathcal{S}_j(x, \mathcal{V}_j^*, \xi_j) = (\perp, \perp) / (\top, \xi_{i+1})$ sequentially. If $\mathcal{S}_j(x, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$, then $\xi_{j+1} := \perp$ for each $j \in [N]$.

3. The output of $\text{Hybi}(\xi_1)$ is $(\perp, \perp)$ if $\text{OUT}_{\mathcal{P}_j}(\mathcal{P}_j(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j))(\mathbf{x}) = \perp$ for some $j \in [i]$ or $\mathcal{S}_j(\mathbf{x}, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$ for some $j \in \{i+1, \dots, N\}$. Otherwise, the output of $\text{Hybi}(\xi_1)$ is $(\top, \xi_{N+1})$.

$\text{Hyb0}(\cdot)$ and $\text{HybN}(\cdot)$ correspond to $\mathcal{S}(\mathbf{x}, \mathcal{V}^*, \cdot)$ and $\text{OUT}'_{\mathcal{P}, \mathcal{V}^*}(\mathcal{P}(\mathbf{w}^{\otimes Nk(|x|)}), \mathcal{V}^*(\cdot))(\mathbf{x})$, respectively. Therefore, it suffices to prove that no distinguisher can distinguish $\text{Hybi}(\cdot)$ from $\text{Hybi} + 1(\cdot)$ for any $i \in [N-1]$. We assume that there exists a distinguisher $\mathcal{D}'$ that distinguishes $(\text{Hybi}(\cdot) \otimes I)\sigma$ from $(\text{Hybi} + 1(\cdot) \otimes I)\sigma$ for a certain state σ , and construct a distinguisher $\mathcal{D}$ that breaks the certified everlasting zero-knowledge property of Σ_{ecd} . $\mathcal{D}$ can access to the channel $\text{O}(\cdot)$, which is either $\mathcal{S}_{i+1}(\mathbf{x}, \mathcal{V}_{i+1}^*, \cdot)$ or $\text{OUT}'_{\mathcal{P}_{i+1}, \mathcal{V}_{i+1}^*}(\mathcal{P}_{i+1}(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_{i+1}^*(\cdot))(\mathbf{x})$, and guesses whether $\text{O}(\cdot)$ is $\mathcal{S}_{i+1}(\mathbf{x}, \mathcal{V}_{i+1}^*, \cdot)$ or $\text{OUT}'_{\mathcal{P}_{i+1}, \mathcal{V}_{i+1}^*}(\mathcal{P}_{i+1}(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_{i+1}^*(\cdot))(\mathbf{x})$. Let us define $\mathcal{D}$ as follows.

The distinguisher $\mathcal{D}(\xi_1)$:

1. For $1 \leq j \leq i$, $\mathcal{D}$ runs $\langle \mathcal{P}_j(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j) \rangle(\mathbf{x})$ sequentially to get $\xi_{j+1} := \text{OUT}_{\mathcal{V}_j^*}(\mathcal{P}_j(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j))(\mathbf{x})$ and $\text{OUT}_{\mathcal{P}_j}(\mathcal{P}_j(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j))(\mathbf{x}) = \top / \perp$.
2. $\mathcal{D}$ runs $\text{O}(\xi_{i+1})$ to get $\text{O}(\xi_{i+1}) = (\perp, \perp) / (\top, \xi_{i+2})$. If $\text{O}(\xi_{i+1}) = (\perp, \perp)$, $\mathcal{D}$ sets $\xi_{i+2} := \perp$.
3. For $i+2 \leq j \leq N$, $\mathcal{D}$ runs $\mathcal{S}_j(\mathbf{x}, \mathcal{V}_j^*, \cdot)$ on ξ_j to get $\mathcal{S}_j(\mathbf{x}, \mathcal{V}_j^*, \xi_j) = (\perp, \perp) / (\top, \xi_{j+1})$ sequentially. If $\mathcal{S}_j(\mathbf{x}, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$, $\mathcal{D}$ sets $\xi_{j+1} := \perp$.
4. $\mathcal{D}$ outputs $(\perp, \perp)$ if $\text{OUT}_{\mathcal{P}_j}(\mathcal{P}_j(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_j^*(\xi_j))(\mathbf{x}) = \perp$ for some $j \in [i]$, $\text{O}(\xi_{i+1}) = (\perp, \perp)$ or $\mathcal{S}_j(\mathbf{x}, \mathcal{V}_j^*, \xi_j) = (\perp, \perp)$ for some $j \in \{i+2, \dots, N\}$, and outputs $(\top, \xi_{N+1})$ otherwise.
5. $\mathcal{D}$ sends the output of $\mathcal{D}$ to $\mathcal{D}'$.
6. If $\mathcal{D}'$ outputs b , $\mathcal{D}$ outputs b .

We can see that $\mathcal{D}$ generates $(\text{Hybi}(\cdot) \otimes I)\sigma$ when $\text{O}(\cdot)$ is $\mathcal{S}_{i+1}(\mathbf{x}, \mathcal{V}_{i+1}^*, \cdot)$ and $\mathcal{D}$ takes σ as input. Similarly, we can see that $\mathcal{D}$ generates $(\text{Hybi} + 1(\cdot) \otimes I)\sigma$ when $\text{O}(\cdot)$ is $\text{OUT}'_{\mathcal{P}_{i+1}, \mathcal{V}_{i+1}^*}(\mathcal{P}_{i+1}(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_{i+1}^*(\cdot))(\mathbf{x})$ and $\mathcal{D}$ takes σ as input. Therefore, if $\mathcal{D}'$ distinguishes $(\text{Hybi}(\cdot) \otimes I)\sigma$ from $(\text{Hybi} + 1(\cdot) \otimes I)\sigma$, then $\mathcal{D}$ can distinguish $\mathcal{S}_{i+1}(\mathbf{x}, \mathcal{V}_{i+1}^*, \cdot)$ from $\text{OUT}'_{\mathcal{P}_{i+1}, \mathcal{V}_{i+1}^*}(\mathcal{P}_{i+1}(\mathbf{w}^{\otimes k(|x|)}), \mathcal{V}_{i+1}^*(\cdot))(\mathbf{x})$. This contradicts the certified everlasting zero-knowledge property of Σ_{ecd} , which completes the proof. $\square$

Chapter 4

Certified Everlasting Secure Functional Encryption

4.1 Introduction

In this chapter, we study a certified everlasting secure version of functional encryption (FE). Though there are several constructions of FE, the security of all previous constructions is a computational one. Hence, in the previous constructions of FE, an adversary with a ciphertext $\text{ct}(m)$ could obtain the information of m in the future when computational assumptions are broken. Furthermore, unfortunately, FE is known as impossible to achieve information theoretical security. In this thesis, we formalize certified everlasting secure versions of various kinds of FE, and show how to construct them.

In Section 4.2, we give a technical overview of our constructions of certified everlasting secure FE. In Section 4.3, we formalize a certified everlasting secure version of collusion-resistant FE and show how to construct it from indistinguishability obfuscation (iO) and one-way functions (OWFs). In Section 4.4, we formalize a certified everlasting version of bounded collusion-resistant FE for NC^1 and show how to construct it from PKE. In Section 4.5, we define a certified everlasting version of compute-and-compare obfuscation and show how to construct it from compute-and-compare obfuscation and FHE. In Section 4.6, we define a certified everlasting version of predicate encryption and show how to construct it from a certified everlasting secure version of compute-and-compare obfuscation and a standard attribute-based encryption. In the following, we explain the background of these results in detail.

4.1.1 Background

In cryptography, computational security is based on the assumption that certain problems are hard to solve. However, these assumptions might be broken in the future due to revolutionary breakthroughs in algorithms or significant advancements in computing hardware. One solution to this issue is to construct cryptographic protocols with information-theoretical security. Unfortunately, many cryptographic primitives are known to be impossible or unlikely to achieve information-theoretical security, even in the quantum realm [LC97, May97, MW18].

Good compromise (intrinsic to quantum properties) has been explored in recent studies [Unr15, BI20, KT20, HMNY21, HMNY22b, Por23]. Notably, certified everlasting security, introduced in [HMNY22b] based on [Unr15, BI20], provides an appealing security guarantee: After receiving quantum-encrypted data, a receiver can issue a certificate proving they have deleted the encrypted data. If the certificate is valid, security

is guaranteed even if the receiver gains unbounded computational power later. A verification key (either private or public) is generated along with the quantum-encrypted data to authenticate certificates.

Certified everlasting security is weaker than information-theoretical security, as a malicious receiver could refuse to issue a valid certificate. However, it remains valuable because a sender can penalize receivers that fail to issue valid certificates. Furthermore, certified everlasting security is unique to quantum because it implies information-theoretical security in the classical world.¹ Certified everlasting security bypasses the impossibility results of information-theoretical security. Several cryptographic primitives, such as commitments and zero-knowledge proofs, have already been shown to achieve this security [HMNY22b]. However, an important open question remains:

Which cryptographic primitives can achieve certified everlasting security?

Functional encryption (FE) is one of the most advanced cryptographic primitives, achieving flexibility in controlling encrypted data [BSW11]. In FE, the owner of a master secret key MSK can generate a functional decryption key sk_f , which hardwires a function f . When a ciphertext ct_m of a message m is decrypted with sk_f , we can obtain only the value of $f(m)$, and no additional information about m is leaked. Information-theoretically secure FE is impossible, and all known constructions rely on computational assumptions [GVW12, GGH⁺16, AP20, AV19, JLS21, JLS22]. A motivating application of FE is to enable secure analysis of sensitive data—such as DNA—while ensuring data privacy. In such cases, users store encrypted data on a remote server and delegate computations. At some point, users might request the server to "forget" their encrypted data. If FE does not possess *certified everlasting security*, rapid advancements in computational power could compromise the privacy of sensitive personal data in the future, exposing information like DNA, which may affect future generations. Encryption with certified deletion could be useful to avoid such realistic attacks.

Therefore, 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 public key encryption (PKE) or other "all-or-nothing" encryption schemes [GMM17].² Specifically, certified everlasting FE ensures that even if a receiver possesses a functional decryption key sk_f for a function f , they cannot learn more than $f(m)$, even with unbounded computational resources after issuing a valid certificate. In contrast, certified everlasting secure PKE offers no such guarantee, as an honest receiver could simply keep the plaintext after decrypting the ciphertext.

Another powerful cryptographic primitive is obfuscation for compute-and-compare programs [WZ17], also known as lockable obfuscation [GKW17]. A compute-and-compare obfuscation scheme can obfuscate a circuit that computes a function P and compares its output with a locked value $lock$. If the input matches the lock value, the circuit returns a message m ; otherwise, it returns $\perp$. Examples include point functions, conjunctions with wildcards, plaintext checkers, and affine testers [GKW17, WZ17]. Hence, certified everlasting secure compute-and-compare obfuscation would enable certified deletion of obfuscated programs within a restricted class of functionalities. Compute-and-compare obfuscation has many applications in cryptography [GKW17, WZ17, CVW⁺18, FFMV23, AYY22, AKYY23]. For example, we can convert

¹In the classical setting, a malicious receiver can freely copy encrypted data, so security against an unbounded adversary must hold at the point it receives the encrypted data. In contrast, the quantum no-cloning theorem prevents a malicious receiver from duplicating quantum-encrypted data.

²Examples include identity-based encryption (IBE), attribute-based encryption (ABE), fully homomorphic encryption (FHE), or witness encryption (WE).

attribute-based encryption (ABE) to predicate encryption (PE) via compute-and-compare obfuscation enables predicate encryption (PE) [KSW08, GVW15b], which is an attribute-hiding variant of ABE and an intermediate primitive between ABE and FE. Achieving certified everlasting secure compute-and-compare obfuscation could thus lead to certified everlasting secure PE, as well as anonymous IBE and PKE.

This gives rise to another significant open problem:

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

4.1.2 Our Results

Contributions. This work addresses the open problems outlined earlier, and our main contributions are summarized as follows:

1. We formally define certified everlasting versions of several cryptographic primitives: FE (Section 4.3.1), compute-and-compare obfuscation (Section 4.5.1), PE (Section 4.6.1), SKE and PKE (Appendix A.3.1), RNCE (Appendix A.4.1), and garbling schemes (Appendix A.5.1).
2. We construct adaptively certified everlasting secure collusion-resistant public-key FE for $\mathbf{P}/\mathbf{poly}$ using indistinguishability obfuscation (iO) and one-way functions (OWFs) (Section 4.3.3). Additionally, we construct adaptively certified everlasting secure bounded collusion-resistant public-key FE for $\mathbf{NC}^1$ based on standard PKE (Section 4.4.4).
3. We develop certified everlasting secure compute-and-compare obfuscation from standard FHE and compute-and-compare obfuscation (Section 4.5.2), both realizable under the LWE assumption. Furthermore, we construct adaptively (resp., selectively) certified everlasting secure PE from standard adaptively (resp., selectively) secure ABE and certified everlasting secure compute-and-compare obfuscation (Section 4.6.2).
4. To achieve adaptively certified everlasting secure bounded collusion-resistant FE, we construct the following cryptographic primitives:
 - Two certified everlasting secure SKE constructions from standard SKE (Appendices A.3.2 and A.3.3): one with a classical certificate relying on the quantum random oracle model (QROM) [BDF⁺11], and another with a quantum certificate but without QROM dependence.
 - Two certified everlasting secure PKE constructions with similar properties as the SKE constructions above (Appendices A.3.4 and A.3.5).
 - A certified everlasting secure RNCE scheme from certified everlasting PKE (Appendix A.4.2).
 - A certified everlasting secure garbling scheme for $\mathbf{P}/\mathbf{poly}$ from certified everlasting SKE (Appendix A.5.2).

All our constructions are privately verifiable, meaning that verification keys for deletion certificates must remain secret. Achieving certified everlasting secure bounded collusion-resistant FE for $\mathbf{P}/\mathbf{poly}$ from standard PKE remains an open problem.

We introduce novel techniques to achieve certified everlasting secure collusion-resistant FE and compute-and-compare obfuscation. For FE, we developed an authentication technique for BB84 states to ensure both functionality and certified everlasting security, detailed in Section 4.2.1. This technique is broadly applicable and was utilized in subsequent work by Kitagawa, Nishimaki, and Yamakawa [KNY23]. For compute-and-compare obfuscation, we introduce a deferred evaluation technique using dummy lock values to balance functionality and certified everlasting security, detailed in Section 4.2.3.

4.2 Technical Overview

4.2.1 Technical Overview: Collusion-Resistant FE

Certified Everlasting Lemma of Bartusek and Khurana. The cornerstone of our construction is the *certified everlasting lemma*, introduced by Bartusek and Khurana [BK23]. This lemma provides a powerful mechanism for ensuring certified everlasting security, which we summarize below.

Consider a family of distributions $\mathcal{Z}(m)_{m \in \{0,1\}^{\lambda+1}}$ over classical strings. For any $m \in \{0,1\}^{\lambda+1}$, the distribution $\mathcal{Z}(m)$ is computationally indistinguishable from $\mathcal{Z}(0^{\lambda+1})$. Conceptually, $\mathcal{Z}(m)$ can be thought of as a computational “encryption” of m . To analyze this setting, we define the following experiment $\tilde{\mathcal{Z}}(b)$ for $b \in \{0,1\}$:

- Randomly sample $z, \theta \leftarrow \{0,1\}^\lambda$.
- Provide the adversary with the quantum state $|z\rangle$, the classical string θ , and the distribution $\mathcal{Z}(\theta, b \oplus \bigoplus_{j:\theta_j=0} z_j)$, where z_j is the j -th bit of z .
- The adversary outputs a string $z' \in \{0,1\}^\lambda$ and a quantum state ρ .
- If $z'_j = z_j$ for all indices j where $\theta_j = 1$, the experiment outputs ρ . Otherwise, it outputs a special symbol $\perp$.

The lemma states that for any quantum polynomial-time (QPT) adversary, the trace distance between the outputs of $\tilde{\mathcal{Z}}(0)$ and $\tilde{\mathcal{Z}}(1)$ is negligible in λ .³

This lemma serves as a versatile tool for constructing cryptographic schemes with certified everlasting security. For example, one can transform a standard public-key encryption (PKE) scheme into a certified everlasting PKE scheme. To encrypt a bit $b \in \{0,1\}$, the ciphertext consists of $|z\rangle, \theta$, and $\text{Enc}(\theta, b \oplus \bigoplus_{j:\theta_j=0} z_j)$, where $z, \theta \leftarrow \{0,1\}^\lambda$ and Enc is the encryption algorithm of the original PKE scheme.

Decryption involves measuring $|z\rangle$ in the Hadamard basis to produce a certificate z' . Verification checks if $z'_j = z_j$ for all j where $\theta_j = 1$. The lemma guarantees that, given a valid certificate, the adversary learns no information about b , thus achieving certified everlasting security.

Public-slot FE. Extending the certified everlasting lemma to functional encryption (FE) is not straightforward. The primary challenge lies in the inability to define a function f' that allows recovery of $f(m)$ from $f'(m \oplus \bigoplus_{j:\theta_j=0} z_j, \theta)$ and z .

To overcome this limitation, we introduce *public-slot functional encryption (public-slot FE)*. In this variant, decryption keys correspond to *two-input functions*, which take both a secret input and a public input. Given a ciphertext encrypting a message m , and a decryption key for a function f , one can compute $f(m, \text{pub})$ for any chosen public input pub . Security is defined similarly to standard FE, with the additional condition that for all decryption key queries f and public inputs pub , the challenge messages $(m^{(0)}, m^{(1)})$ must satisfy $f(m^{(0)}, \text{pub}) = f(m^{(1)}, \text{pub})$.

Many existing FE constructions based on indistinguishability obfuscation (iO) (e.g., [GGH⁺16]) can naturally be adapted to support public-slot FE. Specifically, we demonstrate that a minor modification to the FE scheme by Ananth and Sahai [AS16] yields an adaptively secure public-slot FE scheme, relying on iO. Details of this construction are presented in Appendix A.2.4.

³An interactive version of this lemma is often necessary for practical use. Such a version is implicitly established in [BK23]. See Lemma 4.3.7 for details and Remark 4.3.8 for a discussion.

First attempt. Our first candidate construction of a collusion-resistant functional encryption (FE) scheme with certified everlasting security proceeds as follows. Let Enc be the encryption algorithm of a public-slot FE scheme. A ciphertext for a message $m = m_1 \dots m_n \in \{0, 1\}^n$ consists of the following two items:

- $\{|z_i\rangle_{\theta_i}\}_{i \in [n]}$, where $z_i, \theta_i \leftarrow \{0, 1\}^\lambda$ for each $i \in [n]$.
- $\text{Enc}(\theta_1, \dots, \theta_n, \beta_1, \dots, \beta_n)$. Here, $\beta_i := m_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ for each $i \in [n]$, where $z_{i,j}$ is the j -th bit of z_i .

A decryption key for a function f corresponds to the decryption key of the underlying public-slot FE for a two-input function $g[f]$, defined as follows. The function $g[f]$ takes as input a secret input $(\theta_1, \dots, \theta_n, \beta_1, \dots, \beta_n)$, and a public input $(b_1, \dots, b_n) \in \{0, 1\}^{\lambda \times n}$. It computes $m_i := \beta_i \oplus \bigoplus_{j: \theta_{i,j}=0} b_{i,j}$ for each $i \in [n]$, and outputs $f(m_1, \dots, m_n)$.

To verify decryption correctness, observe that if we first measure $|z_i\rangle_{\theta_i}$ for all $i \in [n]$ in the computational basis to obtain $(b_1, \dots, b_n)$, then for all i, j such that $\theta_{i,j} = 0$, we have $b_{i,j} = z_{i,j}$. Thus, running the decryption algorithm of the public-slot FE scheme with the public input $(b_1, \dots, b_n)$ produces the correct output $f(m_1, \dots, m_n)$. We remark that decryption can also be performed coherently without directly measuring $\{|z_i\rangle_{\theta_i}\}_{i \in [n]}$. The deletion and verification algorithms are defined similarly to those of the certified everlasting public-key encryption (PKE) scheme, as discussed earlier: the deletion algorithm measures $\{|z_i\rangle_{\theta_i}\}_{i \in [n]}$ in the Hadamard basis to obtain $\{z'_i\}_{i \in [n]}$, and the verification algorithm checks whether $z'_{i,j} = z_{i,j}$ for all i, j such that $\theta_{i,j} = 1$.

However, this scheme turns out to be insecure. The core issue is that public-slot FE does not restrict an adversary from using arbitrary public inputs. By running the decryption algorithm with different public inputs multiple times, an adversary can extract more information than just $f(m_1, \dots, m_n)$, which compromises security. For instance, if the adversary chooses a public input $(b_1, \dots, b'_j, \dots, b_n)$, where b'_j is identical to b_j except for $b'_{i,j} \neq b_{i,j}$ for some j such that $\theta_{i,j} = 0$, they can learn $f(m_1, \dots, 1 - m_i, \dots, m_n)$, revealing more information than intended.

Certify the public input by one-time signatures. Our idea to resolve the issue is to certify $\{z_i\}_{i \in [n]}$ in the quantum part of the ciphertext using one-time signatures.

Specifically, the encryption algorithm generates a pair of a verification key $\text{vk}_{i,j}$ and a signing key $\text{sk}_{i,j}$ for a deterministic one-time signature for each $i \in [n]$ and $j \in [\lambda]$. A ciphertext for a message $m = m_1 \dots m_n \in \{0, 1\}^n$ consists of the following two items:

- $\text{Enc}(\{\text{vk}_{i,j}\}_{i \in [n], j \in [\lambda]}, \theta_1, \dots, \theta_n, \beta_1, \dots, \beta_n)$. Here, $\beta_i := m_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$, where $z_{i,j}$ is the j -th bit of z_i and $z_i, \theta_i \leftarrow \{0, 1\}^\lambda$ for each $i \in [n]$.
- Quantum states $\{|\psi_{i,j}\rangle\}_{i \in [n], j \in [\lambda]}$ defined as follows:

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

where $\sigma_{i,j,b}$ is the signature generated using the signing key $\text{sk}_{i,j}$ on the bit $b \in \{0, 1\}$.

This construction allows the state $|\psi_{i,j}\rangle$ to be viewed as a coherent execution of the signing algorithm with $\text{sk}_{i,j}$ on the j -th qubit of $|z_i\rangle_{\theta_i}$. We modify the function $g[f]$, associated with the decryption key in the public-slot FE, to additionally verify the authenticity of the signatures for $b_{i,j}$ for all i, j such that $\theta_{i,j} = 0$.

Formally, $g[f]$ now works as follows:

- Take a secret input $(\{vk_{i,j}\}_{i \in [n], j \in [\lambda]}, \theta_1, \dots, \theta_n, \beta_1, \dots, \beta_n)$, and a public input $(b_1, \dots, b_n, \sigma_1, \dots, \sigma_n)$, where $\sigma_i = (\sigma_{i,1}, \dots, \sigma_{i,\lambda})$ for each $i \in [n]$.
- Verify whether $\sigma_{i,j}$ is a valid signature for $b_{i,j}$ (i.e., $\sigma_{i,j} = \sigma_{i,j,b_{i,j}}$) for all i, j such that $\theta_{i,j} = 0$.
- If any verification fails, $g[f]$ outputs $\perp$. Otherwise, it computes $m_i := \beta_i \oplus \bigoplus_{j:\theta_{i,j}=0} b_{i,j}$ for all $i \in [n]$ and outputs $f(m_1, \dots, m_n)$.

Notably, the state $|\psi_{i,j}\rangle$ contains the valid signature $\sigma_{i,j,z_{i,j}}$ on the bit $z_{i,j}$ whenever $\theta_{i,j} = 0$. Therefore, the correctness of decryption remains intact. Moreover, if we measure $|\psi_{i,j}\rangle$ in the Hadamard basis for i, j such that $\theta_{i,j} = 1$, the outcome $(c_{i,j}, d_{i,j})$ satisfies the relationship $z_{i,j} = c_{i,j} \oplus d_{i,j}(\sigma_{i,j,0} \oplus \sigma_{i,j,1})$. By modifying the verification algorithm to check this equality, we ensure verification correctness as well. Due to the security of the one-time signatures, an adversary is unable to arbitrarily modify the public input when running the decryption algorithm of the underlying public-slot FE.

While this approach seems to prevent straightforward attacks, the challenge remains in proving certified everlasting security for this scheme. Specifically, we wish to leverage the certified everlasting lemma from [BK23], which only allows for bit-wise game transitions. For instance, consider the case where $n = 3$ and the challenge messages are 000 and 111. The necessary hybrid experiments would evolve the challenge message as $000 \rightarrow 100 \rightarrow 110 \rightarrow 111$.⁴ However, the restriction on adversaries only guarantees $f(000) = f(111)$ for decryption key queries f , and not necessarily $f(000) = f(100)$. Without this guarantee, we cannot rely on the security of the underlying public-slot FE, making it difficult to establish indistinguishability between neighboring hybrid experiments.

Redundant encoding. To address the issue mentioned earlier, our approach is to encode the message in a redundant manner, creating space for an additional "spare message." Specifically, we 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 remains the same as described in the previous paragraph, except that the index i now ranges over $[2n + 1]$ instead of $[n]$, and the function $g[f]$ determines which part of the encoded message to use for deriving the output based on the value of the $(2n + 1)$ -th bit.

More precisely, $g[f]$ takes a secret input $(\{vk_{i,j}\}_{i \in [2n+1], j \in [\lambda]}, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1})$, a public input $(b_1, \dots, b_{2n+1}, \sigma_1, \dots, \sigma_{2n+1})$, and first verifies the signatures on the positions i, j where $\theta_{i,j} = 0$, just like 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 remains intact since we always have $m_{2n+1} = 0$ when decrypting an honestly generated ciphertext. Similarly, the verification correctness is not affected because the message encoding method is independent of the verification process.

This redundant encoding strategy solves the earlier problem by ensuring that, at every step of the intermediate hybrids, the encoded challenge message contains at least one of the two challenge messages. Let $m^{(0)}$ and $m^{(1)}$ be a pair of challenge messages. After encoding, they correspond to $m^{(0)} \| 0^{n+1}$ and $m^{(1)} \| 0^{n+1}$, respectively. We then define a series of intermediate hybrids as follows:

⁴Although constructing an FE scheme for 3-bit messages is trivial from any PKE scheme, this example highlights a technical difficulty.

1. Starting from $m^{(0)} \parallel 0^{n+1}$, we change the bits from the $(n+1)$ -th to the $2n$ -th positions one by one, progressing toward $m^{(0)} \parallel m^{(1)} \parallel 0$.
2. Flip the $(2n+1)$ -th bit, resulting in $m^{(0)} \parallel m^{(1)} \parallel 1$.
3. Change the bits from the first to the n -th position one by one, progressing toward $m^{(1)} \parallel m^{(1)} \parallel 1$.
4. Flip the $(2n+1)$ -th bit again, resulting in $m^{(1)} \parallel m^{(1)} \parallel 0$.
5. Finally, change the bits from the $(n+1)$ -th to the $2n$ -th positions one by one, progressing toward $m^{(1)} \parallel 0^{2n+1}$.

Crucially, at every stage in the hybrid sequence, the value of F evaluated on the encoded challenge message is always $f(m^{(0)}) = f(m^{(1)})$. This allows us to apply the certified everlasting lemma alongside the security of the underlying public-slot FE in each hybrid step.

What one-time signatures to use? Finally, we remark that we need to choose one-time signature schemes carefully. The primary reason for incorporating one-time signatures is to prevent an adversary from using "unauthorized" values of $b_{i,j}$ —i.e., those for which the valid signature $\sigma_{i,j,b_{i,j}}$ has not been provided to the adversary. However, due to the correctness requirement of one-time signatures, a valid signature must exist for every possible message. This implies that even for an unauthorized $b_{i,j}$, a valid signature will exist, even though it may be difficult for the adversary to find.

This situation is problematic in light of the security definition for public-slot functional encryption (FE). Recall that the security of public-slot FE demands that for all key queries f and *all* public inputs pub , the challenge message pair $(m^{(0)}, m^{(1)})$ must satisfy $f(m^{(0)}, \text{pub}) = f(m^{(1)}, \text{pub})$. The security definition becomes inapplicable if there exists any pub such that $f(m^{(0)}, \text{pub}) \neq f(m^{(1)}, \text{pub})$, even if such a pub is hard to find.

To address this issue, we utilize Lamport signatures instantiated with a pseudorandom generator (PRG). Let $\text{PRG} : \{0, 1\}^\lambda \rightarrow \{0, 1\}^{2\lambda}$ be a PRG. When the message length is 1, a signing key is represented as $(u_0, u_1) \in \{0, 1\}^{\lambda \times 2}$, and a verification key is set to $(v_0 = \text{PRG}(u_0), v_1 = \text{PRG}(u_1)) \in \{0, 1\}^{2\lambda \times 2}$. A signature for a bit b is simply u_b .

This construction has a critical property: we can "program" a verification key so that it does not have a valid signature for a particular message. For instance, if we want to ensure that no valid signature exists for the message 0, we can set v_0 to be a uniformly random 2λ -bit string. With probability $1 - 2^{-\lambda}$, there will be no preimage of v_0 , meaning that no valid signature exists for the message 0. Using this property, whenever a particular $b_{i,j}$ is deemed unauthorized, we can transition to a hybrid in which there is no valid signature for $b_{i,j}$. This effectively resolves the issue by ensuring that unauthorized values cannot be exploited by the adversary.

4.2.2 Technical Overview: Bounded Collusion-Resistant FE

In this subsection, we present a high-level overview of our construction for certified everlasting secure bounded collusion-resistant functional encryption (FE) schemes. It is known that (bounded collusion-resistant) plain FE can be constructed from public-key encryption (PKE), receiver non-committing encryption (RNCE), and garbling [GVW12]. A natural approach is to construct PKE, RNCE, and garbling with certified everlasting security and use them as the fundamental components. We demonstrate that PKE with certified everlasting security can be constructed using the techniques introduced in [Unr15, HMNY22b]. Additionally, RNCE with certified everlasting security for *classical messages* can be built from certified everlasting PKE in a manner similar to standard RNCE [KNTY19]. However, such an RNCE scheme is insufficient for our purposes of

constructing adaptively-secure FE, as it does not handle *quantum messages*. Moreover, we need to devise a new method to construct garbling with certified everlasting security. The subsequent sections outline these ideas and how they contribute to constructing FE with certified everlasting security.

Certified everlasting garbling for P/poly circuits. In classical cryptography, it is known that we can construct plain garbling from plain symmetric key encryption (SKE) using double-encryption [Yao86, LP09]. Double-encryption refers to generating a nested ciphertext $ct_2 \leftarrow \text{Enc}(sk', ct_1)$, where $ct_1 \leftarrow \text{Enc}(sk, m)$, m is the message, Enc is the encryption algorithm of SKE, and sk, sk' are the secret keys of SKE. This technique is crucial for garbling but poses a challenge for our objective. First, we do not have an SKE scheme with certified everlasting security for *quantum* messages. Second, even if this issue is resolved, there remains another problem: We can obtain a valid certificate showing that ct_1 has been deleted by running the deletion algorithm on ct_2 . However, this certificate does not necessarily confirm the deletion of m . To overcome these issues, we utilize XOR secret sharing instead of double-encryption.⁵

Specifically, we uniformly sample p and compute $(vk', ct') \leftarrow \text{Enc}(sk', p)$ and $(vk, ct) \leftarrow \text{Enc}(sk, p \oplus m)$ to encrypt the message m . Here, Enc is the encryption algorithm of a certified everlasting SKE, and vk', vk are verification keys used to verify deletion certificates. The receiver can retrieve m from (ct', ct) only if it has both sk' and sk , and cannot retrieve anything otherwise, similar to double-encryption. Furthermore, once the receiver issues a deletion certificate for (ct', ct) , it loses the ability to access m even if it becomes computationally unbounded.

It is straightforward to see that we can implement the well-known gate garbling technique [Yao86, LP09] using this XOR secret sharing method in a parallel manner instead of sequential double-encryption. We can prove computational security through a similar argument to that in [LP09]. (While [LP09] uses double-encryption, we can demonstrate security for the XOR secret sharing case analogously.) Furthermore, certified everlasting security can be established by leveraging the certified everlasting security of the SKE. Thus, we obtain certified everlasting garbling. The formal construction of our certified everlasting garbling is given in Appendix A.5.2. For more details, refer to that section.

FE with non-adaptive security. Our next goal is to achieve 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 the labels of a plain garbling scheme [SS10].⁶ In our certified everlasting garbling scheme (as described in the previous paragraph), the labels are classical bit strings, and the deletion algorithm does not take the labels as input. Therefore, the classical construction for plain FE by Sahai and Seyalioglu [SS10] can be directly applied to construct our 1-bounded certified everlasting FE for P/poly circuits with *non-adaptive security*. (The formal construction is presented in Section 4.4.2.)

FE with adaptive security. Now, we want to convert non-adaptive security to adaptive security.⁷ However, this conversion is not trivial. Let us first review the conversion for plain FE.

In classical cryptography, we can transform non-adaptively secure FE into adaptively secure FE using receiver non-committing encryption (RNCE). Roughly speaking, RNCE works similarly to public key encryption (PKE), but with the ability to generate a fake ciphertext $\tilde{ct} \leftarrow \text{Fake}(pk)$ without a plaintext and a fake secret key $\tilde{sk} \leftarrow \text{Reveal}(pk, m)$ that decrypts $\tilde{ct}$ to m . The security of RNCE ensures that

⁵A similar technique was employed by Gentry, Halevi, and Vaikuntanathan [GHV10].

⁶Non-adaptive security implies that the adversary can only make key queries before the challenge encryption query.

⁷Adaptive security implies that the adversary can make key queries both before and after the challenge encryption query.

$(\text{Enc}(\text{pk}, m), \text{sk})$ and $(\text{Fake}(\text{pk}), \text{Reveal}(\text{pk}, m))$ are computationally indistinguishable, where Enc is the real encryption algorithm, and sk is the real secret key.

To achieve adaptive security for FE, we can run the real encryption algorithm Enc of RNCE on the ciphertext nad.ct of the FE. The adaptive security can be proven as follows: The adversary, in the adaptive setting, can issue key queries after the challenge encryption query. However, the sender can simulate the challenge encryption query without generating nad.ct , because the security of RNCE allows us to switch from $(\text{Enc}(\text{pk}, \text{nad.ct}), (\text{sk}, \text{nad.sk}_f))$ to the fake pair $(\text{Fake}(\text{pk}), (\text{Reveal}(\text{pk}, \text{nad.ct}), \text{nad.sk}_f))$, where nad.sk_f is the functional secret key of the non-adaptively secure FE. Thus, the sender need not generate nad.ct before generating nad.sk_f for simulating the adversary's queries, allowing us to reduce adaptive security to non-adaptive security.

How can we adapt this classical approach to the certified everlasting setting? From the above discussion, a natural approach would be to encrypt the ciphertext nad.ct of the certified everlasting FE with non-adaptive security using certified everlasting RNCE as follows: $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, \text{nad.ct})$, 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 approach fails for two reasons. First, nad.ct is a quantum state, and our certified everlasting RNCE scheme does not support quantum messages. Second, even if we could construct RNCE for quantum messages, there remains another issue: A valid certificate for ct is issued by running the deletion algorithm on ct , but such a certificate does not necessarily guarantee the deletion of the plaintext of nad.ct . The first issue pertains to security, and the second to correctness.

Our approach to solving the first issue is to employ quantum teleportation. We construct RNCE for quantum messages from RNCE for classical messages using quantum teleportation.⁸ (We believe the use of quantum teleportation in this way has potential applications beyond RNCE.)

As the ciphertext and secret key for adaptively secure FE, we use

$$\frac{1}{2^{2N}} \sum_{a,b \in \{0,1\}^N} (Z^b X^a (\text{nad.ct}) X^a Z^b)_{C_1} \otimes \text{Enc}(\text{pk}, (a, b))_{C_2} \otimes (\text{nad.sk}_f, \text{sk})_S,$$

where nad.ct is an N -qubit state, registers C_1 and C_2 contain the ciphertext, and register S contains the secret key. Here, $X^a := \bigotimes_{j=1}^N X_j^{a_j}$ and $Z^b := \bigotimes_{j=1}^N Z_j^{b_j}$, where a_j and b_j are the j -th bits of a and b , respectively. Moreover, Enc is the encryption algorithm of RNCE for classical messages, nad.sk_f is the secret key of the non-adaptively secure FE, and sk is the real secret key of RNCE for classical messages.

To demonstrate adaptive security, we reduce it to the non-adaptive security of the underlying FE. In the first step of the hybrid argument, we switch the state to

$$\frac{1}{2^{2N}} \sum_{a,b} (Z^b X^a (\text{nad.ct}) X^a Z^b)_{C_1} \otimes \text{Fake}(\text{pk})_{C_2} \otimes (\text{nad.sk}_f, \text{Reveal}(\text{pk}, (a, b)))_S,$$

using the properties of RNCE for classical messages. In the second step of the hybrid argument, we switch the state to

$$\frac{1}{2^{2N}} \sum_{x,z \in \{0,1\}^N} \mathcal{T}_{A',A}^{x,z} [\text{nad.ct}_{A'} \otimes |\Phi_N\rangle \langle \Phi_N|_{A,C_1}] \otimes \text{Fake}(\text{pk})_{C_2} \otimes (\text{nad.sk}_f, \text{Reveal}(\text{pk}, (x, z)))_S,$$

where $|\Phi_N\rangle$ is N Bell pairs between registers A and C_1 . $\mathcal{T}_{A',A}^{x,z} [\text{nad.ct}_{A'} \otimes |\Phi_N\rangle \langle \Phi_N|_{A,C_1}]$ is the state on register C_1 obtained as follows: the state $\text{nad.ct}_{A'}$ on register A' is entangled with half of N Bell pairs on

⁸A similar technique was applied in the context of multi-party quantum computation [BCKM21].

register A , and the teleportation measurement $\mathcal{T}_{A',A}^{x,z}$ with result (x,z) is applied to registers A and A' . At this point, the sender can simulate the challenge encryption query without knowledge of nad.ct , meaning that the sender need not generate nad.ct before generating nad.sk_f for simulating the adversary's queries.

This idea resolves the first issue. However, the second issue remains: The receiver, possessing $(Z^b X^a(\text{nad.ct}) X^a Z^b, \text{Enc}(\text{pk}, (a, b)))$, can issue a deletion certificate for $Z^b X^a(\text{nad.ct}) X^a Z^b$. This deletion certificate does not necessarily pass the verification algorithm for the deletion of nad.ct , which poses a correctness issue.

We solve this issue by introducing an efficient algorithm, referred to as the *modification algorithm*. Let nad.cert^* denote the deletion certificate of $Z^b X^a(\text{nad.ct}) X^a Z^b$. The modification algorithm takes (a, b) and nad.cert^* as input and outputs nad.cert , which is the deletion certificate for nad.ct . Thus, by using the modification algorithm, we can convert the deletion certificate nad.cert^* of $Z^b X^a(\text{nad.ct}) X^a Z^b$ to the deletion certificate nad.cert of nad.ct . We observe that such a modification algorithm exists for many natural constructions, including our construction.⁹

The formal explanation of the conversion from non-adaptive to adaptive FE is provided in Section 4.4.3.

q -bounded FE for NC^1 circuits. Finally, we explain how to convert 1-bounded one to the q -bounded one.¹⁰ Unfortunately, we do not know how to obtain q -bounded certified everlasting FE for $\mathbf{P/poly}$ 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 $\mathbf{P/poly}$ circuits. For more details, see Section 4.4.4.)

Let us explain how to convert 1-bounded certified everlasting FE for $\mathbf{P/poly}$ 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 $\mathbf{P/poly}$ 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 $\mathbf{P/poly}$ circuits. The formal construction is given in Section 4.4.4.)

4.2.3 Technical Overview: Compute-and-Compare Obfuscation

This section presents an overview of our certified everlasting compute-and-compare obfuscation scheme. A standard compute-and-compare obfuscation takes as input a circuit P , a lock value lock , and a message m and outputs an obfuscated circuit $\tilde{P}$. During evaluation, the message m is recovered from $\tilde{P}$ if the input x satisfies $P(x) = \text{lock}$. In the certified everlasting version, a verification key vk is also generated during obfuscation. A deletion certificate cert can be issued from $\tilde{P}$, and using vk , one can verify its validity. Certified everlasting security ensures that once the validity of the certificate is verified, no information about P , lock , and m remains accessible to the user.

Compute-and-compare obfuscation without a message. We first explain our idea to construct a certified everlasting compute-and-compare obfuscation without any message. More precisely, in a certified everlasting compute-and-compare obfuscation without any message, the evaluation returns 1 if $P(x) = \text{lock}$ holds. Let CC.Obf be the obfuscation algorithm of a standard compute-and-compare obfuscation scheme and Enc, Dec

⁹If the deletion algorithm involves computational-basis measurements followed by Clifford gates, the modification algorithm simply adjusts the Pauli one-time pad, $X^a Z^b$. In fact, all known constructions rely solely on Hadamard basis measurements.

¹⁰ q -bounded means that the adversary can call key queries q times with an a priori bounded polynomial q .

be the encryption, and decryption algorithms of FHE. In our construction, the obfuscated circuit $\tilde{P}$ consists of the following two items:

- $\widetilde{\text{Dec}} := \text{CC.Obf}(\text{Dec}(\text{sk}, \cdot), \text{lock}, 1)$.
- $\{|z_i\rangle_{\theta_i}, \text{ct}_i\}_{i \in [\ell_P]}$ where $\text{ct}_i := \text{Enc}(\theta_i \| \tilde{b}_i)$, $z_i, \theta_i \leftarrow \{0, 1\}^\lambda$ for $i \in [\ell_P]$, $\tilde{b}_i := b_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ and b_i is the i -th bit of the binary string of length ℓ_P representing the circuit P

The verification key vk consists of $(\{z_i, \theta_i\}_{i \in [\ell_P]})$. To evaluate the obfuscated circuit with an input x , we first coherently compute an evaluated FHE ciphertext $|\text{ct}_{U_x(P)}\rangle$ where U_x is a circuit that on input $(\{z_i, \theta_i, \tilde{b}_i\}_{i \in [\ell_P]})$ first recovers b_i , the bits representing P , and then outputs $P(x)$. Then, we coherently evaluate the obfuscated circuit $\widetilde{\text{Dec}}$ with input $|\text{ct}_{U_x(P)}\rangle$ and check that the measured outcome is 1 to decide $P(x) = \text{lock}$. The deletion and verification algorithm works similarly as in the certified everlasting PKE scheme described in Section 4.2.1. 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 lock from an unbounded adversary. More precisely, the obfuscated circuit $\tilde{P}$ includes lock in $\text{CC.Obf}(\text{Dec}(\text{sk}, \cdot), \text{lock}, 1)$ although CC.Obf ensures only computationally security. The unbounded adversary may obtain information of lock from $\text{CC.Obf}(\text{Dec}(\text{sk}, \cdot), \text{lock}, 1)$.

We solve this problem by masking the obfuscated circuit that encodes lock using the XOR function in combination with the BB84 states. More precisely, our obfuscated circuit $\tilde{P}$ consists of the following three items:

- $\widetilde{\text{Dec}} := \text{CC.Obf}(\text{Dec}(\text{sk}, \cdot), \text{lock}, 1)$ where R is dummy lock value $R \leftarrow \{0, 1\}^\lambda$.
- $\{|z_i\rangle_{\theta_i}, \text{ct}_i\}_{i \in [\ell]}$. Here, ct_i encrypts the binary string $b_1 \cdots b_\ell$ representing the circuits $(P \| \tilde{I})$, where $\tilde{I} := \text{CC.Obf}(I, \text{lock}, R)$ and I is an identity circuit such that $I(x) = x$ for all x . More precisely, $\text{ct}_i := \text{Enc}(\theta_i \| \tilde{b}_i)$ for $i \in [\ell_P]$, $\tilde{b}_i := b_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$.

The evaluation algorithm works as before except that the circuit U_x on input $(\{z_i, \theta_i, \tilde{b}_i\}_{i \in [\ell]})$ first reconstructs $(P \| \tilde{I})$ and then outputs the result obtained in the evaluation of $\tilde{I}$ with input $P(x)$. Hence, checking $P(x) = \text{lock}$ is deferred until evaluating $\tilde{I}$. The correctness follows from the fact that U_x returns R if $P(x) = \text{lock}$ and evaluation of $\widetilde{\text{Dec}}$ outputs 1 if $U_x(P \| \tilde{I}) = R$.

To prove the certified everlasting security, we need to construct a simulated circuit $\tilde{\mathcal{P}}_{\text{Sim}}$ which is independent of P and lock that is indistinguishable from the real obfuscated circuit $\tilde{\mathcal{P}}$. The simulated circuit $\tilde{\mathcal{P}}_{\text{Sim}}$ consists of $\widetilde{\text{Dec}} = \text{CC.Obf}(\text{Dec}(\text{sk}, \cdot), R, 1)$ and a set of BB84 states and ciphertexts $\{|z_i\rangle_{\theta_i}, \text{ct}_i\}_{i \in [\ell]}$, where each $\text{ct}_i := \text{Enc}(\theta_i \| \tilde{b}_i)$ and $\tilde{b}_i := 0 \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ for $i \in [\ell]$. Crucially, $\tilde{\mathcal{P}}_{\text{Sim}}$ does not contain any information about the original circuit P or the lock value lock. We rely on the certified everlasting lemma from [BK23] to show that the real obfuscated circuit $\tilde{\mathcal{P}}$ is indistinguishable from the simulated circuit $\tilde{\mathcal{P}}_{\text{Sim}}$ for any unbounded adversary after producing a valid deletion certificate. The certified everlasting lemma only permits flipping one bit at a time, so we use a sequence of ℓ hybrid experiments, where each hybrid modifies a single bit b_i of the obfuscated circuit from 1 to 0. In the i -th hybrid experiment, we replace the bit b_i with 0. To achieve this, we must show that the encryption $\text{Enc}(\theta_i \| \tilde{b}_i)$ is computationally indistinguishable from $\text{Enc}(0 \| \tilde{b}_i)$, enabling the application of the certified everlasting lemma to flip b_i without detection by the adversary. To establish this computational indistinguishability, we first replace the circuit $\tilde{I} \leftarrow \text{CC.Obf}(I, \text{lock}, R)$ with its simulated counterpart $\tilde{I} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_I, 1^{|R|})$. Next, we replace the obfuscated decryption circuit

$\widetilde{\text{Dec}} \leftarrow \text{CC.Obf}(\text{Dec}, \text{lock}, 1)$ with its simulated version $\widetilde{\text{Dec}} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_{\text{Dec}}, 1^1)$, leveraging the security of the underlying compute-and-compare obfuscation scheme. Since the secret key sk is no longer needed to simulate the adversary's view, we can safely change the ciphertexts $\text{Enc}(\theta_i \| \tilde{b}_i)$ to $\text{Enc}(\mathbf{0} \| \tilde{b}_i)$ by invoking the IND-CPA security of the FHE scheme. This allows us to set b_i to 0 using the certified everlasting lemma, completing the proof of indistinguishability.

Compute-and-compare obfuscation with a message. Next, we extend the previous construction into a certified everlasting compute-and-compare obfuscation scheme that obfuscates a circuit P along with a lock value lock and a message $m = m_1 \dots m_n \in \{0, 1\}^n$. The idea is to encrypt the message bits using FHE in combination with BB84 states and recover the message bits during evaluation based on the outcome of the obfuscated circuit $\widetilde{\text{Dec}}$.

The obfuscated circuit $\tilde{P}$ now additionally includes $\{|z_{\ell+k}\rangle_{\theta_{\ell+k}}, \text{ct}_{\ell+k}\}_{k \in [n]}$, where $z_{\ell+k}, \theta_{\ell+k} \leftarrow \{0, 1\}^\lambda$, $\text{ct}_{\ell+k} := \text{Enc}(\theta_{\ell+k} \| \tilde{b}_{\ell+k})$, and $\tilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j: \theta_{\ell+k,j}=0} z_{\ell+k,j}$ for each $k \in [n]$. The evaluation procedure works as before, except that the circuit U_x , given $((\{z_i, \theta_i, \tilde{b}_i\}_{i \in [\ell]}), (z_{\ell+k}, \theta_{\ell+k}, \tilde{b}_{\ell+k}))$, first reconstructs $(P \| \tilde{I})$ from $\{z_i, \theta_i, \tilde{b}_i\}_{i \in [\ell]}$ and recovers m_k from $(z_{\ell+k}, \theta_{\ell+k}, \tilde{b}_{\ell+k})$. It then outputs $m_k \cdot \tilde{I}(P(x))$.

The deletion and verification algorithms are similarly defined as before. The scheme correctly recovers the message m bit by bit. Consider the case where $P(x) = \text{lock}$ and $m_k = 1$. By the definition of U_x and the correctness of the compute-and-compare obfuscation, we have $m_k \cdot \tilde{I}(P(x)) = R$. Consequently, $\widetilde{\text{Dec}}$ evaluates to 1 for an input $\text{ct}_{m_k \cdot \tilde{I}(P(x))}$. If the result of the evaluation is not 1, we set $m_k := 0$.

We prove the certified everlasting security of this scheme using the same approach as for the compute-and-compare obfuscation scheme without a message. The only additional step is to delete the information about the message m using the IND-CPA security of the FHE scheme and the certified everlasting lemma of [BK23] after erasing the information about P and lock .

Certified everlasting predicate encryption. Goyal, Koppula, and Waters [GKW17] and Wichs and Zirdelis [WZ17] presented a generic construction of predicate encryption (PE)¹¹ based on compute-and-compare obfuscation and attribute-based encryption (ABE). Their construction proceeds as follows.

The setup and key generation algorithms follow those of the underlying ABE scheme. Let Enc and Dec denote the encryption and decryption algorithms of ABE. To encrypt a message m with an attribute x , the encryption algorithm selects a random lock $R \in \{0, 1\}^\ell$, computes $\text{ct} := \text{Enc}(x, R)$, and generates $\widetilde{\text{Dec}} := \text{CC.Obf}(\text{Dec}(\cdot, \text{ct}), R, m)$. The ciphertext is the obfuscated circuit $\widetilde{\text{Dec}}$.

Given a secret key sk_P for a policy P , the user evaluates $\widetilde{\text{Dec}}$ with input sk_P to recover the message m . If $P(x) = 1$, the correctness of ABE ensures that $\text{Dec}(\text{sk}_P, \text{ct}) = R$, and thus, $\widetilde{\text{Dec}}(\text{sk}_P)$ outputs m .

One might hope that replacing the compute-and-compare obfuscation and ABE with their certified everlasting counterparts could yield a certified everlasting PE. However, this approach fails because our certified everlasting compute-and-compare obfuscation cannot handle a quantum decryption circuit $\mathcal{D}_{\text{ec}}(\cdot, \text{ct})$ from certified everlasting ABE. The challenge lies in erasing the information about R from the ABE ciphertext ct to apply the certified everlasting security of the compute-and-compare obfuscation. In other words, after a valid certificate of deletion, an unbounded adversary should not be able to distinguish between $\text{Enc}(x, R)$ and $\text{Enc}(x, \mathbf{0})$, something a classical ABE cannot ensure on its own.

We resolve this issue by incorporating BB84 states and certified everlasting compute-and-compare obfuscation into the construction. Specifically, we first sample $z_i, \theta_i \leftarrow \{0, 1\}^\ell$, set $\tilde{r}_i := r_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$,

¹¹This construction satisfies one-sided attribute-hiding security, meaning that both the attribute and message remain hidden to any user without the secret key for decryption.

and compute $ct := \text{Enc}(x, (\theta_1, \dots, \theta_\ell, \tilde{r}_1, \dots, \tilde{r}_\ell))$, where r_i denotes the i -th bit of R . The ciphertext consists of $\widetilde{\text{Dec}} := \text{CCObf}(\text{Dec}(\cdot, ct), R, m)$ 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 vk_{Dec} corresponding to the circuit Dec . The deletion and verification algorithms follow naturally, utilizing the certified everlasting ABE scheme by Bartusek and Khurana non-black-box.

Suppose an adversary queries secret keys sk_P such that $P(x) = 0$ and becomes unbounded after producing a valid certificate of deletion for the ciphertext. Our approach leverages the security of ABE and the certified everlasting lemma of [BK23] to delete information about R . Then, using the certified everlasting security of compute-and-compare obfuscation, we can replace $\widetilde{\text{Dec}}$ with a simulated circuit that does not reveal any information about m or x .

Certified everlasting predicate encryption. Goyal, Koppula and Waters [GKW17] and Wichs and Zirdelis [WZ17] showed a generic construction of PE¹² 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 $ct := \text{Enc}(x, R)$ and $\widetilde{\text{Dec}} := \text{CCObf}(\text{Dec}(\cdot, ct), R, m)$. The ciphertext is the obfuscated circuit $\widetilde{\text{Dec}}$. Given a secret key sk_P for a policy P , a user simply evaluates $\widetilde{\text{Dec}}$ with input sk_P to recover the message m . Note that, if $P(x) = 1$ then by the correctness of ABE, $\text{Dec}(\text{sk}_P, ct) = R$ and hence $\widetilde{\text{Dec}}(\text{sk}_P)$ returns m .

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 $\text{Dec}(\cdot, ct)$ 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 $\text{Enc}(x, R)$ and $\text{Enc}(x, 0)$. 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 $\tilde{r}_i := r_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ and then compute $ct := \text{Enc}(x, (\theta_1, \dots, \theta_\ell, \tilde{r}_1, \dots, \tilde{r}_\ell))$ where r_i denotes the i -th bit of R . The ciphertext consists of $\widetilde{\text{Dec}} := \text{CCObf}(\text{Dec}(\cdot, ct), R, m)$ 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 vk_{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 $\widetilde{\text{Dec}}$ with a simulated circuit that does not contain any information about m, x . The formal security analysis can be found in Section 4.6.2.

¹²It 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.

4.3 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.

4.3.1 Definitions

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

Definition 4.3.1 (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 $(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ 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.*

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

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

$\text{Enc}(\text{MPK}, m) \rightarrow (ct, vk)$: *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 .*

$\text{Dec}(\text{sk}_f, ct) \rightarrow y \text{ or } \perp$: *The decryption algorithm takes sk_f and ct as input, and outputs y or $\perp$.*

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

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

Definition 4.3.2 (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 $f \in \mathcal{F}$,*

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

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

$$\Pr \left[\text{Vrfy}(vk, \text{cert}) \neq \top \mid \begin{array}{l} (\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda) \\ (ct, vk) \leftarrow \text{Enc}(\text{MPK}, m) \\ \text{cert} \leftarrow \text{Del}(ct) \end{array} \right] \leq \text{negl}(\lambda).$$

Remark 4.3.3. In FE, we should be able to run Dec algorithm for many different functions f on the same ciphertext ct . One might think that the quantum ct is destroyed by Dec algorithm, and it can be used only once. However, it is easy to see that Dec algorithm can be always modified so that it does not disturb the quantum state ct by using the gentle measurement lemma [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 4.3.4 (Certified Everlasting Indistinguishable-Security of FE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a functional encryption with certified everlasting deletion for a class $\mathcal{F}$ of functions, plaintext space $\mathcal{M}$. We consider two experiments $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ and $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ 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 $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $(\text{ct}_b, \text{vk}_b) \leftarrow \text{Enc}(\text{MPK}, m_b)$, 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $\text{Vrfy}(\text{vk}_b, \text{cert})$. If the outcome is $\top$ and $f_\ell(m_0) = f_\ell(m_1)$ holds for all key queries f_ℓ , the experiment $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ outputs ρ ; otherwise $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ outputs $\perp$.
8. The challenger sends the outcome of $\text{Vrfy}(\text{vk}_b, \text{cert})$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ outputs b' ; otherwise $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ 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

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

and

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

4.3.2 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 4.3.5 (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

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

For any $\{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}} \in \mathfrak{A}$, consider the following distribution $\{\tilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(b)\}_{\lambda \in \mathbb{N}, b \in \{0,1\}}$ 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 $\perp$.

Then, it holds that

$$\text{TD}(\tilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(0), \tilde{\mathcal{Z}}_\lambda^{\mathcal{A}_\lambda}(1)) \leq \text{negl}(\lambda).$$

Remark 4.3.6. Although the description of the lemma above is slightly different from the original version (see the first item of Remark 4.3.8), 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 4.3.5 to a variant in the interactive game setting as follows.

Lemma 4.3.7 (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 $\text{Expt}_{\mathcal{A},\mathcal{C}}(\lambda, \theta, \beta)$ 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 $\widetilde{\text{Expt}}_{\mathcal{A}',\mathcal{C}}(\lambda, b)$ 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 $\perp$.

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

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

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

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

Remark 4.3.8. 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.¹³
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 $f(m, \text{pub})$ for any public input pub . In the security experiment, we require that a pair of challenge messages (m_0, m_1) must satisfy $f(m_0, \text{pub}) = f(m_1, \text{pub})$ for all key queries f and public inputs pub to prevent trivial attacks. A formal definition is given below.

Definition 4.3.9 (Public-Slot FE (Syntax)). A public-slot functional encryption scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ 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.

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

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

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

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

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

¹³In their notation, the challenger corresponds to $\mathcal{Z}(\theta)$.

Definition 4.3.10 (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 $\text{pub} \in \mathcal{P}$,*

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

Definition 4.3.11 (Security of Public-Slot FE). *Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a public-slot FE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger runs $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$.*
3. *$\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.*
4. *The challenger computes $\text{CT} \leftarrow \text{Enc}(\text{MPK}, m_b)$ 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 $f_\ell(m_0, \text{pub}) = f_\ell(m_1, \text{pub})$ holds for all key queries f_ℓ and public inputs $\text{pub} \in \mathcal{P}$, 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

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

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 A.2.4.

Theorem 4.3.12. *If there exist iO and OWFs, there exists an adaptively indistinguishable-secure public-slot PKFE for $\mathbf{P}/\text{poly}$.*

4.3.3 Collusion-Resistant Construction

Ingredients. We need the following building blocks.

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

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

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

1. Generate $(\text{fe.MPK}, \text{fe.MSK}) \leftarrow \text{FE.Setup}(1^\lambda)$.

2. Output $\text{MPK} := \text{fe.MPK}$ and $\text{MSK} := \text{fe.MSK}$.

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

1. Parse $\text{MSK} = \text{fe.MSK}$.
2. Generate $\text{fe.sk}_{g[f]} \leftarrow \text{FE.KeyGen}(\text{fe.MSK}, g[f])$ where $g[f]$ is a function described in Figure 4.1.
3. Output $\text{sk}_f = \text{fe.sk}_{g[f]}$.

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

1. Parse $\text{MPK} = \text{fe.MPK}$.
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 $v_{i,j,b} \leftarrow \text{PRG}(u_{i,j,b})$ 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

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

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 $j \in [\lambda]$.

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

$\text{CED.Dec}(\text{sk}_f, ct)$:

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

$\text{CED.Del}(ct)$:

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

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

1. Parse $\text{vk} = (U, (z_i, \theta_i)_{i \in [2n+1]})$ and $\text{cert} = (c_{i,j}, d_{i,j})_{i \in [2n+1], j \in [\lambda]}$.
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 $\perp$.

Theorem 4.3.13. *If FE is adaptively indistinguishable-secure public-slot FE for $\mathbf{P}/\text{poly}$ and PRG is a secure PRG, CED is adaptively certified everlasting indistinguishable-secure FE for $\mathbf{P}/\text{poly}$.*

$$\underline{g[f]}$$

Secret Input: $V, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}$

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

Figure 4.1: The description of the function $g[f]$

Decryption Correctness. Let $ct = (\text{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ be an honestly generated ciphertext for a message m and $\text{sk}_f = \text{fe.sk}_{g[f]}$ be an honestly generated decryption key for a function f . Then we have $\text{fe.ct} \in \text{FE.Enc}(\text{fe.MPK}, V \| \theta_1 \| \dots \| \theta_{2n+1} \| \beta_1 \| \dots \| \beta_{2n+1})$ and $|\psi_{i,j}\rangle = |z_{i,j}\rangle |u_{i,j,z_{i,j}}\rangle$ for all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. Since we have $\text{PRG}(u_{i,j,b}) = v_{i,j,b}$ 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 $g[f](V \| \theta_1 \| \dots \| \theta_{2n+1} \| \beta_1 \| \dots \| \beta_{2n+1}, \cdot)$ on $\bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle$ and measure the output, then the resulting outcome is $f(\beta_1 \oplus \bigoplus_{j: \theta_{1,j}=0} z_{1,j} \| \dots \| \beta_n \oplus \bigoplus_{j: \theta_{n,j}=0} z_{n,j}) = f(m)$. Then the decryption correctness follows from that of FE.

Verification Correctness. Let $ct = (\text{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ be an honestly generated ciphertext and $\text{vk} = (U, (z_i, \theta_i)_{i \in [2n+1]})$ 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 $\text{C-Exp}_{\text{CED}, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$. We omit the proof in the main body. See Appendix A.1.

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

Hyb₀: 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 $(\text{fe.MPK}, \text{fe.MSK}) \leftarrow \text{FE.Setup}(1^\lambda)$, sets $\text{MPK} := \text{fe.MPK}$ and $\text{MSK} := \text{fe.MSK}$, 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 $\text{fe.sk}_{g[f]} \leftarrow \text{FE.KeyGen}(\text{fe.MSK}, g[f])$ and returns $\text{sk}_f = \text{fe.sk}_{g[f]}$ to $\mathcal{A}$.

3. $\mathcal{A}$ sends $(m^{(0)}, m^{(1)})$ to the challenger.¹⁴ 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 $(ct, vk) \leftarrow \text{Enc}(\text{MPK}, m^{(0)})$. 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 $v_{i,j,b} \leftarrow \text{PRG}(u_{i,j,b})$ 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

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

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 $\text{fe.ct} \leftarrow \text{FE.Enc}(\text{fe.MPK}, V \parallel \theta_1 \parallel \dots \parallel \theta_{2n+1} \parallel \beta_1 \parallel \dots \parallel \beta_{2n+1})$.
- (f) Set $ct = (\text{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ and $vk = (U, (z_i, \theta_i)_{i \in [2n+1]})$.

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

5. $\mathcal{A}$ sends $\text{cert} = (c_{i,j}, d_{i,j})_{i \in [2n+1], j \in [\lambda]}$ 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.
7. The experiment outputs ρ as a final output.

Hyb_{1,k}: For $k = 0, 1, \dots, n$, this hybrid is identical to **Hyb₀** 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 **Hyb_{1,0}** is identical to **Hyb₀**.

Hyb₂: This hybrid is identical to **Hyb_{1,n}** except that β_{2n+1} is flipped. That is, β_i is generated as follows.

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

¹⁴We use $(m^{(0)}, m^{(1)})$ instead of (m_0, m_1) to denote a pair of challenge messages to avoid a notational collision.

Hyb_{3,k}: For $k = 0, 1, \dots, n$, this hybrid is identical to **Hyb₂** 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 **Hyb_{3,0}** is identical to **Hyb₂**.

Hyb₄: This hybrid is identical to **Hyb_{3,n}** except that β_{2n+1} is flipped. That is, β_i is generated as follows.

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

Hyb_{5,k}: For $k = 0, 1, \dots, n$, this hybrid is identical to **Hyb₄** 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 **Hyb_{5,0}** is identical to **Hyb₄**.

Remark that **Hyb_{5,n}** is exactly the everlasting adaptive experiment where the challenge bit is set to be 1. Thus, we have to prove

$$\text{TD}(\text{Hyb}_0, \text{Hyb}_{5,n}) \leq \text{negl}(\lambda). \quad (4.1)$$

We prove this by the following lemmata.

Lemma 4.3.14. *If FE is adaptively secure and PRG is a secure PRG, for any $k \in [n]$,*

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

Lemma 4.3.15. *If FE is adaptively secure and PRG is a secure PRG,*

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

Lemma 4.3.16. *If FE is adaptively secure and PRG is a secure PRG, for any $k \in [n]$,*

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

Lemma 4.3.17. *If FE is adaptively secure and PRG is a secure PRG,*

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

Lemma 4.3.18. *If FE is adaptively secure and PRG is a secure PRG, for any $k \in [n]$,*

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

Noting that $\text{Hyb}_{1,0}$, $\text{Hyb}_{3,0}$, and $\text{Hyb}_{5,0}$ are identical to Hyb_0 , Hyb_2 , and Hyb_4 , respectively, Lemmata 4.3.14 to 4.3.18 imply Equation (4.1).

What is left is to prove these lemmata. Actually, the proofs of these lemmata are very similar. We give a full proof of Lemma 4.3.14 below. After that, we also explain how to modify it to prove Lemma 4.3.15. The proofs of Lemmata 4.3.16 and 4.3.18 are almost identical to that of Lemma 4.3.14 and the proof of Lemma 4.3.17 is almost identical to that of Lemma 4.3.15, and thus we omit them.

Proof of Lemma 4.3.14. First, we prove Lemma 4.3.14 below.

Proof. For applying Lemma 4.3.7, we consider the following experiment $\text{Expt}_{\mathcal{B}, \mathcal{C}}^{1,k}(\lambda, \theta, \beta)$ between a QPT adversary $\mathcal{B}$ and a challenger $\mathcal{C}$ for $\theta \in \{0, 1\}^\lambda$ and $\beta \in \{0, 1\}$ as follows:

$\text{Expt}_{\mathcal{B}, \mathcal{C}}^{1,k}(\lambda, \theta, \beta)$: In this experiment, $\mathcal{B}$ and $\mathcal{C}$ play the roles of $\mathcal{A}$ and the challenger of $\text{Hyb}_{1,k}$ with the differences that $\mathcal{C}$ sets $\beta_{n+k} := \beta$ and $\theta_{n+k} := \theta$, $\mathcal{C}$ does not generate $|\psi_{n+k,j}\rangle$ 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 $(\text{fe.MPK}, \text{fe.MSK}) \leftarrow \text{FE.Setup}(1^\lambda)$, sets $\text{MPK} := \text{fe.MPK}$ and $\text{MSK} := \text{fe.MSK}$, 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 $\text{fe.sk}_{g[f]} \leftarrow \text{FE.KeyGen}(\text{fe.MSK}, g[f])$ and returns $\text{sk}_f = \text{fe.sk}_{g[f]}$ 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 $v_{i,j,b} \leftarrow \text{PRG}(u_{i,j,b})$ 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

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

where $\theta_{i,j}$ (resp. $z_{i,j}$) is the j -th bit of θ_i (resp. z_i) for every $i \in [2n+1] \setminus \{n+k\}$ 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 $\text{fe.ct} \leftarrow \text{FE.Enc}(\text{fe.MPK}, V\|\theta_1\| \dots \|\theta_{2n+1}\|\beta_1\| \dots \|\beta_{2n+1}\|)$.
 $\mathcal{C}$ sends $(\text{fe.ct}, \bigotimes_{i \in [2n+1] \setminus \{n+k\}, j \in [\lambda]} |\psi_{i,j}\rangle, \{u_{n+k,j,b}\}_{j \in [\lambda], b \in \{0,1\}})$ to $\mathcal{B}$.
 5. $\mathcal{B}$ outputs a bit b' as a final output of the experiment.

We prove the following lemma.

Lemma 4.3.19. *For any QPT $\mathcal{B}$,*

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

Before proving Lemma 4.3.19, we complete the proof of Lemma 4.3.14 assuming that Lemma 4.3.19 is true. By Lemmata 4.3.7 and 4.3.19, for any QPT adversary $\mathcal{B}'$, we have

$$\text{TD}(\widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 0), \widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 1)) \leq \text{negl}(\lambda). \quad (4.2)$$

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

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

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 $\text{Expt}_{\mathcal{B},\mathcal{C}}^{1,k}(\lambda, \theta, b \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j})$ 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 $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$ is that β_{n+k} is set to be $0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\text{Hyb}_{1,k-1}$ and $m_k^{(1)} \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\text{Hyb}_{1,k}$. 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 $\widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 0)$ and $\widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 1)$ using $\mathcal{A}$ that distinguishes $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$ as follows.

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

1. $\mathcal{B}'$ plays the role of $\mathcal{A}$ in $\text{Hyb}_{1,k}$ where the external challenger $\mathcal{C}$ of $\widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, b)$ is used to simulate the challenger of $\text{Hyb}_{1,k}$. $\mathcal{C}$ 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 $|b\rangle \rightarrow |b\rangle |u_{n+k,j,b}\rangle$ 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 $z' := z'_{n+k,1} \parallel \dots \parallel z'_{n+k,\lambda}$ and $\mathcal{A}$'s internal state ρ .

We can see that $\mathcal{B}'$ perfectly simulates $\text{Hyb}_{1,k-1}$ if $b = 0$ and $\text{Hyb}_{1,k}$ 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 $\widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, b)$) 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 $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$). Therefore, we must have

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

Combined with Equation (4.2), this completes the proof of Lemma 4.3.14. $\square$

Now, we are left to prove Lemma 4.3.19.

Proof of Lemma 4.3.19. We further consider the following sequence of hybrids:

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

$\text{Expt}_{\mathcal{B},C}^{1,k,b}(\lambda, \theta, \beta)$: This is identical to $\text{Expt}_{\mathcal{B},C}^{1,k,a}(\lambda, \theta, \beta)$ 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 $\text{Expt}_{\mathcal{B},C}^{1,k,a}(\lambda, \theta, \beta)$.

Proposition 4.3.20. *If PRG is a secure PRG,*

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

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 $\text{Expt}_{\mathcal{B},C}^{1,k}(\lambda, \theta, \beta)$, Proposition 4.3.20 directly follows from the security of PRG. $\square$

Proposition 4.3.21. *If FE is adaptively secure,*

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

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 $\text{PRG}(u) = v_{i,j,1 \oplus z_{i,j}}$ 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 $\Pr[\text{Good}] \geq 1 - 2n\lambda 2^{-\lambda} = 1 - \text{negl}(\lambda)$. We prove that whenever Good occurs, we have

$$\begin{aligned} & g[f]((V, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}) \\ &= g[f]((V, \theta_1, \dots, \theta_{n+k-1}, 0^n, \theta_{n+k+1}, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}) \end{aligned} \quad (4.3)$$

for all key queries f and $(b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}$. If this is proven, Proposition 4.3.21 directly follows from the adaptive security of FE.

Below, we prove Equation (4.3). We consider the following two cases.

- If $\text{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ 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 (4.3) are equal to $f(m^{(0)})$.

- Otherwise, both sides of Equation (4.3) are equal to $\perp$.

In either case, Equation (4.3) holds. This completes the proof of Proposition 4.3.21. $\square$

Proposition 4.3.22. *If PRG is a secure PRG,*

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

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 $\text{Expt}_{\mathcal{B},C}^{1,k}(\lambda, 0^n, \beta)$, Proposition 4.3.22 directly follows from the security of PRG. $\square$

Lemma 4.3.19 follows from the above propositions. $\square$

Proof of Lemma 4.3.15. Next, we prove Lemma 4.3.15.

Proof. Since the proof of Lemma 4.3.15 is very similar to that of Lemma 4.3.14, we only explain the difference. First, we define an experiment $\text{Expt}_{\mathcal{B},C}^2(\lambda, \theta, \beta)$ that is similar to $\text{Expt}_{\mathcal{B},C}^{1,k}(\lambda, \theta, \beta)$ except that $n+k$ is replaced with $2n+1$. Then by almost the same argument as that in the proof of Lemma 4.3.14 using Lemma 4.3.7, we only have to prove

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

for all QPT $\mathcal{B}$. Its proof is also similar to that of Lemma 4.3.19. We define $\text{Expt}_{\mathcal{B},C}^{2,a}(\lambda, \theta, \beta)$ and $\text{Expt}_{\mathcal{B},C}^{2,b}(\lambda, \theta, \beta)$ similarly to $\text{Expt}_{\mathcal{B},C}^{1,k,a}(\lambda, \theta, \beta)$ and $\text{Expt}_{\mathcal{B},C}^{1,k,b}(\lambda, \theta, \beta)$ except that $n+k$ is replaced with $2n+1$. Then the computational indistinguishability between $\text{Expt}_{\mathcal{B},C}^2(\lambda, \theta, \beta)$ and $\text{Expt}_{\mathcal{B},C}^{2,a}(\lambda, \theta, \beta)$ and between $\text{Expt}_{\mathcal{B},C}^{2,b}(\lambda, \theta, \beta)$ and $\text{Expt}_{\mathcal{B},C}^2(\lambda, 0^n, \beta)$ immediately follow from the security of PRG. We argue the computational indistinguishability between $\text{Expt}_{\mathcal{B},C}^{2,a}(\lambda, \theta, \beta)$ and $\text{Expt}_{\mathcal{B},C}^{2,b}(\lambda, \theta, \beta)$ based on the security of FE as follows.

Let Good be the event that there is no u such that $\text{PRG}(u) = v_{i,j,1 \oplus z_{i,j}}$ for all $i \in [2n]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$. We have $\Pr[\text{Good}] \geq 1 - \text{negl}(\lambda)$. Similarly to the proof of Proposition 4.3.21, it suffices to prove that whenever Good occurs, we have

$$\begin{aligned} g[f]((V, \theta_1, \dots, \theta_{2n+1}, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}) \\ = g[f]((V, \theta_1, \dots, \theta_{2n}, 0^\lambda, \beta_1, \dots, \beta_{2n+1}), (b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}) \end{aligned} \quad (4.4)$$

for all key queries f and $(b_{i,j}, u_{i,j})_{i \in [2n+1], j \in [\lambda]}$. Below, we prove Equation (4.4). We consider the following two cases.

- If $\text{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ 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 (4.4) 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 (4.4) 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 (4.4) are equal to $f(m^{(0)}) = f(m^{(1)})$.

- Otherwise, both sides of Equation (4.4) are equal to $\perp$.

In either case, Equation (4.4) holds. This completes the proof of Lemma 4.3.15. $\square$

4.4 Bounded Collusion-Resistant Functional Encryption with Certified Everlasting Deletion

4.4.1 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 4.4.3.

Definition 4.4.1 (Verification Correctness with QOTP). *There exists a negligible function negl and a PPT algorithm Recover such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}^*) = \perp \mid \begin{array}{l} (\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, 1^q) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{MPK}, m) \\ a, b \leftarrow \{0, 1\}^{p(\lambda)} \\ \widetilde{\text{cert}} \leftarrow \text{Del}(Z^b X^a \text{ct} X^a Z^b) \\ \text{cert}^* \leftarrow \text{Recover}(a, b, \widetilde{\text{cert}}) \end{array} \right] \leq \text{negl}(\lambda).$$

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

1. *The challenger runs $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, 1^q)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$ 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 $\mathcal{V} := \{y_i := f_i(m), f_i, \text{sk}_{f_i}\}_{i \in [q_{\text{pre}}]}$.*
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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{MPK}, m)$, and sends ct to $\mathcal{A}_1$.*
 - *If $b = 1$, the challenger computes $(\text{ct}, \text{st}_{q_{\text{pre}}}) \leftarrow \text{Sim}_1(\text{MPK}, \mathcal{V}, 1^{|m|})$, and sends ct to $\mathcal{A}_1$, where $\text{st}_{q_{\text{pre}}}$ is a quantum state.*

5. $\mathcal{A}_1$ is allowed to make arbitrary key queries at most $q - q_{\text{pre}}$ times. For the ℓ -th key query, the challenger works as follows.
 - If $b = 0$, the challenger receives $f_\ell \in \mathcal{F}$, computes $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}_1$.
 - If $b = 1$, the challenger receives $f_\ell \in \mathcal{F}$, computes $(\text{sk}_{f_\ell}, \text{st}_\ell) \leftarrow \text{Sim}_2(\text{MSK}, f_\ell, f_\ell(m), \text{st}_{\ell-1})$, and sends sk_{f_ℓ} to $\mathcal{A}_1$, where st_ℓ is a quantum state.
6. If $b = 1$, the challenger runs $\text{vk} \leftarrow \text{Sim}_3(\text{st}_q)$.
7. At some point, $\mathcal{A}_1$ sends cert to the challenger and its internal state to $\mathcal{A}_2$.
8. The challenger computes $\text{Vrfy}(\text{vk}, \text{cert})$. If the output is $\top$, then the challenger outputs $\top$, and sends MSK to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ 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 $\perp$.

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

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

Remark 4.4.3. Note that Definitions 4.4.2 and 4.4.4 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 4.4.2 and 4.4.4 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 4.4.4 (q -Bounded Non-Adaptive Certified Everlasting Simulation-Security). Let q be a polynomial of λ . Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a q -bounded FE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-noada-sim}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let Sim be a QPT algorithm.

1. The challenger runs $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$ 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 $\mathcal{V} := \{y_i := f_i(m), f_i, \text{sk}_{f_i}\}_{i \in [q]}$.
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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{MPK}, m)$, and sends ct to $\mathcal{A}_1$.
 - If $b = 1$, the challenger computes $(\text{vk}, \text{ct}) \leftarrow \text{Sim}(\text{MPK}, \mathcal{V}, 1^{|m|})$, 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 $\text{Vrfy}(\text{vk}, \text{cert})$. If the output is $\top$, then the challenger outputs $\top$, and sends MSK to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ 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 $\perp$.

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

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

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

1. The challenger runs $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, 1^q)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$. Let q be the total number of times that $\mathcal{A}$ makes key queries. Let $\mathcal{V} := \{y_i := f_i(m), f_i, \text{sk}_{f_i}\}_{i \in [q]}$.
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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{MPK}, m)$, and sends ct to $\mathcal{A}$.
 - If $b = 1$, the challenger computes $\text{ct} \leftarrow \text{Sim}(\text{MPK}, \mathcal{V}, 1^{|m|})$, and sends ct to $\mathcal{A}$.
5. $\mathcal{A}$ outputs $b' \in \{0, 1\}$. The output of the experiment is b' .

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

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

Definition 4.4.6 (q -Bounded Adaptive Simulation-Security for FE with Certified Everlasting Deletion [GVW12]). Let q be a polynomial of λ . Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a q -bounded FE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ada-sim}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$. Let Sim_1 and Sim_2 be a QPT algorithm.

1. The challenger runs $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda, 1^q)$ 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $\mathcal{V} := \{y_i := f_i(m), f_i, \text{sk}_{f_i}\}_{i \in [q_{\text{pre}}]}$.

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

5. $\mathcal{A}$ is allowed to make arbitrary key queries at most $(q - q_{\text{pre}})$ times. For the ℓ -th key query, the challenger works as follows:

- If $b = 0$, the challenger receives $f_\ell \in \mathcal{F}$, computes $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, and sends sk_{f_ℓ} to $\mathcal{A}$.
- If $b = 1$, the challenger receives $f_\ell \in \mathcal{F}$, computes $(\text{sk}_{f_\ell}, \text{st}_\ell) \leftarrow \text{Sim}_2(\text{MSK}, f_\ell, f_\ell(m), \text{st}_{\ell-1})$, 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 $\text{Sim} = (\text{Sim}_1, \text{Sim}_2)$ such that for any QPT adversary $\mathcal{A}$ it holds that

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

4.4.2 1-Bounded Construction with Non-Adaptive Security

To achieve q -bounded adaptive certified everlasting simulation-secure FE in Section 4.4.4, we prepare building blocks in this section and Section 4.4.3. In this section, we construct a 1-bounded non-adaptive certified everlasting simulation-secure FE scheme from a certified everlasting secure garbling scheme (Definition A.5.1) and a certified everlasting secure PKE scheme (Definition A.3.8). See Appendices A.3.4, A.3.5 and A.5.2 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 $\Sigma_{\text{cefe}} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ from a certified everlasting secure garbling scheme $\Sigma_{\text{cegc}} = \text{GC}(\text{Setup}, \text{Garble}, \text{Eval}, \text{Del}, \text{Vrfy})$ (Definition A.5.1) and a certified everlasting secure PKE scheme $\Sigma_{\text{cepk}} = \text{PKE}(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ (Definition A.3.8).

Setup(1^λ):

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

KeyGen(MSK, f):

- Parse $\text{MSK} = \{\text{pke.sk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0, 1\}}$ and $f = (f_1, \dots, f_s)$.

- Output $sk_f := (f, \{pke.sk_{i,f[i]}\}_{i \in [s]})$.

$Enc(MPK, m)$:

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

$Dec(sk_f, ct)$:

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

$Del(ct)$:

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

$Vrfy(vk, cert)$:

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

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

Security: The following two theorems hold.

Theorem 4.4.7. *If Σ_{cegc} satisfies the selective security (Definition A.5.4) and Σ_{cepk} satisfies the IND-CPA security (Definition A.3.12), Σ_{cefe} satisfies the 1-bounded non-adaptive simulation-security (Definition 4.4.5).*

Its proof is similar to that of Theorem 4.4.8, and therefore we omit it.

Theorem 4.4.8. *If Σ_{cegc} satisfies the selective certified everlasting security (Definition A.5.5) and Σ_{cepk} satisfies the certified everlasting IND-CPA security (Definition A.3.13), Σ_{cefe} satisfies the 1-bounded non-adaptive certified everlasting simulation-security (Definition 4.4.4).*

Proof of Theorem 4.4.8. Let us describe how the simulator Sim works.

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

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

Let us define the sequence of hybrids as follows.

Hyb₀: This is identical to $\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-non-adapt}}(\lambda, 0)$.

1. The challenger generates $(\text{pke.pk}_{i,\alpha}, \text{pke.sk}_{i,\alpha}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\text{pke.pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ 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 $(f, \{\text{pke.sk}_{i,f[i]}\}_{i \in [s]})$ to $\mathcal{A}_1$.
3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to the challenger.
4. The challenger computes $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$, $(\tilde{u}, \text{gc.vk}) \leftarrow \text{GC.Garble}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$, and $(\text{pke.vk}_{i,\alpha}, \text{pke.ct}_{i,\alpha}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,\alpha}, L_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $(\tilde{u}, \{\text{pke.ct}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ sends $(\text{gc.cert}, \{\text{pke.cert}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
6. If $\top \leftarrow \text{GC.Vrfy}(\text{gc.vk}, \text{gc.cert})$, and $\top \leftarrow \text{PKE.Vrfy}(\text{pke.vk}_{i,\alpha}, \text{pke.cert}_{i,\alpha})$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, the challenger outputs $\top$, and sends $\{\text{pke.sk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs b' . If the challenger outputs $\top$, the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

Hyb₁: This is identical to **Hyb₀** 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 $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$ in step 2 regardless of whether a key query is called or not. Third, the challenger does not compute $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$ in step 4. Fourth, the challenger computes $(\text{pke.vk}_{i,f[i] \oplus 1}, \text{pke.ct}_{i,f[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,f[i] \oplus 1}, L_{i,f[i] \oplus 1}^*)$ for every $i \in [s]$ instead of computing $(\text{pke.vk}_{i,f[i] \oplus 1}, \text{pke.ct}_{i,f[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,f[i] \oplus 1}, L_{i,f[i] \oplus 1})$ for every $i \in [s]$.

Hyb₂: This is identical to **Hyb₁** except for the following point. The challenger computes $(\tilde{u}, \text{gc.vk}) \leftarrow \text{GC.Sim}(1^\lambda, 1^{|f|}, U(f, m), \{L_{i,f[i]}\}_{i \in [s]})$ instead of computing $(\tilde{u}, \text{gc.vk}) \leftarrow \text{GC.Garble}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$.

From the definition of $\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-non-adapt}}(\lambda, b)$ and Sim , it is clear that $\Pr[\text{Hyb}_0 = 1] =$

$\Pr[\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-non-adapt}}(\lambda, 0) = 1]$ and $\Pr[\text{Hyb}_2 = 1] = \Pr[\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-non-adapt}}(\lambda, 1) = 1]$. Therefore, Theorem 4.4.8 easily follows from the following Propositions 4.4.9 and 4.4.10. (whose proof is given later.) $\square$

Proposition 4.4.9. *If Σ_{cepk} satisfies the certified everlasting IND-CPA security,*

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

Proposition 4.4.10. *If Σ_{cegc} satisfies the certified everlasting selective security,*

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

Proof of Proposition 4.4.9. For the proof, we use Lemma A.4.9 whose statement and proof is given in Appendix A.4.2. We assume that $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$ defined in Lemma A.4.9. This contradicts the certified everlasting IND-CPA of Σ_{cepk} from Lemma A.4.9. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $\{\text{pke.pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ from the challenger of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$, and sends $\{\text{pke.pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ 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 $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$. 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 $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$.
3. $\mathcal{B}$ sends $(f, L_{1,f[1] \oplus 1}, L_{2,f[2] \oplus 1}, \dots, L_{s,f[s] \oplus 1}, L_{1,f[1] \oplus 1}^*, L_{2,f[2] \oplus 1}^*, \dots, L_{s,f[s] \oplus 1}^*)$ to the challenger of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$.
4. $\mathcal{B}$ receives $(\{\text{pke.sk}_{i,f[i]}\}_{i \in [s]}, \{\text{pke.ct}_{i,f[i] \oplus 1}\}_{i \in [s]})$ from the challenger. If a key query is called, $\mathcal{B}$ sends $(f, \{\text{pke.sk}_{i,f[i]}\}_{i \in [s]})$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$, and sends m to $\mathcal{B}$.
6. $\mathcal{B}$ computes $(\tilde{\mathcal{U}}, \text{gc.vk}) \leftarrow \text{GC.Garble}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ and $(\text{pke.vk}_{i,f[i]}, \text{pke.ct}_{i,f[i]}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,f[i]}, L_{i,f[i]})$ for every $i \in [s]$, and sends $(\tilde{\mathcal{U}}, \{\text{pke.ct}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to $\mathcal{A}_1$.
7. $\mathcal{A}_1$ sends $(\text{gc.cert}, \{\text{pke.cert}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to $\mathcal{B}$, and sends its internal state to $\mathcal{A}_2$.
8. $\mathcal{B}$ sends $\{\text{pke.cert}_{i,f[i] \oplus 1}\}_{i \in [s]}$ to the challenger, and receives $\{\text{pke.sk}_{i,f[i] \oplus 1}\}_{i \in [s]}$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.
9. $\mathcal{B}$ sends $\{\text{pke.sk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ to $\mathcal{A}_2$.
10. $\mathcal{A}_2$ outputs b' .
11. $\mathcal{B}$ computes GC.Vrfy for gc.cert and PKE.Vrfy for all $\{\text{pke.cert}_{i,f[i]}\}_{i \in [s]}$, and outputs b' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_1 = 1]$. By assumption, $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ 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 A.4.9. $\square$

Proof of Proposition 4.4.10. We assume that $|\Pr[\text{Hyb}_1 = 1] - \Pr[\text{Hyb}_2 = 1]|$ 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 $(\text{pke.pk}_{i,\alpha}, \text{pke.sk}_{i,\alpha}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\text{pke.pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ 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 $(f, \{\text{pke.sk}_{i,f[i]}\}_{i \in [s]})$ 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 $\text{Exp}_{\mathcal{B}, \Sigma_{\text{cegc}}}^{\text{cert-ever-sel-gbl}}(1^\lambda, b)$.
5. The challenger computes $\{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}} \leftarrow \text{GC.Setup}(1^\lambda)$ and does the following:
 - If $b = 0$, the challenger computes $(\tilde{U}, \text{gc.vk}) \leftarrow \text{GC.Garble}(1^\lambda, U(\cdot, m), \{L_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$, and sends $(\tilde{U}, \{L_{i,f[i]}\}_{i \in [s]})$ to $\mathcal{B}$.
 - If $b = 1$, the challenger computes $(\tilde{U}, \text{gc.vk}) \leftarrow \text{GC.Sim}(1^\lambda, 1^{|f|}, U(f, m), \{L_{i,f[i]}\}_{i \in [s]})$, and sends $(\tilde{U}, \{L_{i,f[i]}\}_{i \in [s]})$ to $\mathcal{B}$.
6. $\mathcal{B}$ computes $(\text{pke.vk}_{i,f[i]}, \text{pke.ct}_{i,f[i]}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,f[i]}, L_{i,f[i]})$ and $(\text{pke.vk}_{i,f[i] \oplus 1}, \text{pke.ct}_{i,f[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,f[i] \oplus 1}, L_{i,f[i] \oplus 1}^*)$ for every $i \in [s]$.
7. $\mathcal{B}$ sends $(\tilde{U}, \{\text{pke.ct}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to $\mathcal{A}_1$.
8. $\mathcal{A}_1$ sends $(\text{gc.cert}, \{\text{pke.cert}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}})$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
9. $\mathcal{B}$ sends gc.cert to the challenger, and receives $\top$ or $\perp$ from the challenger. If $\mathcal{B}$ receives $\perp$ from the challenger, it outputs $\perp$ and aborts.
10. $\mathcal{B}$ sends $\{\text{pke.sk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ to $\mathcal{A}_2$.
11. $\mathcal{A}_2$ outputs b' .
12. $\mathcal{B}$ computes PKE.Vrfy for all $\text{pke.cert}_{i,\alpha}$, and outputs b' if all results are $\top$. Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_1 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_2 = 1]$. By assumption, $|\Pr[\text{Hyb}_1 = 1] - \Pr[\text{Hyb}_2 = 1]|$ 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} . $\square$

4.4.3 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 A.4.1). See Appendix A.4.2 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 $\Sigma_{\text{cefe}} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ from a 1-bounded non-adaptive certified everlasting secure FE scheme $\Sigma_{\text{nad}} = \text{NAD}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$, where the ciphertext space is $\mathcal{C} := \mathcal{Q}^{\otimes n}$, and a certified everlasting secure RNCE scheme $\Sigma_{\text{cence}} = \text{NCE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal}, \text{Del}, \text{Vrfy})$ (Definition A.4.1). Let NAD.Recover be a QPT algorithm such that

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

for any m .

Our construction is as follows.

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

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

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

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

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

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

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

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

$\text{Del}(\text{ct})$:

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

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

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

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

Security: The following two theorems hold.

Theorem 4.4.11. *If Σ_{nad} satisfies the 1-bounded non-adaptive simulation-security (Definition 4.4.5) and Σ_{cence} satisfies the RNC security (Definition A.4.3), Σ_{cefe} satisfies the 1-bounded adaptive simulation-security (Definition 4.4.6).*

Its proof is similar to that of Theorem 4.4.12, and therefore we omit it.

Theorem 4.4.12. *If Σ_{nad} satisfies the 1-bounded non-adaptive certified everlasting simulation-security (Definition 4.4.4) and Σ_{cence} satisfies the certified everlasting RNC security (Definition A.4.4), Σ_{cefe} satisfies the 1-bounded adaptive certified everlasting simulation-security (Definition 4.4.2).*

Proof of Theorem 4.4.12. 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 NAD.Sim be the simulating algorithm of the ciphertext nad.ct . Let us describe how the simulator $\text{Sim} = (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$ works below.

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

1. Parse $\text{MPK} = (\text{nad.MPK}, \text{nce.pk})$ and $\mathcal{V} = (f, f(m), (\text{nad.sk}_f, \text{nce.sk}))$ or $\emptyset$.¹⁵
2. Sim_1 does the following:
 - If $\mathcal{V} = \emptyset$, generate $|\widetilde{0^n 0^n}\rangle$ and $(\text{nce.vk}, \widetilde{\text{nce.ct}}, \text{nce.aux}) \leftarrow \text{NCE.Fake}(\text{nce.pk})$. Let $\Psi_A := \text{Tr}_B(|\widetilde{0^n 0^n}\rangle \langle \widetilde{0^n 0^n}|)$ and $\Psi_B := \text{Tr}_A(|\widetilde{0^n 0^n}\rangle \langle \widetilde{0^n 0^n}|)$. Output $ct := (\Psi_A, \widetilde{\text{nce.ct}})$ and $st := (\text{nce.aux}, \text{nce.pk}, \text{nad.MPK}, \Psi_B, 1^{|m|}, \text{nce.vk}, 0)$.
 - If $\mathcal{V} = (f, f(m), (\text{nad.sk}_f, \text{nce.sk}))$, generate $a, c \leftarrow \{0, 1\}^n$, $(\text{nce.vk}, \text{nce.ct}) \leftarrow \text{NCE.Enc}(\text{nce.pk}, (a, c))$, $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Sim}(\text{nad.MPK}, (f, f(m), \text{nad.sk}_f), 1^{|m|})$ and $\Psi := Z^c X^a \text{nad.ct} X^a Z^c$. Output $ct := (\Psi, \text{nce.ct})$ and $st := (\text{nad.vk}, \text{nce.vk}, a, c, 1)$.

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

¹⁵If an adversary calls a key query before the adversary receives a challenge ciphertext, then $\mathcal{V} = (f, f(m), (\text{nad.sk}_f, \text{nce.sk}))$. Otherwise, $\mathcal{V} = \emptyset$.

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

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

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

Let us define the sequence of hybrids as follows.

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

1. The challenger generates $(\text{nad.MPK}, \text{nad.MSK}) \leftarrow \text{NAD.Setup}(1^\lambda)$ and $(\text{nce.pk}, \text{nce.MSK}) \leftarrow \text{NCE.Setup}(1^\lambda)$, and sends $(\text{nad.MPK}, \text{nce.pk})$ 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 $\text{nad.sk}_f \leftarrow \text{NAD.KeyGen}(\text{nad.MSK}, f)$ and $\text{nce.sk} \leftarrow \text{NCE.KeyGen}(\text{nce.MSK})$, and sends $(\text{nad.sk}_f, \text{nce.sk})$ 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 $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$, $\Psi := Z^c X^a \text{nad.ct} X^a Z^c$ and $(\text{nce.vk}, \text{nce.ct}) \leftarrow \text{NCE.Enc}(\text{nce.pk}, (a, c))$, and sends $(\Psi, \text{nce.ct})$ 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 $\text{nad.sk}_f \leftarrow \text{NAD.KeyGen}(\text{nad.MSK}, f)$ and $\text{nce.sk} \leftarrow \text{NCE.KeyGen}(\text{nce.MSK})$, and sends $(\text{nad.sk}_f, \text{nce.sk})$ to $\mathcal{A}_1$.
6. $\mathcal{A}_1$ sends $(\text{nad.cert}, \text{nce.cert})$ to the challenger and its internal state to $\mathcal{A}_2$.
7. The challenger computes $\text{nad.cert}^* \leftarrow \text{NAD.Recover}(a, c, \text{nad.cert})$. The challenger computes $\text{NCE.Vrfy}(\text{nce.vk}, \text{nce.cert})$ and $\text{NAD.Vrfy}(\text{nad.vk}, \text{nad.cert}^*)$. If the results are $\top$, the challenger outputs $\top$ and sends $(\text{nad.MSK}, \text{nce.MSK})$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$ and sends $\perp$ 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 $\perp$.

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 $(\text{nce.vk}, \widetilde{\text{nce.ct}}, \text{nce.aux}) \leftarrow \text{NCE.Fake}(\text{nce.pk})$ and sends $(\Psi, \widetilde{\text{nce.ct}})$

to $\mathcal{A}_1$ instead of computing $(\text{nce.vk}, \text{nce.ct}) \leftarrow \text{NCE.Enc}(\text{nce.pk}, (a, c))$ and sending $(\Psi, \text{nce.ct})$ to $\mathcal{A}_1$. Second, in step 5, the challenger computes $\text{nce.sk} \leftarrow \text{NCE.Reveal}(\text{nce.pk}, \text{nce.MSK}, \text{nce.aux}, (a, c))$ and sends $(\text{nad.sk}_f, \text{nce.sk})$ to $\mathcal{A}_1$ instead of computing $\text{nce.sk} \leftarrow \text{NCE.KeyGen}(\text{nce.MSK})$ and sending $(\text{nad.sk}_f, \text{nce.sk})$ to $\mathcal{A}_1$.

Hyb₂: This is different from Hyb₁ in the following three points. First, when a key query is not called in step 2, the challenger generates $|0^n 0^n\rangle$ instead of generating $a, c \leftarrow \{0, 1\}^n$ and $\Psi = Z^c X^a \text{nad.ct} X^a Z^c$. Let $\Psi_A := \text{Tr}_B(|0^n 0^n\rangle\langle 0^n 0^n|)$ and $\Psi_B := \text{Tr}_A(|0^n 0^n\rangle\langle 0^n 0^n|)$. Second, when a key query is not called in step 2, the challenger sends $(\Psi_A, \text{nce.ct})$ to $\mathcal{A}_1$ instead of sending $(\Psi, \text{nce.ct})$ to $\mathcal{A}_1$ and then that measures the i -th qubit of 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 $\text{nce.sk} \leftarrow \text{NCE.Reveal}(\text{nce.pk}, \text{nce.MSK}, \text{nce.aux}, (x, z))$ instead of computing $\text{nce.sk} \leftarrow \text{NCE.Reveal}(\text{nce.pk}, \text{nce.MSK}, \text{nce.aux}, (a, c))$ in step 5 and computes $\text{nad.cert}^* \leftarrow \text{NAD.Recover}(x, z, \text{nad.cert})$ instead of computing $\text{nad.cert}^* \leftarrow \text{NAD.Recover}(a, c, \text{nad.cert})$ in step 7.

Hyb₃: This is different from Hyb₂ in the following three points. First, when a key query is not called in step 2, the challenger does not generate $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$ and measure the i -th qubit of 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 $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$ and measures the i -th qubit of nad.ct and Ψ_B in the Bell basis for all $i \in [n]$ after it computes $\text{nad.sk}_f \leftarrow \text{NAD.KeyGen}(\text{nad.MSK}, f)$. Third, if a key query is not called throughout the experiment, the challenger computes $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$, measures the i -th qubit of nad.ct and Ψ_B in the Bell basis after step 5.

Hyb₄: This is identical to Hyb₃ except that the challenger computes $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Sim}(\text{nad.MPK}, \mathcal{V}, 1^{|m|})$ instead of computing $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$, where $\mathcal{V} = (f, f(m), \text{nad.sk}_f)$ if a key query is called and $\mathcal{V} = \emptyset$ if a key query is not called.

From the definition of $\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, b)$ and $\text{Sim} = (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$, it is clear that $\Pr[\text{Hyb}_0 = 1] = \Pr[\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, 0) = 1]$ and $\Pr[\text{Hyb}_4 = 1] = \Pr[\text{Exp}_{\Sigma_{\text{cefe}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, 1) = 1]$. Therefore, Theorem 4.4.12 easily follows from Propositions 4.4.13 to 4.4.16. (Whose proof is given later.) $\square$

Proposition 4.4.13. *If Σ_{cence} is certified everlasting RNC secure, it holds that*

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

Proposition 4.4.14.

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

Proposition 4.4.15.

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

Proposition 4.4.16. *If Σ_{nad} is 1-bounded non-adaptive certified everlasting simulation-secure, it holds that*

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

Proof of Proposition 4.4.13. When an adversary makes key queries in step 2, it is clear that $\Pr[\text{Hyb}_0 = 1] = \Pr[\text{Hyb}_1 = 1]$. Hence, we consider the case where the adversary does not make a key query in step 2 below.

We assume that $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ 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 nce.pk from the challenger of $\text{Exp}_{\Sigma_{\text{cence}}, \mathcal{B}}^{\text{cert-ever-rec-nc}}(\lambda, b)$, generates $(\text{nad.MPK}, \text{nad.MSK}) \leftarrow \text{NAD.KeyGen}(1^\lambda)$, and sends $(\text{nad.MPK}, \text{nce.pk})$ to $\mathcal{A}_1$.
2. $\mathcal{B}$ receives a message $m \in \mathcal{M}$, computes $(\text{nad.vk}, \text{nad.ct}) \leftarrow \text{NAD.Enc}(\text{nad.MPK}, m)$, generates $a, c \leftarrow \{0, 1\}^n$, computes $\Psi := Z^c X^a \text{nad.ct} X^a Z^c$, sends (a, c) to the challenger, receives $(\text{nce.ct}^*, \text{nce.sk}^*)$ from the challenger, and sends $(\Psi, \text{nce.ct}^*)$ 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 $\text{nad.sk}_f \leftarrow \text{NAD.KeyGen}(\text{nad.MSK}, f)$, and sends $(\text{nad.sk}_f, \text{nce.sk}^*)$ to $\mathcal{A}_1$.
4. $\mathcal{A}_1$ sends $(\text{nad.cert}, \text{nce.cert})$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.
5. $\mathcal{B}$ sends nce.cert to the challenger, and receives nce.MSK or $\perp$ from the challenger. $\mathcal{B}$ computes $\text{nad.cert}^* \leftarrow \text{NAD.Recover}(a, c, \text{nad.cert})$ and $\text{NAD.Vrfy}(\text{nad.vk}, \text{nad.cert}^*)$. If the result is $\top$ and $\mathcal{B}$ receives nce.MSK from the challenger, $\mathcal{B}$ sends $(\text{nad.MSK}, \text{nce.MSK})$ 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_1 = 1]$. By assumption, $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ 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} . $\square$

Proof of Proposition 4.4.14. 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 $Z^c X^a \text{nad.ct} X^a Z^c$ 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 $Z^z X^x \text{nad.ct} X^x Z^z$. These two things are obvious from Lemma 2.3.3. $\square$

Proof of Proposition 4.4.15. The difference between Hyb_2 and Hyb_3 is only the order of operating the algorithm NAD.Enc and the Bell measurement on 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 . $\square$

Proof of Proposition 4.4.16. We assume that $|\Pr[\text{Hyb}_3 = 1] - \Pr[\text{Hyb}_4 = 1]|$ 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 $\text{Exp}_{\Sigma_{\text{nad}}, \mathcal{B}}^{\text{cert-ever-noada-sim}}(\lambda, b)$, generates $(\text{nce.pk}, \text{nce.MSK}) \leftarrow \text{NCE.Setup}(1^\lambda)$, and sends $(\text{nad.MPK}, \text{nce.pk})$ 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 nad.sk_f from the challenger, computes $\text{nce.sk} \leftarrow \text{NCE.KeyGen}(\text{nce.MSK})$, and sends $(\text{nad.sk}_f, \text{nce.sk})$ to $\mathcal{A}_1$.

3. $\mathcal{A}_1$ chooses $m \in \mathcal{M}$ and sends m to $\mathcal{B}$.
4. $\mathcal{B}$ does the following.
 - If a key query is called in step 2, $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.ct from the challenger, generates $a, c \leftarrow \{0, 1\}^n$, $\Psi := Z^c X^a \text{nad.ct} X^a Z^c$ and $(\text{nce.vk}, \text{nce.ct}) \leftarrow \text{NCE.Enc}(\text{nce.pk}, (a, c))$, and sends $(\Psi, \text{nce.ct})$ to $\mathcal{A}_1$.
 - If a key query is not called in step 2, $\mathcal{B}$ generates $|0^n 0^n\rangle$. Let $\Psi_A := \text{Tr}_B(|0^n 0^n\rangle\langle 0^n 0^n|)$ and $\Psi_B := \text{Tr}_A(|0^n 0^n\rangle\langle 0^n 0^n|)$. $\mathcal{B}$ computes $(\text{nce.vk}, \widetilde{\text{nce.ct}}, \text{nce.aux}) \leftarrow \text{NCE.Fake}(\text{nce.pk})$ and sends $(\Psi_A, \widetilde{\text{nce.ct}})$ 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 nad.sk_f from the challenger. $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.ct , measures the i -th qubit of nad.ct and Ψ_B in the Bell basis, and let (x_i, z_i) be the measurement outcome for all $i \in [n]$. $\mathcal{B}$ computes $\widetilde{\text{nce.sk}} \leftarrow \text{NCE.Reveal}(\text{nce.pk}, \text{nce.MSK}, \text{nce.aux}, (x, z))$ and sends $(\text{nad.sk}_f, \widetilde{\text{nce.sk}})$ to $\mathcal{A}_1$.
6. If $\mathcal{B}$ does not receive a key query throughout the experiment, $\mathcal{B}$ sends a challenge query m to the challenger, receives nad.ct , and measures the i -th qubit of nad.ct and Ψ_B in the Bell basis and let (x_i, z_i) be the measurement outcome for all $i \in [n]$.
7. $\mathcal{A}_1$ sends $(\text{nad.cert}, \text{nce.cert})$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.
8. $\mathcal{B}$ computes $\text{nad.cert}^* \leftarrow \text{NAD.Recover}(x^*, z^*, \text{nad.cert})$, 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 $\text{NCE.Vrfy}(\text{nce.vk}, \text{nce.cert})$. If the result is $\top$ and $\mathcal{B}$ receives nad.MSK from the challenger, $\mathcal{B}$ sends $(\text{nad.MSK}, \text{nce.MSK})$ 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_3 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_4 = 1]$. By assumption, $|\Pr[\text{Hyb}_3 = 1] - \Pr[\text{Hyb}_4 = 1]|$ 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} . $\square$

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

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

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

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

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

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

Setup(1^λ):

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

KeyGen(MSK, C):

- Parse $\text{MSK} = \{\text{one.MSK}_i\}_{i \in [N]}$.
- 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 $\text{one.sk}_{C,\Delta,i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C,\Delta})$.
- Output $\text{sk}_C := (\Gamma, \Delta, \{\text{one.sk}_{C,\Delta,i}\}_{i \in \Gamma})$.

Enc(MPK, x):

- Parse $\text{MPK} = \{\text{one.MPK}_i\}_{i \in [N]}$.
- 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 $(\text{one.vk}_i, \text{one.ct}_i) \leftarrow \text{ONE.Enc}(\text{one.MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \xi_1(i), \dots, \xi_S(i)))$.
- Output $\text{vk} = \{\text{one.vk}_i\}_{i \in [N]}$ and $\text{ct} := \{\text{one.ct}_i\}_{i \in [N]}$.

Dec(sk_C, ct):

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

Del(ct):

- Parse $ct = \{\text{one}.ct_i\}_{i \in [N]}$.
- For $i \in [N]$, compute $\text{one}.cert_i \leftarrow \text{ONE}.Del(\text{one}.ct_i)$.
- Output $\text{cert} := \{\text{one}.cert_i\}_{i \in [N]}$.

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

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

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

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

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

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

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

Security: The following two theorems hold.

Theorem 4.4.17. *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 4.4.18, and therefore we omit it.

Theorem 4.4.18. *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 4.4.18. Let us denote the simulating algorithm of Σ_{one} as $\text{ONE}.Sim = \text{ONE}.(\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$. Let us describe how the simulator $Sim = (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$ works below.

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

1. Parse $\text{MPK} := \{\text{one}.MPK_i\}_{i \in [N]}$ and $\mathcal{V} := \{C_j, C_j(x), (\Gamma_j, \Delta_j, \{\text{one}.sk_{C_j, \Delta_j, i}\}_{i \in [\Gamma_j]})\}_{j \in [q^*]}$.
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 $i \in \{q^* + 1, \dots, q\}$. Let $\Delta_0 := \emptyset$. Let $\mathcal{L} := \bigcup_{i \neq i'} (\Gamma_i \cap \Gamma_{i'})$. Sim_1 aborts if $|\mathcal{L}| > t$ or there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j \neq i} \Delta_j) = \emptyset$.
3. Sim_1 uniformly and independently samples ℓ random degree t polynomials $\mu_1, \dots, \mu_\ell$ whose constant terms are all 0.
4. Sim_1 samples the polynomials $\xi_1, \dots, \xi_S$ as follows for $j \in [q]$:
 - fix $a^* \in \Delta_j \setminus (\Delta_0 \cup \dots \cup \Delta_{j-1})$;

- for all $a \in (\Delta_j \setminus (\Delta_0 \cup \dots \cup \Delta_{j-1})) \setminus \{a^*\}$, 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:

$$\eta_j(\cdot) = C_j(\mu_1(\cdot), \dots, \mu_\ell(\cdot)) + \sum_{a \in \Delta_j} \xi_a(\cdot).$$

- Finally, for all $a \notin (\Delta_1 \cup \dots \cup \Delta_q)$, set ξ_a to be a uniformly random degree Dt polynomial whose constant term is 0.

5. For each $i \in \mathcal{L}$, Sim_1 computes

$$(\text{one.vk}_i, \text{one.ct}_i) \leftarrow \text{ONE.Enc}(\text{one.MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \xi_1(i), \dots, \xi_S(i))).$$

6. For each $i \notin \mathcal{L}$, Sim_1 does the following:

- If $i \in \Gamma_j$ for some $j \in [q^*]$ ¹⁶, computes

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

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

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

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

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

1. Parse $\text{MSK} := \{\text{one.MSK}_i\}_{i \in [N]}$ and $st_{j-1} := (\{\Gamma_i\}_{i \in [q]}, \{\Delta_i\}_{i \in [q]}, \{\eta_s(i)\}_{s \in \{q^*+1, \dots, q\}, i \in \mathcal{L}}, \{\text{one.st}_i\}_{i \in [N] \setminus \mathcal{L}}, \{\text{one.vk}_i\}_{i \in \mathcal{L}})$.
2. For each $i \in \Gamma_j \cap \mathcal{L}$, generate $\text{one.sk}_{C_j, \Delta_j, i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C_j, \Delta_j})$.
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

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

For simplicity, let us denote one.st_i^* as one.st_i for $i \in \Gamma_j \setminus \mathcal{L}$.

4. Output $\text{sk}_{C_j} := (\Gamma_j, \Delta_j, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j})$ and $st_j := (\{\Gamma_i\}_{i \in [q]}, \{\Delta_i\}_{i \in [q]}, \{\eta_j(i)\}_{j \in \{q^*+1, \dots, q\}, i \in \mathcal{L}}, \{\text{one.st}_i\}_{i \in [N] \setminus \mathcal{L}}, \{\text{one.vk}_i\}_{i \in \mathcal{L}})$.

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

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

¹⁶Note that j is uniquely determined since $i \notin \mathcal{L}$.

3. Output $\text{vk} := \{\text{one.vk}_i\}_{i \in [N]}$.

Let us define the sequence of hybrids as follows.

Hyb₀: This is identical to $\text{Exp}_{\Sigma_{\text{cefer}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, 0)$.

1. The challenger generates $(\text{one.MPK}_i, \text{one.MSK}_i) \leftarrow \text{ONE.Setup}(1^\lambda)$ 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 $\text{one.sk}_{C_j, \Delta_j, i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C_j, \Delta_j})$, and sends $(\Gamma_j, \Delta_j, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j})$ 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 $(\text{one.vk}_i, \text{one.ct}_i) \leftarrow \text{ONE.Enc}(\text{one.MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \xi_1(i), \dots, \xi_S(i)))$, and sends $\{\text{one.ct}_i\}_{i \in [N]}$ 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 $\text{one.sk}_{C_j, \Delta_j, i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C_j, \Delta_j})$, and sends $(\Gamma_j, \Delta_j, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j})$ to $\mathcal{A}_1$.
6. $\mathcal{A}_1$ sends $\{\text{one.cert}_i\}_{i \in [N]}$ to the challenger and its internal state to $\mathcal{A}_2$.
7. If $\top \leftarrow \text{ONE.Vrfy}(\text{one.vk}_i, \text{one.cert}_i)$ for all $i \in [N]$, the challenger outputs $\top$ and sends $\{\text{one.MSK}_i\}_{i \in [N]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$ and sends $\perp$ to $\mathcal{A}_2$.
8. $\mathcal{A}_2$ outputs b .
9. The experiment outputs b if the challenger outputs $\top$. Otherwise, the experiment outputs $\perp$.

Hyb₁: This is identical to Hyb₀ 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 $i \in \{q^* + 1, \dots, q\}$ 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 $\perp$. 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 $\perp$.

Hyb₂: This is identical to Hyb₁ except that the challenger samples $\xi_1, \dots, \xi_S, \eta_1, \dots, \eta_q$ as in the simulator Sim_1 described above.

Hyb₃: This is identical to Hyb₂ except that the challenger generates $\{\text{one.ct}_i\}_{i \in [N] \setminus \{\mathcal{L}\}}, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j}$ for $j \in \{q^* + 1, \dots, q'\}$, and $\text{vk} := \{\text{one.vk}_i\}_{i \in [N] \setminus \{\mathcal{L}\}}$ as in the simulator $\text{Sim} = (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$ described above, where q' is the number of key queries that the adversary makes in total.

Hyb₄: This is identical to Hyb₃ except that the challenger generates $\mu_1, \dots, \mu_\ell$ as in the simulator Sim_1 described above.

From the definition of $\text{Exp}_{\Sigma_{\text{cefer}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, b)$ and $\text{Sim} = (\text{Sim}_1, \text{Sim}_2, \text{Sim}_3)$, it is clear that $\Pr[\text{Hyb}_0 = 1] = \Pr[\text{Exp}_{\Sigma_{\text{cefer}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, 0) = 1]$ and $\Pr[\text{Hyb}_4 = 1] = \Pr[\text{Exp}_{\Sigma_{\text{cefer}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, 1) = 1]$. Therefore, Theorem 4.4.18 easily follows from Propositions 4.4.19 to 4.4.22 (whose proofs are given later). $\square$

Proposition 4.4.19. $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]| \leq \text{negl}(\lambda)$.

Proposition 4.4.20. $\Pr[\text{Hyb}_1 = 1] = \Pr[\text{Hyb}_2 = 1]$.

Proposition 4.4.21. If Σ_{one} is 1-bounded adaptive certified everlasting simulation-secure,

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

Proposition 4.4.22. $\Pr[\text{Hyb}_3 = 1] = \Pr[\text{Hyb}_4 = 1]$.

Proof of Proposition 4.4.19. 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 $i \in \{q^* + 1, \dots, q\}$ in step 4. It is clear that $\Pr[\text{Hyb}_0 = 1] = \Pr[\text{Hyb}'_0 = 1]$.

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

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

from Lemma 2.2.31.

Let 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 Collide does not occur. Hence, it is clear that $\Pr[\text{Hyb}^*_0 = 1 \wedge \overline{\text{Collide}}] = \Pr[\text{Hyb}_1 = 1 \wedge \overline{\text{Collide}}]$. Therefore, it holds that

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

from Lemma 2.2.31.

From the discussion above, we have

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

The following Lemmata 4.4.23 and 4.4.24 shows that $\Pr[|\mathcal{L}| > t] \leq 2^{-\Omega(\lambda)}$ and $\Pr[\text{Collide}] \leq q2^{-\Omega(\lambda)}$, which completes the proof. $\square$

Lemma 4.4.23 ([GVW12]). Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ 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 $\Gamma_1, \dots, \Gamma_q$.

Lemma 4.4.24 ([GVW12]). Let $\Delta_1, \dots, \Delta_q \subseteq [S]$ be randomly chosen subsets of size v . Let $v(\lambda) = \Theta(\lambda)$ and $S(\lambda) = \Theta(vq^2)$. Let Collide be the event that there exists some $i \in [q]$ such that $\Delta_i \setminus (\bigcup_{j \neq i} \Delta_j) = \emptyset$. Then, we have

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

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

Proof of Proposition 4.4.20. In the encryption in Hyb_1 , ξ_{a^*} is chosen at random and $\eta_j(\cdot)$ is defined by the relation. Sim essentially chooses $\eta_j(\cdot)$ at random which defines ξ_{a^*} . It is easy to see that reversing the order of how the polynomials are chosen produces the same distribution. $\square$

Proof of Proposition 4.4.21. To prove the proposition, let us define a hybrid experiment Hyb_2^s for each $s \in [N]$ as follows.

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

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

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

for $s \in [N]$. (Its proof is given later.) From these facts, we obtain Proposition 4.4.21.

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 $\left| \Pr[\text{Hyb}_2^{s-1} = 1] - \Pr[\text{Hyb}_2^s = 1] \right|$ 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 one.MPK from the challenger of $\text{Exp}_{\Sigma_{\text{one}}, \mathcal{A}}^{\text{cert-ever-ada-sim}}(\lambda, b)$. $\mathcal{B}$ sets $\text{one.MPK}_s := \text{one.MPK}$.
2. $\mathcal{B}$ generates $(\text{one.MPK}_i, \text{one.MSK}_i) \leftarrow \text{ONE.Setup}(1^\lambda)$ for all $i \in [N] \setminus s$, and sends $\{\text{one.MPK}_i\}_{i \in [N]}$ 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 $\text{one.sk}_{C_j, \Delta_j, i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C_j, \Delta_j})$. If $s \in \Gamma_j$, $\mathcal{B}$ sends G_{C_j, Δ_j} to the challenger, receives $\text{one.sk}_{C_j, \Delta_j, s}$ from the challenger, and sends $(\Gamma_j, \Delta_j, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j})$ 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 $i \in \{q^* + 1, \dots, q\}$. $\mathcal{B}$ generates a random degree t polynomial $\mu_i(\cdot)$ whose constant term is $x[i]$ for $i \in [\ell]$, and $\xi_1, \dots, \xi_s, \eta_1, \dots, \eta_q$ as in the simulator Sim_1 . For $i \in [s-1] \setminus \mathcal{L}$, $\mathcal{B}$ generates one.ct_i as in the simulator Sim_1 . For $i \in \{s+1, \dots, N\} \cup \mathcal{L}$, $\mathcal{B}$ generates $(\text{one.vk}_i, \text{one.ct}_i) \leftarrow \text{ONE.Enc}(\text{one.MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \xi_1(i), \dots, \xi_s(i)))$. $\mathcal{B}$ sends $\mu_1(s), \dots, \mu_\ell(s), \xi_1(s), \dots, \xi_s(s)$ to the challenger, and receives one.ct_s from the challenger. $\mathcal{B}$ sends $\{\text{one.ct}_i\}_{i \in [N]}$ 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 $\text{one.sk}_{C_j, \Delta_j, i} \leftarrow \text{ONE.KeyGen}(\text{one.MSK}_i, G_{C_j, \Delta_j})$. For $i \in \Gamma_j \cap [s-1]$, $\mathcal{B}$ generates $\text{one.sk}_{C_j, \Delta_j, i}$ as in the simulator Sim_2 . If $s \in \Gamma_j$, $\mathcal{B}$ sends G_{C_j, Δ_j} to the challenger, and receives $\text{one.sk}_{C_j, \Delta_j, s}$ from the challenger. $\mathcal{B}$ sends $(\Gamma_j, \Delta_j, \{\text{one.sk}_{C_j, \Delta_j, i}\}_{i \in \Gamma_j})$ to $\mathcal{A}_1$.
7. For $i \in [s-1] \setminus \mathcal{L}$, $\mathcal{B}$ generates one.vk_i as in the simulator Sim_3 ¹⁷.
8. $\mathcal{A}_1$ sends $\{\text{one.cert}_i\}_{i \in [N]}$ to $\mathcal{B}$ and its internal state to $\mathcal{A}_2$.

¹⁷For $i \in \{s+1, \dots, N\} \cup \mathcal{L}$, $\mathcal{B}$ generated one.vk_i in step 5.

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

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_2^{s-1} = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_2^s = 1]$. By assumption, $|\Pr[\text{Hyb}_2^{s-1} = 1] - \Pr[\text{Hyb}_2^s = 1]|$ 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 adaptive certified everlasting simulation-security of Σ_{one} . $\square$

Proof of Proposition 4.4.22. In Hyb_3 , the polynomials $\mu_1, \dots, \mu_\ell$ are chosen with constant terms $x_1, \dots, x_\ell$, respectively. In Hyb_4 , these polynomials are now chosen with 0 constant terms. This only affects the distribution of $\mu_1, \dots, \mu_\ell$ themselves and polynomials $\xi_1, \dots, \xi_S$. 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 $\{\mu_1(i), \dots, \mu_\ell(i)\}_{i \in \mathcal{L}}$ 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 $\{\xi_1(i), \dots, \xi_S(i)\}_{i \in \mathcal{L}}$ depend only on the values $\{\mu_1(i), \dots, \mu_\ell(i)\}_{i \in \mathcal{L}}$.

Proposition 4.4.22 follows from these observations. $\square$

4.4.5 Discussion on q -Bounded Consturction for All Circuits

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

Gorbunov, Vaikuntanathan, and Wee [GVW12] presented a conversion from FE for $\mathbf{NC}^1$ to FE for $\mathbf{P}/\text{poly}$ 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 $\widetilde{f(m)}$.¹⁸ 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 $\mathbf{NC}^1$, 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 $\text{Enc}(m)$ 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 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

¹⁸For simplicity, we ignore how to set randomness for randomized encoding here since it is not an essential issue.

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 fhe.ct and an FHE decryption key fhe.sk and outputs fhe.ct and $\text{FHE.Dec}(\text{fhe.sk}, \text{FHE.Eval}(f, \text{fhe.ct}))$. Here, we must output fhe.ct as the public part because we use FE for NC^1 and need to apply $f \in \mathbf{P/poly}$ by FHE.Eval in the public part (though the FHE decryption part is in NC^1).¹⁹ 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 a 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 4.5.2 (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, fhe.ct is an encryption of $m \oplus \bigoplus_i \theta_i$ where θ is a basis choice as in Section 4.5.2. To unmask $\bigoplus_i \theta_i$, we need to coherently apply f to 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 $\mathbf{P/poly}$ *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 $\mathbf{P/poly}$ do not work in the certified everlasting setting.

4.5 Compute-and-Compare Obfuscation with Certified Everlasting Deletion

4.5.1 Definition

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

Definition 4.5.1 (Compute-and-Compare Obfuscation with Certified Everlasting Deletion (Syntax)). A compute-and-compare obfuscation with certified everlasting deletion is a tuple of algorithms $(\text{CCObf}, \text{Del}, \text{Vrfy})$ for the family of distributions $D = \{D_{\text{param}}\}_{\text{param}}$ and message space $\mathcal{M}$.

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

$\text{Del}(\tilde{P}) \rightarrow \text{cert}$: The deletion algorithm takes as input an obfuscated circuit $\tilde{P}$ and outputs a classical certificate cert .

$\text{Vrfy}(\text{vk}, \text{cert}) \rightarrow \top \text{ or } \perp$: The verification algorithm takes as input the verification key vk and a certificate cert , and outputs $\top$ or $\perp$.

¹⁹See [GVW12] for the detail.

Definition 4.5.2 (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 $D = \{D_{\text{param}}\}_{\text{param}}$ 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

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

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

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}) \neq \top \mid \begin{array}{l} (\tilde{P}, \text{vk}) \leftarrow \text{CCObf}(1^\lambda, P, \text{lock}, m) \\ \text{cert} \leftarrow \text{Del}(\tilde{P}) \end{array} \right] \leq \text{negl}(\lambda).$$

Definition 4.5.3 (Certified Everlasting Security of Compute-and-Compare Obfuscation). Let $\Sigma_{\text{CCO}} = (\text{CCObf}, \text{Del}, \text{Vrfy})$ be a compute-and-compare obfuscation with certified everlasting deletion for the family of distributions $D = \{D_{\text{param}}\}_{\text{param}}$ and a message space $\mathcal{M}$. We consider experiments $\text{EV-Exp}_{\Sigma_{\text{CCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, b)$ and $\text{C-Exp}_{\Sigma_{\text{CCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, b)$ played between a challenger and a non-uniform QPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda, |\psi\rangle_\lambda\}_{\lambda \in \mathbb{N}}$. Let 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 $(P, \text{lock}, \text{aux}) \leftarrow D_{\text{param}}$.
3. The challenger computes $(\tilde{P}^{(0)}, \text{vk}^{(0)}) \leftarrow \text{CCObf}(1^\lambda, P, \text{lock}, m)$ or $(\tilde{P}^{(1)}, \text{vk}^{(1)}) \leftarrow \text{Sim}(1^\lambda, \text{pp}_P, 1^{|m|})$ and sends $(\tilde{P}^{(b)}, \text{aux})$ 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 $\text{Vrfy}(\text{vk}^{(b)}, \text{cert})$. If the outcome is $\top$, the experiment $\text{EV-Exp}_{\Sigma_{\text{CCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, b)$ outputs ρ ; otherwise if the outcome is $\perp$ then $\text{EV-Exp}_{\Sigma_{\text{CCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, b)$ outputs $\perp$ and ends.
6. The challenger sends the outcome of $\text{Vrfy}(\text{vk}^{(b)}, \text{cert})$ to $\mathcal{A}_\lambda$.
7. $\mathcal{A}_\lambda$ outputs its guess $b' \in \{0, 1\}$ which is the output of the experiment $\text{C-Exp}_{\Sigma_{\text{CCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, b)$.

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

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

and

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

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

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

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

1. Sample $R \leftarrow \{0, 1\}^\lambda$.
2. Sample $(\text{fpk}, \text{fsk}) \leftarrow \text{FHE.KeyGen}(1^\lambda)$.
3. Compute $\widetilde{\text{fDec}} \leftarrow \text{CC.Obf}(1^\lambda, \text{fDec}, R, 1)$ where $\text{fDec}(\cdot) = \text{FHE.Dec}(\text{fsk}, \cdot)$.
4. Compute $\widetilde{I} \leftarrow \text{CC.Obf}(1^\lambda, I, \text{lock}, R)$ where $I(X) = X$ for every X .
5. Represent $(P \parallel \widetilde{I}) = (b_1, \dots, b_\ell) \in \{0, 1\}^\ell$.
6. Sample $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ 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 $m = (m_1, \dots, m_n) \in \{0, 1\}^n$.
9. Sample $\theta_{\ell+k}, z_{\ell+k} \leftarrow \{0, 1\}^\lambda$ 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 $\text{fct}_i \leftarrow \text{FHE.Enc}(\text{fpk}, (\theta_i, \widetilde{b}_i))$ for all $i \in [\ell + n]$.
12. Output $\widetilde{\mathcal{P}} := (\widetilde{\text{fDec}}, \{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell+n]}, \text{fpk})$ and $\text{vk} := (\{z_i, \theta_i\}_{i \in [\ell+n]})$.

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

1. Parse $\widetilde{\mathcal{P}} := (\widetilde{\text{fDec}}, \{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell+n]}, \text{fpk})$.
2. Define a circuit $\widehat{U}_x$ as in Figure 4.2.
3. To compute an evaluated ciphertext for $\text{FHE.Eval}(\text{fpk}, \widehat{U}_x, \cdot)$, apply $\widehat{U}_x$ homomorphically in superposition with the input $(\{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell]}, (|z_{\ell+k}\rangle_{\theta_{\ell+k}}, \text{fct}_{\ell+k}))$ and obtain a ciphertext $|\text{fct}_{\ell+k,P}\rangle$ for each $k \in [n]$.
4. Apply $\widetilde{\text{fDec}}(\cdot)$ in superposition with the input $|\text{fct}_{\ell+k,P}\rangle$ 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 $m = (m_1, \dots, m_n)$.

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

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

Circuit $\hat{U}_x$

Hardware: x

Input: $(\{(z_i, \theta_i, \tilde{b}_i)\}_{i \in [\ell]}, (z_{\ell+k}, \theta_{\ell+k}, \tilde{b}_{\ell+k}))$

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

Figure 4.2: The description of the circuit $\hat{U}_x$

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

1. Parse $\text{vk} = (\{(z_i, \theta_i)\}_{i \in [\ell+n]})$ and $\text{cert} = (z'_1, \dots, z'_{\ell+n})$.
2. If $\left((z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1) \right)$ holds for all $i \in [\ell+n]$ and $j \in [\lambda]$, then output $\top$; otherwise output $\perp$.

Security. We use Lemma 4.3.5 by Bartusek and Khurana [BK23] to prove the security of our construction.

Theorem 4.5.4. *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 $D = \{D_{\text{param}}\}_{\text{param}}$.*

Proof of Theorem 4.5.4. We first describe the simulator of CCObf , denoted as 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 $(P, \text{lock}, \text{aux}) \leftarrow D_{\text{param}}$, the algorithm Sim works as follows:

$\text{Sim}(1^\lambda, \text{pp}_P, 1^n)$:

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

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

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

using Lemma 4.3.5 and the post-quantum security of Σ_{CCO} and Σ_{fhe} . We use the following sequence of hybrids to prove this.

Hyb₀: This is the same as $\text{EV-Exp}_{\Sigma_{\text{CECCO}}, \mathcal{A}}^{\text{sim-ccobf}}(\lambda, 0)$. Let $\tilde{\mathcal{P}}^{(0)} := (\widetilde{\text{fDec}}, \{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell+n]}, \text{fpk})$ be the compute-and-compare obfuscated circuit computed using the honest *CCObf* algorithm.

Hyb₁: This hybrid works as follows:

1. $\mathcal{A}$ submits a message $m \in \{0, 1\}^n$ to the challenger.
2. The challenger chooses $(P, \text{lock}, \text{aux}) \leftarrow D_{\text{param}}$.
3. The challenger computes the obfuscated circuit as follows:
 - (a) Sample $(\text{fpk}, \text{fsk}) \leftarrow \text{FHE.KeyGen}(1^\lambda)$ and $R \leftarrow \{0, 1\}^\lambda$.
 - (b) Compute $\widetilde{\text{fDec}} \leftarrow \text{CC.Obf}(1^\lambda, \text{fDec}, R, 1)$.
 - (c) Sample $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell + n]$.
 - (d) Set $\tilde{b}_i := 0 \oplus \bigoplus_{j: \theta_{ij}=0} z_{i,j}$ for $i \in [\ell]$.
 - (e) Set $\tilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j: \theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$.
 - (f) Compute $\text{fct}_i \leftarrow \text{FHE.Enc}(\text{fpk}, (\theta_i, \tilde{b}_i))$ for all $i \in [\ell + n]$.
 - (g) Set $\tilde{\mathcal{P}} := (\widetilde{\text{fDec}}, \{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell+n]}, \text{fpk})$.

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

4. $\mathcal{A}$ sends a deletion certificate $\text{cert} := (z'_1, \dots, z'_{\ell+n})$ and its internal state ρ to the challenger.
5. The challenger checks if $((z_{i,j} = z'_{i,j}) \wedge (\theta_{i,j} = 1))$ 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 $\{\text{fct}_i\}_{i \in [\ell]}$ contain no information about the lock value lock , the random string R and the circuit P . To prove the indistinguishability between Hyb₀ and Hyb₁, we consider a sequence of intermediate hybrids Hyb_{1,k} for $k \in [0, \ell]$ where Hyb_{1,0} is identical to Hyb₀ and the only difference between Hyb_{1,k-1} and Hyb_{1,k} is that fct_k is an encryption of $(\theta_k, b_k \oplus \bigoplus_{j: \theta_{k,j}=0} z_{k,j})$ where b_k in Hyb_{1,k-1} is the same as b_k in Hyb₀ and b_k in Hyb_{1,k} is set to zero for $k \in [\ell]$.

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

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

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

(f) Set $\tilde{b}_i$ for $i \in [\ell]$ as follows:

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

(g) Set $\tilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$.

(h) Compute $\text{fct}_i \leftarrow \text{FHE.Enc}(\text{fpk}, (\theta_i, \tilde{b}_i))$ for all $i \in [\ell+n]$ where $\theta_k := \theta$.

The challenge sends $(\widetilde{\text{fDec}}, \{(|z_i\rangle_{\theta_i}, \text{fct}_i)\}_{i \in [\ell+n] \setminus \{k\}}, \text{fct}_k, \text{fpk})$ to $\mathcal{B}$.

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

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

$$\left| \Pr[\mathcal{Z}_\lambda^k(\theta) = 1] - \Pr[\mathcal{Z}_\lambda^k(\mathbf{0}_\lambda) = 1] \right| \leq \text{negl}(\lambda). \quad (4.5)$$

$\mathcal{Z}_\lambda^{k,1}$: This is exactly the same as $\mathcal{Z}_\lambda^k(\theta)$ except the challenger uses the bits of $\tilde{I} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_I, 1^{|R|})$ instead of $\tilde{I} \leftarrow \text{CC.Obf}(1^\lambda, I, \text{lock}, R)$ to set $\tilde{b}_i$ for all $i \in [k+1, \ell]$. The indistinguishability between the distributions $\mathcal{Z}_\lambda^k(\theta)$ 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 $\widetilde{\text{fDec}} \leftarrow \text{CC.Obf}(1^\lambda, \text{fDec}, R, 1)$ with the simulated obfuscated circuit $\widetilde{\text{fDec}} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_{\text{fDec}}, 1^1)$. 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 $\text{fct}_k \leftarrow \text{FHE.Enc}(\text{fpk}, (\mathbf{0}_\lambda, \tilde{b}_k))$ instead of encrypting $(\theta, \tilde{b}_k)$. 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 $\widetilde{\text{fDec}} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_{\text{fDec}}, 1^1)$ with the real obfuscated circuit $\widetilde{\text{fDec}} \leftarrow \text{CC.Obf}(1^\lambda, \text{fDec}, R, 1)$. 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 $\tilde{I} \leftarrow \text{CC.Obf}(1^\lambda, I, \text{lock}, R)$ instead of $\tilde{I} \leftarrow \text{CC.Sim}(1^\lambda, \text{pp}_I, 1^1)$ to set $\tilde{b}_i$ 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 (4.5) holds for all $k \in [\ell]$. Therefore, by Lemma 4.3.5, for any (unbounded) adversary $\mathcal{B}'$ we have

$$\text{TD}(\widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, 0), \widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, 1)) \leq \text{negl}(\lambda) \quad (4.6)$$

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

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

1. Sample $\mathbf{z}, \boldsymbol{\theta} \leftarrow \{0, 1\}^\lambda$.
2. $\mathcal{B}'$ receives $(1^\lambda, |\mathbf{z}\rangle_\theta)$ as input.
3. $\mathcal{B}'$ interacts with $\mathcal{C}$ as in $\text{Expt}_{\mathcal{B}, \mathcal{C}}^{1,k}(\lambda, \boldsymbol{\theta}, b \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j})$ where $\mathcal{B}'$ plays the role of $\mathcal{B}$.
4. $\mathcal{B}'$ outputs a string $\mathbf{z}' \in \{0, 1\}^\lambda$ 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 $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$ is that $\tilde{b}_k$ is set to be $b_k \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\text{Hyb}_{1,k-1}$ and $0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ in $\text{Hyb}_{1,k}$. Let us assume $b_k = 1$, since otherwise $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$ are identical. We construct $\mathcal{B}'$ that distinguishes $\widetilde{\text{Expt}}_{\mathcal{B}', \mathcal{C}}^{1,k}(\lambda, 0)$ and $\widetilde{\text{Expt}}_{\mathcal{B}', \mathcal{C}}^{1,k}(\lambda, 1)$ if $\mathcal{A}$ distinguishes between the hybrids $\text{Hyb}_{1,k-1}$ and $\text{Hyb}_{1,k}$.

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

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

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

$$\text{TD}(\text{Hyb}_{1,k-1}, \text{Hyb}_{1,k}) \leq \text{TD}(\tilde{\mathcal{Z}}_\lambda^k(0), \tilde{\mathcal{Z}}_\lambda^k(1)). \quad (4.7)$$

Combining Equations (4.6) and (4.7), we have

$$\text{TD}(\text{Hyb}_{1,k-1}, \text{Hyb}_{1,k}) \leq \text{negl}(\lambda). \quad (4.8)$$

Recall that $\text{Hyb}_{1,0} \equiv \text{Hyb}_0$ and $\text{Hyb}_{1,\ell} \equiv \text{Hyb}_1$. Therefore, combining the advantages of $\mathcal{A}$ in the sequence of intermediate hybrids as obtained in Equation 4.8, we have

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

Hyb₂: 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 $\{\text{fct}_{\ell+k}\}_{k \in [n]}$ are encrypted to the message $\mathbf{0}_n$. More precisely, the challenger samples $\boldsymbol{\theta}_{\ell+k}, \mathbf{z}_{\ell+k} \leftarrow \{0, 1\}^\lambda$ and sets $\tilde{b}_{\ell+k} := 0 \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$ for all $k \in [n]$ instead of setting $\tilde{b}_{\ell+k} := m_k \oplus \bigoplus_{j:\theta_{\ell+k,j}=0} z_{\ell+k,j}$. Finally, it obtains $\text{fct}_{\ell+k} \leftarrow \text{FHE.Enc}(\text{fpk}, (\boldsymbol{\theta}_{\ell+k}, \tilde{b}_{\ell+k}))$ for all $k \in [n]$ where the encrypted bits $\{\tilde{b}_{\ell+k}\}_{k \in [n]}$ 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

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

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

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

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

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

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 (4.5) except that fct_k is changed from encryption of $(\theta, \tilde{b}_k)$ to $(\theta, 0 \oplus \bigoplus_{j:\theta_{k,j}=0} z_{k,j})$ (instead of changing it from $(\theta, \tilde{b}_k)$ to $(0_\lambda, \tilde{b}_k)$ in $\mathcal{Z}_\lambda^{k,3}$). This completes the proof. $\square$

4.6 Predicate Encryption with Certified Everlasting Deletion

4.6.1 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 4.6.1 (PE with Certified Everlasting Deletion (Syntax)). A certified everlasting PE is tuple of QPT algorithms $(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ with a class predicates $\mathcal{P}$, a class of attributes $\mathcal{X}$ and a message space $\mathcal{M}$.

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

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

$\text{Enc}(\text{pk}, x, m) \rightarrow (ct, vk)$: 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 .

$\text{Dec}(\text{sk}_P, ct) \rightarrow m' \text{ or } \perp$: The decryption algorithm takes as input a secret key sk_P and a quantum ciphertext ct , and outputs a classical plaintext m' or $\perp$.

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

$\text{Vrfy}(vk, \text{cert}) \rightarrow \top \text{ or } \perp$: The verification algorithm takes as input the verification key vk and a certificate cert , and outputs $\top$ or $\perp$.

Definition 4.6.2 (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 $\lambda \in \mathbb{N}, P \in \mathcal{P}, x \in \mathcal{X}, m \in \mathcal{M}$ such that $P(x) = 1$,

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

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

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

Definition 4.6.3 (Certified Everlasting Security of PE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ 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 $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ and $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ 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 $(\text{pk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda)$ and sends pk to $\mathcal{A}_\lambda(|\psi\rangle_\lambda)$.
2. $\mathcal{A}_\lambda$ sends $P \in \mathcal{P}$ to the challenger and receives $\text{sk}_P \leftarrow \text{KeyGen}(\text{msk}, P)$ 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 $P(x_0) = P(x_1) = 0$ for all P queried so far in the key query phase.
4. The challenger computes $(ct_b, \text{vk}_b) \leftarrow \text{Enc}(\text{pk}, x_b, m_b)$ and sends ct_b to $\mathcal{A}_\lambda$.
5. $\mathcal{A}_\lambda$ can make further key queries with P satisfying $P(x_0) = P(x_1) = 0$.
6. $\mathcal{A}_\lambda$ sends a certificate of deletion cert and its internal state ρ to the challenger.
7. The challenger computes $\text{Vrfy}(\text{vk}_b, \text{cert})$. If the outcome is $\top$, the experiment $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ outputs ρ ; otherwise if the outcome is $\perp$ then $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ output $\perp$ and ends.
8. The challenger sends the outcome of $\text{Vrfy}(\text{vk}^{(b)}, \text{cert})$ to $\mathcal{A}_\lambda$.
9. Again, $\mathcal{A}_\lambda$ can make key queries with polynomial number of policies P satisfying $P(x_0) = P(x_1) = 0$.
10. $\mathcal{A}_\lambda$ outputs its guess $b' \in \{0, 1\}$ which is the output of the experiment $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$.

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

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

and

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

We can define similar experiment $\text{EV-Exp}_{\Sigma, \mathcal{A}}^{\text{sel-ind}}(\lambda, b)$ and $\text{C-Exp}_{\Sigma, \mathcal{A}}^{\text{ada-ind}}(\lambda, b)$ 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

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

and

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

4.6.2 Construction

In this section, we construct a PE with certified everlasting deletion from a compute-and-compare obfuscation with certified everlasting deletion introduced in Section 4.5 and a classical ABE.

Ingredients. We use the following building blocks.

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

Let $D = \{D_{\text{apk}, x, \{\theta_i\}_i, \{z_i\}_i}\}_{\text{apk}, x, \{\theta_i\}_i, \{z_i\}_i}$ be a family of distributions where $D_{\text{apk}, x, \{\theta_i\}_i, \{z_i\}_i}$ outputs $(\text{aDec}, \text{lock}, \text{aux})$ generated as follows.

- Generate $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x, (\theta_i, \bigoplus_{j:\theta_{i,j}=0} z_{i,j}))$ for all $i \in [\ell]$.
- Construct aDec described in Figure 4.3.
- Choose $\text{lock} \leftarrow \{0, 1\}^\ell = \mathcal{K}$.
- Output $(\text{aDec}, \text{lock}, \text{aux} := \perp)$.

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

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

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

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

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

$\text{Enc}(\text{apk}, x, m)$

1. Parse $\text{pk} = \text{apk}$.
2. Sample $R \leftarrow \{0, 1\}^\ell$ and denote $R = (r_1, \dots, r_\ell)$.
3. Sample $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell]$.
4. Set $\tilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.

Hardware: $\{\text{act}_i\}_{i \in [\ell]}$

Input: $(\{z_i\}_{i \in [\ell]}, \text{sk}_P)$

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

Figure 4.3: The description of the circuit aDec

5. Compute $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x, (\theta_i, \tilde{r}_i))$ for all $i \in [\ell]$.
6. $(\widetilde{\text{aDec}}, \text{vk}_{\text{aDec}}) \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R, m)$ where aDec is defined in Figure 4.3.
7. Output $ct := (\widetilde{\text{aDec}}, \{|z_i\rangle_{\theta_i}\}_{i \in [\ell]})$ and $\text{vk} := (\{(z_i, \theta_i)\}_{i \in [\ell]}, \text{vk}_{\text{aDec}})$.

$\mathcal{Dec}(\text{sk}_P, ct)$

1. Parse $ct = (\widetilde{\text{aDec}}, \{|z_i\rangle_{\theta_i}\}_{i \in [\ell]})$.
2. Apply $\widetilde{\text{aDec}}$ in superposition to the input $(\{|z_i\rangle_{\theta_i}\}_{i \in [\ell]}, \text{sk}_P)$ and measure the output register to obtain m' .
3. Output m' .

$\mathcal{Del}(ct)$

1. Parse $ct = (\widetilde{\text{aDec}}, \{|z_i\rangle_{\theta_i}\}_{i \in [\ell]})$.
2. Measure $|z_i\rangle_{\theta_i}$ in the Hadamard basis for all $i \in [\ell]$ and obtain $z' := (z'_1, \dots, z'_\ell)$.
3. Compute $\text{cert}_{\text{aDec}} \leftarrow \text{CCO.Del}(\widetilde{\text{aDec}})$.
4. Output $\text{cert} := (z', \text{cert}_{\text{aDec}})$.

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

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

Theorem 4.6.4. *If Σ_{CECCO} is a certified everlasting secure compute-and-compare obfuscation for a message space $\mathcal{M}_{\text{pe}}$ and the family of distributions $D = \{D_{\text{apk}, x, \{\theta_i\}_i, \{z_i\}_i}\}_{\text{apk}, x, \{\theta_i\}_i, \{z_i\}_i}$ and Σ_{abe} is an adaptively (resp. selectively) secure ABE for a class of predicates $\mathcal{P}$, then $\Sigma_{\text{pe-ce}}$ is an adaptively (resp. selectively) certified everlasting secure predicate encryption scheme for the class of predicates $\mathcal{P}$, message space $\mathcal{M}_{\text{pe}}$.*

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 $\Sigma_{\text{pe-ce}}$. We consider the following sequence of hybrids.

Hyb₀ : This is the original certified everlasting security experiment where the challenge bit is set to 0 ($\text{EV-Exp}_{\Sigma_{\text{pe-ce}}, \mathcal{A}}^{\text{ada-ind}}(\lambda, 0)$). More precisely, it works as follows:

1. The challenger computes $(\text{apk}, \text{amsk}) \leftarrow \text{ABE.Setup}(1^\lambda)$, sets $\text{pk} := \text{apk}$, 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 $\text{sk}_P \leftarrow \text{ABE.KeyGen}(\text{amsk}, P)$ 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 $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell]$ where $\theta_i = (\theta_{i,j})_{j \in [\lambda]}$ and $z_i = (z_{i,j})_{j \in [\lambda]}$.
 - (c) Set $\tilde{r}_i := r_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
 - (d) Compute $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x_0, (\theta_i, \tilde{r}_i))$ for all $i \in [\ell]$.
 - (e) $(\widetilde{\text{aDec}}, \text{vk}_{\text{aDec}}) \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_0)$ where aDec is defined in Figure 4.3.
 - (f) Set $ct^* := (\widetilde{\text{aDec}}, \{z_i\}_{i \in [\ell]})$ and $\text{vk} := (\{(z_i, \theta_i)\}_{i \in [\ell]}, \text{vk}_{\text{aDec}})$.

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

5. $\mathcal{A}$ sends a certificate $\text{cert} = (z' = (z'_1, \dots, z'_\ell), \text{cert}_{\text{aDec}})$ 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 $\text{CCO.Vrfy}(\text{vk}_{\text{aDec}}, \text{cert}_{\text{aDec}}) = \top$. 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.

Hyb₁ : This hybrid proceeds exactly similar to **Hyb₀** 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 $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x_0, (\theta_i, 0 \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}))$ for all $i \in [\ell]$.

To prove the indistinguishability between **Hyb₀** and **Hyb₁**, we consider a sequence of intermediate hybrids **Hyb_{1,k}** for $k \in [\ell]$ where we take **Hyb_{1,0}** is identical to **Hyb₀** and the only difference between **Hyb_{1,k-1}** and **Hyb_{1,k}** is that act_k is an encryption of $(\theta_k, r_k \oplus \bigoplus_{j: \theta_{k,j}=0} z_{k,j})$ in **Hyb_{1,k-1}** whereas it is an encryption of $(\theta_k, 0 \oplus \bigoplus_{j: \theta_{k,j}=0} z_{k,j})$ in **Hyb_{1,k}**.

Now, we consider a sequence of experiments $\text{Expt}_{\mathcal{B}, \mathcal{C}}^{1,k}(\lambda, \theta, \beta)$ for $k \in [\ell]$ between a QPT adversary $\mathcal{B}$ and a challenger $\mathcal{C}$ for $\theta \in \{0, 1\}^\lambda$ and $\beta \in \{0, 1\}$. The experiment $\text{Expt}_{\mathcal{B}, \mathcal{C}}^{1,0}(\lambda, \theta, \beta)$ is basically the same as **Hyb₀** where $\mathcal{B}$ plays the role of $\mathcal{A}$ and $\mathcal{C}$ plays the role of the challenger. In particular, it works as follows:

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

1. $\mathcal{C}$ computes $(\text{apk}, \text{amsk}) \leftarrow \text{ABE.Setup}(1^\lambda)$, sets $\text{pk} := \text{apk}$, 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 $\text{sk}_P \leftarrow \text{ABE.KeyGen}(\text{amsk}, P)$ 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 $R^* = (r_1, \dots, r_\ell) \leftarrow \{0, 1\}^\ell$.
 - (b) Sample $z_i, \theta_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell] \setminus \{k\}$ where $\theta_i = (\theta_{i,j})_{j \in [\lambda]}$ and $z_i = (z_{i,j})_{j \in [\lambda]}$.
 - (c) Set $\tilde{r}_i$ as follows:

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

- (d) Compute act_i as follows:

$$\text{act}_i \leftarrow \begin{cases} \text{ABE.Enc}(\text{apk}, x_0, (\theta, \tilde{r}_k)) & \text{if } i = k \\ \text{ABE.Enc}(\text{apk}, x_0, (\theta_i, \tilde{r}_i)) & \text{if } i \in [\ell] \setminus \{k\} \end{cases}.$$

- (e) $(\widetilde{\text{aDec}}, \text{vk}_{\text{aDec}}) \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_0)$ where aDec is defined in Figure 4.3.

The challenger sends $(\widetilde{\text{aDec}}, \{|z_i\rangle_{\theta_i}\}_{i \in [\ell] \setminus \{k\}})$ 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

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

Therefore, by Lemma 4.3.7, for any QPT (unbounded) adversary $\mathcal{B}'$, we have

$$\text{TD}(\widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 0), \widetilde{\text{Expt}}_{\mathcal{B}',\mathcal{C}}^{1,k}(\lambda, 1)) \leq \text{negl}(\lambda) \quad (4.9)$$

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

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

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

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

1. $\mathcal{B}'$ plays the role of $\mathcal{A}$ in $\text{Hyb}_{1,k}$ where the external challenger C of $\widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, b)$ is used to simulate the challenger of $\text{Hyb}_{1,k}$. C provides everything that should be sent to $\mathcal{A}$ (as in Hyb_0).
2. Suppose $\mathcal{A}$ sends a certificate $\text{cert} = ((z'_1, \dots, z'_\ell), \text{cert}_{\text{aDec}})$ 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 $\text{Hyb}_{1,k}$ if $b = 0$ and $\text{Hyb}_{1,k-1}$ if $b = 1$ (since we are assuming $r_k = 1$). Therefore, we can write

$$\text{TD}(\text{Hyb}_{1,k-1}, \text{Hyb}_{1,k}) \leq \text{TD}(\widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, 0), \widetilde{\text{Expt}}_{\mathcal{B}',C}^{1,k}(\lambda, 1)). \quad (4.10)$$

Combining Equations 4.9 and 4.10, we have

$$\text{TD}(\text{Hyb}_{1,k-1}, \text{Hyb}_{1,k}) \leq \text{negl}(\lambda). \quad (4.11)$$

Recall that $\text{Hyb}_{1,0} \equiv \text{Hyb}_0$ and $\text{Hyb}_{1,\ell} \equiv \text{Hyb}_1$. Therefore, combining the advantages of $\mathcal{A}$ in the sequence of intermediate hybrids as obtained in Equation 4.11, we have

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

Hyb₂ : This hybrid proceeds exactly similar to Hyb_1 except that the obfuscated circuit is now replaced with a simulated version of it. In particular, $\widetilde{\text{aDec}} \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_0)$ is replaced with the circuit $\widetilde{\text{aDec}} \leftarrow \text{CCO.Sim}(1^\lambda, \text{pp}_{\text{aDec}}, 1^{|m_b|})$. In particular, the hybrid works as follows:

1. The challenger computes $(\text{apk}, \text{amsk}) \leftarrow \text{ABE.Setup}(1^\lambda)$, sets $\text{pk} := \text{apk}$, 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 $\text{sk}_P \leftarrow \text{ABE.KeyGen}(\text{amsk}, P)$ 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) $\widetilde{\text{aDec}} \leftarrow \text{CCO.Sim}(1^\lambda, \text{pp}_{\text{aDec}}, 1^{|m_b|})$ where aDec is defined in Figure 4.3. (Note that, we do not need to compute ABE ciphertexts since we only require the lengths of a ABE ciphertext in order to calculate pp_{aDec} .)
 - (c) Set $ct^* := (\widetilde{\text{aDec}}, \{z_i\}_{i \in [\ell]})$ and $\text{vk} := (\{(z_i, \theta_i)\}_{i \in [\ell]}, \text{vk}_{\text{aDec}})$.

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

5. $\mathcal{A}$ sends a certificate $\text{cert} = (z' = (z'_1, \dots, z'_\ell), \text{cert}_{\text{aDec}})$ 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 $\text{CCO.Vrfy}(\text{vk}_{\text{aDec}}, \text{cert}_{\text{aDec}}) = \top$. 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

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

Hyb₃ : 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 $\widetilde{\text{aDec}} \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_1)$ where the circuit aDec is defined using the hardwired values $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x_1, (\theta_i, 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}))$. In particular, the hybrid works as follows:

1. The challenger computes $(\text{apk}, \text{amsk}) \leftarrow \text{ABE.Setup}(1^\lambda)$, sets $\text{pk} := \text{apk}$, 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 $\text{sk}_P \leftarrow \text{ABE.KeyGen}(\text{amsk}, P)$ 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 $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell]$ where $\theta_i = (\theta_{i,j})_{j \in [\lambda]}$ and $z_i = (z_{i,j})_{j \in [\lambda]}$.
 - (c) Set $\tilde{r}_i := 0 \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
 - (d) Compute $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x_1, (\theta_i, \tilde{r}_i))$ for all $i \in [\ell]$.
 - (e) $\widetilde{\text{aDec}} \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_1)$ where aDec is defined in Figure 4.3.
 - (f) Set $ct^* := (\widetilde{\text{aDec}}, \{ |z_i\rangle_{\theta_i} \}_{i \in [\ell]})$ and $\text{vk} := (\{ (z_i, \theta_i) \}_{i \in [\ell]}, \text{vk}_{\text{aDec}})$.

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

5. $\mathcal{A}$ sends a certificate $\text{cert} = (z' = (z'_1, \dots, z'_\ell), \text{cert}_{\text{aDec}})$ 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 $\text{CCO.Vrfy}(\text{vk}_{\text{aDec}}, \text{cert}_{\text{aDec}}) = \top$. 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

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

Hyb₄ : This hybrid proceeds exactly similar to Hyb_3 except that the ABE ciphertexts act_i is now replaced with encryption of $(\theta_i, \tilde{r}_i)$ where $\tilde{r}_i := r_i \oplus \bigoplus_{j:\theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$. In particular, the hybrid works as follows:

1. The challenger computes $(\text{apk}, \text{amsk}) \leftarrow \text{ABE.Setup}(1^\lambda)$, sets $\text{pk} := \text{apk}$, 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 $\text{sk}_P \leftarrow \text{ABE.KeyGen}(\text{amsk}, P)$ 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 $\theta_i, z_i \leftarrow \{0, 1\}^\lambda$ for all $i \in [\ell]$ where $\theta_i = (\theta_{i,j})_{j \in [\lambda]}$ and $z_i = (z_{i,j})_{j \in [\lambda]}$.
 - (c) Set $\tilde{r}_i := r_i \oplus \bigoplus_{j: \theta_{i,j}=0} z_{i,j}$ for all $i \in [\ell]$.
 - (d) Compute $\text{act}_i \leftarrow \text{ABE.Enc}(\text{apk}, x_1, (\theta_i, \tilde{r}_i))$ for all $i \in [\ell]$.
 - (e) $\widetilde{\text{aDec}} \leftarrow \text{CCO.Obf}(1^\lambda, \text{aDec}, R^*, m_1)$ where aDec is defined in Figure 4.3.
 - (f) Set $ct^* := (\widetilde{\text{aDec}}, \{|z_i\rangle_{\theta_i}\}_{i \in [\ell]})$ and $\text{vk} := (\{(z_i, \theta_i)\}_{i \in [\ell]}, \text{vk}_{\text{aDec}})$.

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

5. $\mathcal{A}$ sends a certificate $\text{cert} = (z' = (z'_1, \dots, z'_\ell), \text{cert}_{\text{aDec}})$ 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 $\text{CCO.Vrfy}(\text{vk}_{\text{aDec}}, \text{cert}_{\text{aDec}}) = \top$. 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 satisfy 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

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

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

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

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

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

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

Chapter 5

Conclusion

In this thesis, we have introduced a novel cryptographic application of quantum mechanics called quantum cryptography with certified everlasting security and constructed several cryptographic primitives with certified everlasting security. In cryptography, ideally, we should achieve information-theoretical security because it ensures security against computationally unbounded adversaries. However, information-theoretical security is impossible to achieve for almost all cryptographic primitives even in the realm of quantum. Certified everlasting security provides a nice compromise of information theoretical security. In certified everlasting security, one can verify whether the adversary deletes encrypted data. Once the deletion is verified, the security becomes information theoretical security, and hence even unbounded adversaries cannot obtain the information of encrypted data after that.

In Section 3.1, we have explored a certified everlasting secure variant of zero-knowledge proof for **QMA**. In this primitive, the zero-knowledge property initially relies on computational security while soundness satisfies information theoretical security. However, the prover is equipped with a mechanism to verify whether the encrypted quantum data has been deleted. Once deletion is confirmed, the zero-knowledge property transitions to an information-theoretic guarantee. It is widely believed that in zero-knowledge proofs for **QMA**, it is impossible to achieve both soundness and zero-knowledge with information-theoretic security simultaneously. Indeed, in previous constructions of zero-knowledge proofs for **QMA**, either the soundness or the zero-knowledge property relies on computational security. Consequently, if soundness is guaranteed with information-theoretic security, the zero-knowledge property becomes computational, posing a risk that a malicious verifier may extract information beyond the validity of the statements in the future. Compared to all previous works, in our primitive, the zero-knowledge property changes to information theoretical security once the deletion is verified while soundness satisfies information theoretical security.

In Chapter 4, we have explored certified everlasting secure variants of various cryptographic primitives, including functional encryption, compute-and-compare obfuscation, and predicate encryption. In Section 4.3, we have constructed a certified everlasting secure variant of collusion-resistant functional encryption, based on indistinguishability obfuscation and one-way functions. In this primitive, an adversary possessing the quantum ciphertext $ct(m)$ can obtain polynomially many secret keys $sk_{f_i} \in [n]$ and evaluate $f_i(m)$ by using them. Additionally, it includes a mechanism to verify whether the adversary has deleted the quantum ciphertext. Once the deletion is verified, the security becomes an information-theoretic guarantee. As a result, even if the adversary becomes computationally unbounded afterward, they cannot learn anything beyond $f_i(m) \in [n]$. In Section 4.4, assuming the existence of public key encryption, we have constructed bounded collusion-resistant functional encryption, which is a weaker variant of collusion-resistant functional encryption. The key difference lies in the number of secret keys a receiver is allowed to obtain: it is fixed in

advance. If a receiver obtains more keys than this predetermined limit, security is no longer guaranteed.

In Section 4.5, we have constructed a certified everlasting secure variant of compute-and-compare obfuscation from standard compute-and-compare obfuscation and fully homomorphic encryption. In this primitive, we can obfuscate a circuit P along with a lock value lock and m and obtain an obfuscated circuit $\tilde{P}$ and a verification key vk . The adversary of $\tilde{P}$ can issue a deletion certificate cert of $\tilde{P}$. Once the deletion is verified, even computationally unbounded adversaries cannot obtain information about P , lock , and m after that. In Section 4.6, we study a certified everlasting secure variant of predicate encryption from attribute-based encryption and certified everlasting secure compute-and-compare obfuscation. In the primitive, the adversary possessing the quantum ciphertext $\text{ct}_x(m)$ can receive polynomially many secret keys $\{\text{sk}_{f_i}\}_{i \in [n]}$ and obtain m if $f_i(x) = 1$ for some $i \in [n]$. Additionally, it includes a mechanism to verify whether the adversary has deleted the quantum ciphertext. Once the deletion is verified, the security becomes an information-theoretic guarantee. Consequently, even if the adversary becomes computationally unbounded afterward, they cannot learn anything about plaintext m and attribute x .

Bibliography

- [Aar09] Scott Aaronson. Quantum copy-protection and quantum money. In *Proceedings of the 2009 24th Annual IEEE Conference on Computational Complexity*, CCC '09, page 229–242, USA, 2009. IEEE Computer Society. (Cited on page [10](#).)
- [Aar18] Scott Aaronson. Shadow tomography of quantum states. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 325–338. ACM Press, June 2018. (Cited on page [10](#).)
- [ABDP15] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015. (Cited on page [13](#).)
- [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 [13](#).)
- [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 [83](#).)
- [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 [30](#), [35](#).)
- [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 [103](#).)
- [AK21] Prabhanjan Ananth and Fatih Kaleoglu. Unclonable encryption, revisited. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part I*, volume 13042 of *LNCS*, pages 299–329. Springer, Heidelberg, November 2021. (Cited on page [14](#).)
- [AKL⁺22] Prabhanjan Ananth, Fatih Kaleoglu, Xingjian Li, Qipeng Liu, and Mark Zhandry. On the feasibility of unclonable encryption, and more. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 212–241. Springer, Heidelberg, August 2022. (Cited on page [14](#), [15](#).)

- [AKL23] Prabhanjan Ananth, Fatih Kaleoglu, and Qipeng Liu. Cloning games: A general framework for unclonable primitives. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 66–98, Cham, 2023. Springer Nature Switzerland. (Cited on page [15](#).)
- [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 [58](#).)
- [ALS16] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016. (Cited on page [13](#).)
- [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 [27](#), [58](#).)
- [AQY22] Prabhanjan Ananth, Luowen Qian, and Henry Yuen. Cryptography from pseudorandom quantum states. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 208–236. Springer, Heidelberg, August 2022. (Cited on page [15](#).)
- [AR17] Shweta Agrawal and Alon Rosen. Functional encryption for bounded collusions, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 173–205. Springer, Heidelberg, November 2017. (Cited on page [13](#).)
- [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 [60](#), [144](#), [146](#).)
- [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 [13](#), [58](#), [104](#).)
- [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 [58](#).)
- [BB84] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. In *IEEE International Conference on Computers Systems and Signal Processing*, pages 175–179. IEEE, 1984. (Cited on page [10](#).)
- [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 [15](#), [65](#).)

- [BCQ23] Zvika Brakerski, Ran Canetti, and Luowen Qian. On the computational hardness needed for quantum cryptography. 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 24:1–24:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. (Cited on page [15](#).)
- [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 [15](#), [16](#), [30](#), [34](#), [59](#).)
- [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](#), [16](#), [31](#), [34](#), [36](#), [52](#).)
- [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 [12](#), [26](#).)
- [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 [27](#).)
- [BGK⁺24] James Bartusek, Vipul Goyal, Dakshita Khurana, Giulio Malavolta, Justin Raizes, and Bhaskar Roberts. Software with certified deletion. In *Advances in Cryptology – EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part IV*, page 85–111, Berlin, Heidelberg, 2024. Springer-Verlag. (Cited on page [18](#).)
- [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 [27](#).)
- [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 [10](#), [11](#), [28](#), [29](#), [32](#), [33](#), [34](#), [35](#), [36](#), [57](#).)
- [BJSW16] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for QMA. In Irit Dinur, editor, *57th FOCS*, pages 31–40. IEEE Computer Society Press, October 2016. (Cited on page [14](#), [16](#).)
- [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 [17](#), [18](#), [60](#), [62](#), [67](#), [68](#), [69](#), [71](#), [72](#), [73](#), [84](#), [86](#), [104](#), [107](#).)
- [BL20] Anne Broadbent and Sébastien Lord. Uncloneable quantum encryption via oracles. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. (Cited on page [10](#), [14](#).)
- [BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007. (Cited on page [12](#).)
- [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 [9](#), [10](#), [13](#), [58](#).)
- [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 [27](#).)
- [CCY21] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. A black-box approach to post-quantum zero-knowledge in constant rounds. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 315–345, Virtual Event, August 2021. Springer, Heidelberg. (Cited on page [14](#).)
- [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 [135](#).)
- [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 [135](#).)
- [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 [27](#).)
- [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 [18](#).)
- [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 [58](#).)
- [DFS04] Ivan Damgård, Serge Fehr, and Louis Salvail. Zero-knowledge proofs and string commitments withstanding quantum attacks. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 254–272. Springer, Heidelberg, August 2004. (Cited on page [14](#).)
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page [8](#), [9](#), [11](#).)

- [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 58.)
- [For87] Lance Fortnow. The complexity of perfect zero-knowledge (extended abstract). In Alfred Aho, editor, *19th ACM STOC*, pages 204–209. ACM Press, May 1987. (Cited on page 14, 16, 32, 34.)
- [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 18, 74, 135, 138, 139.)
- [GGH⁺13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013. (Cited on page 13.)
- [GGH⁺13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 479–499. Springer, Heidelberg, August 2013. (Cited on page 12.)
- [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 58, 60.)
- [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 25.)
- [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 64.)
- [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 18.)
- [GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996. (Cited on page 14.)
- [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 13, 28, 58, 68, 69.)
- [GLSV21] Alex B. Grilo, Huijia Lin, Fang Song, and Vinod Vaikuntanathan. Oblivious transfer is in MiniQCrypt. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 531–561. Springer, Heidelberg, October 2021. (Cited on page 15.)

- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *14th ACM STOC*, pages 365–377. ACM Press, May 1982. (Cited on page 12.)
- [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 58.)
- [GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989. (Cited on page 32.)
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *CRYPTO’86*, volume 263 of *LNCS*, pages 171–185. Springer, Heidelberg, August 1987. (Cited on page 9, 10, 14.)
- [GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7:1–32, 1994. (Cited on page 54.)
- [Got04] Daniel Gottesman. Uncloneable encryption, 2004. (Cited on page 10, 14.)
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309. (Cited on page 12.)
- [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 21.)
- [GSV98] Oded Goldreich, Amit Sahai, and Salil P. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *30th ACM STOC*, pages 399–408. ACM Press, May 1998. (Cited on page 34.)
- [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 27.)
- [GSY19] Alex Bredariol Grilo, William Slofstra, and Henry Yuen. Perfect zero knowledge for quantum multiprover interactive proofs. In David Zuckerman, editor, *60th FOCS*, pages 611–635. IEEE Computer Society Press, November 2019. (Cited on page 14.)
- [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 13, 58, 63, 66, 83, 85, 96, 101, 103, 104.)

- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013. (Cited on page 12.)
- [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 26.)
- [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 12, 59.)
- [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 24.)
- [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 28, 34, 57.)
- [HMNY22a] Taiga Hiroka, Tomoyuki Morimae, Ryo Nishimaki, and Takashi Yamakawa. Certified everlasting functional encryption. Cryptology ePrint Archive, Paper 2022/969, 2022. (Cited on page 18.)
- [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 57, 58, 63, 159, 160, 162.)
- [HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 1–10. ACM Press, June 2007. (Cited on page 13.)
- [IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th FOCS*, pages 230–235. IEEE Computer Society Press, October / November 1989. (Cited on page 13.)
- [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 135.)
- [JLS18] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. Pseudorandom quantum states. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 126–152. Springer, Heidelberg, August 2018. (Cited on page 15.)
- [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 58.)

- [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 58.)
- [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 63, 136, 167.)
- [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 59.)
- [Kre21] William Kretschmer. Quantum pseudorandomness and classical complexity. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. (Cited on page 15.)
- [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 59.)
- [KT20] Srijita Kundu and Ernest Tan. Composably secure device-independent encryption with certified deletion. *arXiv*, 2011.12704, 2020. (Cited on page 57.)
- [LC97] Hoi-Kwong Lo and H. F. Chau. Is quantum bit commitment really possible? *Phys. Rev. Lett.*, 78:3410–3413, 1997. (Cited on page 10, 34, 57.)
- [LC19] Ching-Yi Lai and Kai-Min Chung. Quantum encryption and generalized shannon impossibility. *Designs, Codes and Cryptography*, 87(9):1961–1972, January 2019. (Cited on page 10.)
- [LMS22] Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited or: How to do quantum rewinding undetectably. In *63rd FOCS*, pages 851–859. IEEE Computer Society Press, October / November 2022. (Cited on page 14.)
- [LMW24] Alex Lombardi, Fermi Ma, and John Wright. A one-query lower bound for unitary synthesis and breaking quantum cryptography. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 979–990, New York, NY, USA, 2024. Association for Computing Machinery. (Cited on page 15.)
- [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 23, 64.)
- [LS19] Alex Lombardi and Luke Schaeffer. A note on key agreement and non-interactive commitments. Cryptology ePrint Archive, Report 2019/279, 2019. <https://eprint.iacr.org/2019/279>. (Cited on page 22.)
- [May97] Dominic Mayers. Unconditionally secure quantum bit commitment is impossible. *Phys. Rev. Lett.*, 78:3414–3417, 1997. (Cited on page 10, 34, 57.)

- [MW18] Sanketh Menda and John Watrous. Oracle separations for quantum statistical zero-knowledge. *arXiv:1801.08967*, 2018. (Cited on page 10, 14, 16, 32, 33, 57.)
- [MY22] Tomoyuki Morimae and Takashi Yamakawa. Quantum commitments and signatures without one-way functions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2022. (Cited on page 15.)
- [Nao90] Moni Naor. Bit commitment using pseudo-randomness. In Gilles Brassard, editor, *CRYPTO’89*, volume 435 of *LNCS*, pages 128–136. Springer, Heidelberg, August 1990. (Cited on page 10.)
- [Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, January 1991. (Cited on page 13.)
- [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 57.)
- [Rab79] Michael O. Rabin. Digital signatures and public key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology, January 1979. (Cited on page 12.)
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th ACM STOC*, pages 84–93. ACM Press, May 2005. (Cited on page 12.)
- [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 21.)
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990. (Cited on page 10.)
- [RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO’91*, volume 576 of *LNCS*, pages 433–444. Springer, Heidelberg, August 1992. (Cited on page 12.)
- [RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978. (Cited on page 9, 12.)
- [Sha49] C. E. Shannon. Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949. (Cited on page 8, 9.)
- [Sha79] Adi Shamir. How to share a secret. *cacm*, 22(11):612–613, nov 1979. (Cited on page 96.)
- [Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994. (Cited on page 12.)
- [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 28.)

- [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 [64](#), [86](#).)
- [SW05] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. (Cited on page [9](#), [10](#), [12](#).)
- [Unr13] Dominique Unruh. Everlasting multi-party computation. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 380–397. Springer, Heidelberg, August 2013. (Cited on page [32](#), [33](#).)
- [Unr14] Dominique Unruh. Quantum position verification in the random oracle model. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2014. (Cited on page [10](#).)
- [Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015. (Cited on page [30](#), [35](#), [38](#), [57](#), [63](#), [160](#), [162](#), [163](#), [165](#).)
- [VDG98] Jeroen Van De Graaf. *Towards a formal definition of security for quantum protocols*. PhD thesis, CAN, 1998. AAINQ35648. (Cited on page [14](#).)
- [Wat02] John Watrous. Limits on the power of quantum statistical zero-knowledge. In *43rd FOCS*, pages 459–470. IEEE Computer Society Press, November 2002. (Cited on page [34](#).)
- [Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009. (Cited on page [14](#), [30](#), [33](#).)
- [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 [13](#).)
- [Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, jan 1983. (Cited on page [10](#).)
- [Win99] Andreas J. Winter. Coding theorem and strong converse for quantum channels. *IEEE Trans. Inf. Theory*, 45(7):2481–2485, 1999. (Cited on page [70](#).)
- [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 [12](#), [28](#), [58](#), [68](#), [69](#).)
- [Yan22] Jun Yan. General properties of quantum bit commitments (extended abstract). In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 628–657, Cham, 2022. Springer Nature Switzerland. (Cited on page [15](#).)
- [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 [64](#), [175](#).)
- [Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indistinguishability. 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 [30](#), [43](#).)

Appendix A

Appendix

A.1 Omitted Proofs for Collusion-Resistant FE

We prove the adaptive security of our collusion-resistant scheme CED in Section 4.3.3. That is, we show

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

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

Hyb₀: 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 $(\text{fe.MPK}, \text{fe.MSK}) \leftarrow \text{FE.Setup}(1^\lambda)$, sets $\text{MPK} := \text{fe.MPK}$ and $\text{MSK} := \text{fe.MSK}$, 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 $\text{fe.sk}_{g[f]} \leftarrow \text{FE.KeyGen}(\text{fe.MSK}, g[f])$ and returns $\text{sk}_f = \text{fe.sk}_{g[f]}$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $(m^{(0)}, m^{(1)})$ to the challenger.¹ 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 $(ct, vk) \leftarrow \text{Enc}(\text{MPK}, m^{(0)})$. 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 $v_{i,j,b} \leftarrow \text{PRG}(u_{i,j,b})$ 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

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

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 $j \in [\lambda]$.

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

(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 $\text{fe.ct} \leftarrow \text{FE.Enc}(\text{fe.MPK}, V \parallel \theta_1 \parallel \dots \parallel \theta_{2n+1} \parallel \beta_1 \parallel \dots \parallel \beta_{2n+1})$.

(f) Set $ct = (\text{fe.ct}, \bigotimes_{i \in [2n+1], j \in [\lambda]} |\psi_{i,j}\rangle)$ and $\text{vk} = (U, (z_i, \theta_i)_{i \in [2n+1]})$.

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

5. If $\mathcal{A}$ sends a certificate of deletion cert , the challenger computes $\text{Vrfy}(\text{vk}_0, \text{cert})$ 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 .

Hyb₁: This is identical to Hyb₀ except that $v_{i,j,1 \oplus z_{i,j}}$ is uniformly chosen from $\{0, 1\}^{2\lambda}$ instead of being set to be $\text{PRG}(u_{i,j,1 \oplus z_{i,j}})$ for all $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$.

Hyb₂: This is identical to Hyb₁ except that $(\beta_i)_{i \in [2n+1]}$ is generated as

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

Hyb₃: This is identical to Hyb₂ except that $v_{i,j,b}$ is set to be $\text{PRG}(u_{i,j,b})$ for all $i \in [2n+1]$, $j \in [\lambda]$, and $b \in \{0, 1\}$.

Note that Hyb₃ is identical to the original adaptive security experiment where the challenge bit is set to be 1. Thus, we only have to prove

$$|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_3 = 1]| \leq \text{negl}(\lambda). \quad (\text{A.1})$$

We prove Equation (A.1) by the following lemmata.

Lemma A.1.1. *If PRG is a secure PRG,*

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

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 Hyb₀, Lemma A.1.1 directly follows from the security of PRG. Note that we can simulate $\text{Vrfy}(\text{vk}_0, \text{cert})$ where $\text{cert} = (c_{i,j}, d_{i,j})_{i,j}$ since we need $\{z_{i,j}\}_{i,j}$ and $\{u_{i,j,b}\}_{i,j,b}$ such that $\theta_{i,j} = 1$ for verification. $\square$

Lemma A.1.2. *If FE is adaptively secure,*

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

Proof. For each $i \in [2n+1]$ and $j \in [\lambda]$ such that $\theta_{i,j} = 0$, there is no u such that $\text{PRG}(u) = v_{i,j,1 \oplus z_{i,j}}$ 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 $\Pr[\text{Good}] \geq 1 - (2n+1)\lambda 2^{-\lambda} = 1 - \text{negl}(\lambda)$. We prove that whenever *Good* occurs, we have

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

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.1.2 directly follows from the adaptive security of FE.

Below, we prove Equation (A.2). We consider the following two cases.

- If $\text{PRG}(u_{i,j}) = v_{i,j,b_{i,j}}$ 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 (A.2) is equal to $f(m^{(0)})$ and the RHS of Equation (A.2) 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 (A.2) are equal to $f(m^{(0)}) = f(m^{(1)})$.
- Otherwise, both sides of Equation (A.2) are equal to $\perp$.

In either case, Equation (A.2) holds. Note that we can simulate $\text{Vrfy}(\text{vk}_b, \text{cert})$ where $\text{cert} = (c_{i,j}, d_{i,j})_{i,j}$ 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.1.2. $\square$

Lemma A.1.3. *If PRG is a secure PRG,*

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

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.1.1 directly follows from the security of PRG. Note that we can simulate $\text{Vrfy}(\text{vk}_1, \text{cert})$ where $\text{cert} = (c_{i,j}, d_{i,j})_{i,j}$ since we need $\{z_{i,j}\}_{i,j}$ and $\{u_{i,j,b}\}_{i,j,b}$ such that $\theta_{i,j} = 1$ for verification. $\square$

A.2 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 essential building block. Our adaptively secure public-slot PKFE scheme is presented in Appendix A.2.4.

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 A.2.3.

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 A.2.1 and A.2.2.

A.2.1 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 A.2.1 (Secret-Key RNCE (Syntax)). A secret-key NCE scheme is a tuple of PPT algorithms (KeyGen, Enc, Dec, Fake, Reveal) with plaintext space $\mathcal{M}$.

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

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

$\text{Dec}(\text{dk}, \text{ct}) \rightarrow m' \text{ or } \perp$: The decryption algorithm takes as input dk and ct and outputs a plaintext m' or $\perp$.

$\text{Fake}(\text{ek}, \text{aux}) \rightarrow \tilde{\text{ct}}$: The fake encryption algorithm takes dk and aux , and outputs a fake ciphertext $\tilde{\text{ct}}$.

$\text{Reveal}(\text{ek}, \text{aux}, \tilde{\text{ct}}, m) \rightarrow \tilde{\text{dk}}$: The reveal algorithm takes ek , aux , $\tilde{\text{ct}}$ and m , and outputs a fake secret key $\tilde{\text{dk}}$.

Definition A.2.2 (Correctness of secret-key NCE). There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,

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

Definition A.2.3 (Receiver Non-Committing (RNC) Security for SKE). A secret-key NCE scheme is RNC secure if it satisfies the following. Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal})$ be a secret-key NCE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sk-rec nc}}(\lambda, b)$.

1. The challenger computes $(\text{ek}, \text{dk}, \text{aux}) \leftarrow \text{KeyGen}(1^\lambda)$ 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 $\text{ct} \leftarrow \text{Enc}(\text{ek}, m)$ 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 $ct \leftarrow \text{Enc}(ek, m)$ and returns (ct, dk) to $\mathcal{A}$.
 - If $b = 1$, the challenger generates $\tilde{ct} \leftarrow \text{Fake}(ek, aux)$ and $\tilde{dk} \leftarrow \text{Reveal}(ek, aux, \tilde{ct}, m)$ and returns $(\tilde{ct}, \tilde{dk})$ to $\mathcal{A}$.
5. Again $\mathcal{A}$ can send encryption queries.
6. $\mathcal{A}$ outputs $b' \in \{0, 1\}$.

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

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

Theorem A.2.4 ([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 A.2.5 (Public-Key FE (Syntax)). *A public-key functional encryption (PKFE) scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ 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.*

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

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

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

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

We require that a PKFE scheme satisfies correctness defined below.

Definition A.2.6 (Correctness of PKFE). *There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ and $f \in \mathcal{F}$*

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

Definition A.2.7 (Selective Security of PKFE). *Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a PKFE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-ind}}(\lambda, b)$ 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 $(\text{MPK}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$, computes $\text{CT} \leftarrow \text{Enc}(\text{MPK}, m_b)$, and sends (MPK, 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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $\perp$.

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

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

Definition A.2.8 (Secret-Key FE (Syntax)). A secret-key functional encryption (SKFE) scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ 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.

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

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

$\text{Enc}(\text{MSK}, m) \rightarrow \text{CT}$: The encryption algorithm takes MSK and $m \in \mathcal{M}$ as input, and outputs a ciphertext $\text{CT} \in \mathcal{C}$.

$\text{Dec}(\text{sk}_f, \text{CT}) \rightarrow y \text{ or } \perp$: The decryption algorithm takes sk_f and CT as input, and outputs y or $\perp$.

We require that an SKFE scheme satisfies correctness defined below.

Definition A.2.9 (Correctness of SKFE). There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$ and $f \in \mathcal{F}$

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

Definition A.2.10 (Adaptive Single-Key Single-Ciphertext Security of SKFE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be an SKFE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ada-1key-1ct}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.

1. The challenger runs $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$.
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 $\text{sk}_f \leftarrow \text{KeyGen}(\text{MSK}, f)$, 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 $\text{CT} \leftarrow \text{Enc}(\text{MSK}, m_b)$, 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 $\perp$.

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

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

Definition A.2.11 (Selective Single-Key Function Privacy of SKFE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be an SKFE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-1key-fp}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.

1. The challenger sends $(m_{1,0}, m_{1,1}), \dots, (m_{q,0}, m_{q,1}) \in \mathcal{M}^{2q}$ to the challenger.
2. The challenger runs $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$, computes $\text{CT}_i \leftarrow \text{Enc}(\text{MSK}, m_{i,b})$ for all $i \in [q]$, and returns $(\text{CT}_1, \dots, \text{CT}_q)$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends $(f_0, f_1) \in \mathcal{F}^2$ to the challenger. The challenger computes $\text{sk}_{f_b} \leftarrow \text{KeyGen}(\text{MSK}, f_b)$, 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 $\perp$.

We say that Σ is selectively single-key function private if for any QPT adversary $\mathcal{A}$ it holds that

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

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 A.2.12 (2-input FE (Syntax)). A 2-input FE scheme for a class $\mathcal{F}$ of functions is a tuple of PPT algorithms $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ 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.

$\text{Setup}(1^\lambda) \rightarrow \text{MSK}$: The setup algorithm takes the security parameter 1^λ as input, and outputs a master secret key $\text{MSK} \in \mathcal{MSK}$ and two encryption keys EK_1 and EK_2 .

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

$\text{Enc}(\text{EK}_i, x) \rightarrow \text{CT}_i$: The encryption algorithm takes EK_i and $x \in \mathcal{M}$ as input, and outputs a ciphertext $\text{CT}_i \in \mathcal{C}$.

$\text{Dec}(\text{sk}_f, \text{CT}_1, \text{CT}_2) \rightarrow z$ or $\perp$: The decryption algorithm takes sk_f and $(\text{CT}_1, \text{CT}_2)$ as input, and outputs y or $\perp$.

We require that a 2-input FE scheme satisfies correctness defined below.

Definition A.2.13 (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 $f \in \mathcal{F}$*

$$\Pr \left[z \neq f(x, y) \mid \begin{array}{l} (\text{MSK}, \text{EK}_1, \text{EK}_2) \leftarrow \text{Setup}(1^\lambda) \\ \text{sk}_f \leftarrow \text{KeyGen}(\text{MSK}, f) \\ \text{CT}_1 \leftarrow \text{Enc}(\text{EK}_1, x), \text{CT}_2 \leftarrow \text{Enc}(\text{EK}_2, y) \\ z \leftarrow \text{Dec}(\text{sk}_f, \text{CT}_1, \text{CT}_2) \end{array} \right] \leq \text{negl}(\lambda).$$

Definition A.2.14 ((1, 1)-sel-ind Security of 2-FE). *Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a 2-input FE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-1ct}}(\lambda, b)$ 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 $(\text{MSK}, \text{EK}_1, \text{EK}_2) \leftarrow \text{Setup}(1^\lambda)$, computes $\text{CT}_1 \leftarrow \text{Enc}(\text{EK}_1, x_b)$ and $\text{CT}_2 \leftarrow \text{Enc}(\text{EK}_2, y_b)$, and sends $(\text{EK}_2, \text{CT}_1, \text{CT}_2)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ can send a key query $f_i \in \mathcal{F}$ to the challenger. The challenger computes $\text{sk}_{f_i} \leftarrow \text{KeyGen}(\text{MSK}, f_i)$, 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 $\perp$.*

We say that Σ is (1, 1)-sel-ind secure if for any QPT adversary $\mathcal{A}$ it holds that

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

Here, (1, 1) means that $\mathcal{A}$ is given one encryption key EK_2 and one challenge ciphertext vector $(\text{CT}_1, \text{CT}_2)$.

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 A.2.15 ([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.

A.2.2 Variants of Security Definitions

We can consider the secret-key variant of Definition 4.3.9, 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 4.3.10. We omit syntax and correctness for public-slot SKFE.

Below, we introduce variants of security definitions for public-slot SKFE.

Definition A.2.16 (Single-Key Security of Public-Slot SKFE). *Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be a public-slot SKFE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ada-1ct}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger runs $\text{MSK} \leftarrow \text{Setup}(1^\lambda)$.*

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 $\text{sk}_{f_\ell} \leftarrow \text{KeyGen}(\text{MSK}, f_\ell)$, 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 $\text{CT} \leftarrow \text{Enc}(\text{MSK}, m_b)$ 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 $f_\ell(m_0, \text{pub}) = f_\ell(m_1, \text{pub})$ holds for all key queries f_ℓ and public inputs $\text{pub} \in \mathcal{P}$, 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

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

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 $\text{Adv}_{\Sigma, \mathcal{A}}^{\text{sel-1ct}}(\lambda)$ and $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-1ct}}(\lambda)$, 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 $\text{Adv}_{\Sigma, \mathcal{A}}^{\text{xxx-1key-1ct}}(\lambda)$ and $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{xxx-1key-1ct}}(\lambda)$, respectively, where $\text{xxx} \in \{\text{sel}, \text{ada}\}$.

A.2.3 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 1selFE.

Ingredients.

- $(1, 1)$ -sel-ind secure 2-input FE $2\text{FE} = 2\text{FE}.$ (Setup, KeyGen, Enc, Dec) for all polynomial-size circuits.

Scheme description. Our scheme $1\text{selFE} = 1\text{selFE}.$ (Setup, KeyGen, Enc, Dec) is as follows.

$1\text{selFE}.\text{Setup}(1^\lambda)$:

1. Generate $(2\text{fe.msk}, 2\text{fe.ek}_1, 2\text{fe.ek}_2) \leftarrow 2\text{FE}.\text{Setup}(1^\lambda)$.
2. Output $\text{MSK} := (2\text{fe.msk}, 2\text{fe.ek}_1, 2\text{fe.ek}_2)$.

$1\text{selFE}.\text{KeyGen}(\text{MSK}, f)$:

1. Parse $\text{MSK} = (2\text{fe.msk}, 2\text{fe.ek}_1, 2\text{fe.ek}_2)$.
2. Generate $2\text{fe.SK}_f \leftarrow 2\text{FE}.\text{KeyGen}(2\text{fe.msk}, f)$.
3. Output $\text{SK}_f := 2\text{fe.sk}_f$.

$1\text{selFE}.\text{Enc}(\text{MSK}, m)$:

1. Parse $\text{MSK} = (2\text{fe.msk}, 2\text{fe.ek}_1, 2\text{fe.ek}_2)$.
2. Generate $2\text{fe.ct}_1 \leftarrow 2\text{FE}.\text{Enc}(2\text{fe.ek}_1, m)$.

3. Output $CT := (2fe.ct_1, 2fe.ek_2)$.

$1selFE.Dec(SK_f, CT, pub)$:

1. Parse $SK_f = 2fe.sk_f$ and $CT = (2fe.ct_1, 2fe.ek_2)$.
2. Compute $2fe.ct_2 \leftarrow 2FE.Enc(2fe.ek_2, pub)$.
3. Compute and output $y := 2FE.Dec(2fe.sk_f, 2fe.ct_1, 2fe.ct_2)$.

Correctness. It is easy to see correctness holds due to correctness of 2FE.

Theorem A.2.17. *If 2FE is $(1, 1)$ -sel-ind secure 2-input FE for all polynomial-size circuits, 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 $\text{Exp}_{1selFE, \mathcal{A}}^{\text{sel-1ct}}(\lambda, b)$ denote the selective single-ciphertext security of public-slot SKFE. We construct an algorithm $\mathcal{B}$ that breaks $(1, 1)$ -sel-ind security of 2FE by using an adversary $\mathcal{A}$ that breaks selectively single-ciphertext security of 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 $(2fe.ek_2, 2fe.ct_1, 2fe.ct_2)$ from its challenger, $\mathcal{B}$ sets $CT := (2fe.ct_1, 2fe.ek_2)$ 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 $2fe.sk_{f_i} \leftarrow 2FE.KeyGen(2fe.msk, f_i)$, and passes $SK_{f_i} := 2fe.sk_{f_i}$ 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 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 2FE since $\mathcal{B}$ received $2fe.ek_2$. It is easy to see the following.

- If $2fe.ct_1 \leftarrow 2FE.Enc(2fe.msk, x_0)$ and $2fe.ct_2 \leftarrow 2FE.Enc(2fe.msk, y)$, $\mathcal{B}$ perfectly simulates $\text{Exp}_{1selFE, \mathcal{A}}^{\text{sel-1ct}}(\lambda, 0)$.
- If $2fe.ct_1 \leftarrow 2FE.Enc(2fe.msk, x_1)$ and $2fe.ct_2 \leftarrow 2FE.Enc(2fe.msk, y)$, $\mathcal{B}$ perfectly simulates $\text{Exp}_{1selFE, \mathcal{A}}^{\text{sel-1ct}}(\lambda, 1)$.

Thus, if $\mathcal{A}$ distinguishes $\text{Exp}_{1selFE, \mathcal{A}}^{\text{sel-1ct}}(\lambda, 0)$ from $\text{Exp}_{1selFE, \mathcal{A}}^{\text{sel-1ct}}(\lambda, 1)$, $\mathcal{B}$ distinguishes $\text{Exp}_{2FE, \mathcal{B}}^{\text{sel-1ct}}(\lambda, 0)$ from $\text{Exp}_{2FE, \mathcal{B}}^{\text{sel-1ct}}(\lambda, 1)$. This completes the proof. $\square$

Adaptively single-key single-ciphertext public-slot SKFE. Next, we present our adaptively single-key single-ciphertext public-slot SKFE scheme 1adaFE.

Ingredients.

- Selectively single-key single-ciphertext public-slot SKFE $1\text{selfFE} = 1\text{selfFE}.\text{(Setup, KeyGen, Enc, Dec)}$ for all polynomial-size circuits.
- Receiver non-committing encryption $\text{NCE} = \text{NCE}.\text{(KeyGen, Enc, Dec, Fake, Reveal)}$.

Scheme description. Our scheme $1\text{adaFE} = 1\text{adaFE}.\text{(Setup, KeyGen, Enc, Dec)}$ is as follows.

$1\text{adaFE}.\text{Setup}(1^\lambda)$:

1. Generate $\text{sel.msk} \leftarrow 1\text{selfFE}.\text{Setup}(1^\lambda)$.
2. Generate $(\text{nce.ek}, \text{nce.dk}, \text{nce.aux}) \leftarrow \text{NCE}.\text{KeyGen}(1^\lambda)$.
3. Output $\text{MSK} := (\text{sel.msk}, \text{nce.ek}, \text{nce.dk}, \text{nce.aux})$.

$1\text{adaFE}.\text{KeyGen}(\text{MSK}, f)$:

1. Parse $\text{MSK} = (\text{sel.MSK}, \text{nce.ek}, \text{nce.dk}, \text{nce.aux})$.
2. Generate $\text{sel.sk}_f \leftarrow 1\text{selfFE}.\text{KeyGen}(\text{sel.msk}, f)$.
3. Generate $\text{nce.ct} \leftarrow \text{NCE}.\text{Enc}(\text{nce.ek}, \text{fe.sk}_f)$.
4. Output $\text{SK}_f := \text{nce.ct}$.

$1\text{adaFE}.\text{Enc}(\text{MSK}, m)$:

1. Parse $\text{MSK} = (\text{sel.msk}, \text{nce.ek}, \text{nce.dk}, \text{nce.aux})$.
2. Generate $\text{sel.ct} \leftarrow 1\text{selfFE}.\text{Enc}(\text{sel.msk}, m)$.
3. Output $\text{CT} := (\text{sel.ct}, \text{nce.dk})$.

$1\text{adaFE}.\text{Dec}(\text{SK}_f, \text{CT}, \text{pub})$:

1. Parse $\text{SK}_f = \text{nce.ct}$ and $\text{CT} = (\text{sel.ct}, \text{nce.dk})$.
2. Compute $\text{sk}'_f \leftarrow \text{NCE}.\text{Dec}(\text{nce.dk}, \text{nce.ct})$.
3. Compute and output $y := 1\text{selfFE}.\text{Dec}(\text{sk}'_f, \text{sel.ct}, \text{pub})$.

Correctness. It is easy to see correctness holds due to correctness of 1selfFE and NCE .

Theorem A.2.18. *If 1selfFE is selectively single-key single-ciphertext secure public-slot SKFE for all polynomial-size circuits and NCE is RNC secure, 1adaFE is adaptively single-key single-ciphertext public-slot SKFE for all polynomial-size circuits.*

Proof. Let $\text{Hyb}_0(b)$ denote $\text{Exp}_{1\text{adaFE}, \mathcal{A}}^{\text{ada-1key-1ct}}(\lambda, b)$. We define a hybrid game $\text{Hyb}_1(b)$ as follows.

$\text{Hyb}_1(b)$: This is the same as $\text{Exp}_{1\text{adaFE}, \mathcal{A}}^{\text{ada-1key-1ct}}(\lambda, b)$ except that:

1. if $\mathcal{A}$ sends a key query f before an encryption query (m_0, m_1) , we generate $\text{nce.ct} \leftarrow \text{NCE}.\text{Fake}(\text{nce.ek}, \text{nce.aux})$ instead of $\text{nce.ct} \leftarrow \text{NCE}.\text{Enc}(\text{nce.ek}, \text{sel.sk}_f)$ and return $\text{SK}_f := \text{nce.ct}$ for the key query.

2. when $\mathcal{A}$ sends an encryption query (m_0, m_1) after the key query f above, we generate $\text{sel.ct} \leftarrow \text{1selFE.Enc}(\text{sel.msk}, m_b)$, $\text{sel.sk}_f \leftarrow \text{1selFE.KeyGen}(\text{sel.msk}, f)$, and $\text{nce.dk} \leftarrow \text{NCE.Reveal}(\text{nce.ek}, \text{nce.aux}, \text{nce.ct}, \text{sel.sk}_f)$, and return $\text{CT} := (\text{sel.ct}, \text{nce.dk})$ for the encryption query.

First, we show the following.

Proposition A.2.19. *It holds $|\Pr[\text{Hyb}_0(b) = 1] - \Pr[\text{Hyb}_1(b) = 1]| \leq \text{negl}(\lambda)$ 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 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 $\text{sel.msk} \leftarrow \text{1selFE.Setup}(1^\lambda)$.
2. When $\mathcal{A}$ sends a key query f , $\mathcal{B}$ generates $\text{sel.sk}_f \leftarrow \text{1selFE.KeyGen}(\text{sel.msk}, f)$, sends sel.sk_f to its challenger, and receives $(\text{nce.ct}^*, \text{nce.dk}^*)$. $\mathcal{B}$ passes $\text{SK}_f := \text{nce.ct}^*$ to $\mathcal{A}$.
3. After the key query f above, for an encryption query (m_0, m_1) , $\mathcal{B}$ generates $\text{sel.ct} \leftarrow \text{1selFE.Enc}(\text{sel.msk}, m_b)$ and returns $\text{CT} := (\text{sel.ct}, \text{nce.dk}^*)$ to $\mathcal{A}$.
4. When $\mathcal{A}$ outputs b' , $\mathcal{B}$ outputs b' .

It is easy to see the following.

- If $\text{nce.ct}^* \leftarrow \text{NCE.Enc}(\text{nce.ek}, \text{sel.sk}_f)$ and $\text{nce.dk}^* = \text{nce.dk}$ where $(\text{nce.ek}, \text{nce.dk}, \text{nce.aux}) \leftarrow \text{NCE.KeyGen}(1^\lambda)$, $\mathcal{B}$ perfectly simulates $\text{Hyb}_0(b)$.
- If $\text{nce.ct}^* := \text{nce.ct} \leftarrow \text{NCE.Fake}(\text{nce.ek}, \text{nce.aux})$ and $\text{nce.dk}^* \leftarrow \text{NCE.Reveal}(\text{nce.ek}, \text{nce.aux}, \text{nce.ct}, \text{sel.sk}_f)$, $\mathcal{B}$ perfectly simulates $\text{Hyb}_1(b)$.

Thus, if $\mathcal{A}$ distinguishes $\text{Hyb}_0(b)$ from $\text{Hyb}_1(b)$, $\mathcal{B}$ distinguishes $\text{Exp}_{\text{NCE}, \mathcal{B}}^{\text{sk-rec-nc}}(\lambda, b)$.

Next, we show the following.

Proposition A.2.20. *It holds $|\Pr[\text{Hyb}_1(0) = 1] - \Pr[\text{Hyb}_1(1) = 1]| \leq \text{negl}(\lambda)$ if 1selFE is selectively single-key single-ciphertext secure.*

We construct an algorithm $\mathcal{B}$ that breaks selective single-key single-ciphertext security of 1selFE by using an adversary $\mathcal{A}$ that breaks adaptive single-key single-ciphertext security of 1adaFE. $\mathcal{B}$ does the following.

1. First, $\mathcal{B}$ generates $(\text{nce.ek}, \text{nce.dk}, \text{nce.aux}) \leftarrow \text{NCE.KeyGen}(1^\lambda)$.
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 $\text{nce.ct} \leftarrow \text{NCE.Fake}(\text{nce.ek}, \text{nce.aux})$ passes $\text{SK}_f := \text{nce.ct}$ 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 sel.ct^* . Then, $\mathcal{B}$ sends f to its challenger and receives $\text{sel.sk}_f \leftarrow \text{1selFE}(\text{sel.msk}, f)$. Finally, $\mathcal{B}$ generates $\text{nce.dk} \leftarrow \text{NCE.Reveal}(\text{nce.ek}, \text{nce.dk}, \text{nce.ct}, \text{sel.sk}_f)$ and sends $\text{CT} := (\text{sel.ct}^*, \text{nce.dk})$ 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 sel.ct^* , and returns $\text{CT} := (\text{sel.ct}^*, \text{nec.dk})$ 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 $\text{sel.sk}_f \leftarrow \text{1selFE.KeyGen}(\text{sel.msk}, f)$. $\mathcal{B}$ returns $\text{SK}_f := \text{nec.ct} \leftarrow \text{NCE.Enc}(\text{nec.ek}, \text{sel.sk}_f)$ 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-slot SKFE 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 1selFE. In addition, it is easy to see the following.

- If $\text{sel.ct}^* \leftarrow \text{1selFE.Enc}(\text{sel.msk}, m_0)$, $\mathcal{B}$ perfectly simulates $\text{Hyb}_1(0)$.
- If $\text{sel.ct}^* \leftarrow \text{1selFE.Enc}(\text{sel.msk}, m_1)$, $\mathcal{B}$ perfectly simulates $\text{Hyb}_1(1)$.

Thus, if $\mathcal{A}$ distinguishes $\text{Hyb}_1(0)$ from $\text{Hyb}_1(1)$, $\mathcal{B}$ distinguishes $\text{Exp}_{\text{1selFE}, \mathcal{B}}^{\text{sel-1key-1ct}}(\lambda, 0)$ from $\text{Exp}_{\text{1selFE}, \mathcal{B}}^{\text{sel-1key-1ct}}(\lambda, 1)$. Therefore, we obtain $|\Pr[\text{Hyb}_1(0) = 1] - \Pr[\text{Hyb}_1(1) = 1]| \leq \text{negl}(\lambda)$, which is our goal. $\square$

A.2.4 Adaptively Secure Public-Slot PKFE Scheme

Note that the construction in this section is basically the same as that by Ananth and Sahai [AS16] except that we use single-key single-ciphertext public-slot SKFE as a building block instead of single-key single-ciphertext standard SKFE.

Ingredients.

- Selectively secure PKFE $\text{PKFE} = \text{PKFE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for all polynomial-size circuits.
- Selectively single-key function private SKFE $\text{1KeySKFE} = \text{1KeySKFE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for polynomial-size circuits.
- Adaptively single-key single-ciphertext public-slot SKFE $\text{SKFE} = \text{SKFE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ for polynomial-size circuits.
- A PRF $\text{PRF} : \mathcal{K} \times \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$.
- SKE with pseudorandom ciphertext $\text{SKE} = \text{SKE}(\text{Setup}, \text{Enc}, \text{Dec})$.

Scheme description. The adaptively secure public-slot PKFE scheme $\text{FE} = \text{FE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ is as follows.

Setup (1^λ) :

1. Generate $(\text{pkfe.MPK}, \text{pkfe.MSK}) \leftarrow \text{PKFE.Setup}(1^\lambda)$.
2. Output $\text{MPK} := \text{pkfe.MPK}$ and $\text{MSK} := \text{pkfe.MSK}$.

KeyGen (msk, f) :

$$g[f, C_{\text{ske}}, \tau]$$

Input: 1keyskfe.MSK, K, ske.SK, β

1. Parse $\tau = (\tau_0 \| \tau_1 \| \tau_2 \| \tau_3)$.
2. If $\beta = 0$ then
 - Compute $R_i \leftarrow \text{PRF}(K, \tau_i)$ for $i \in \{0, 1, 2, 3\}$.
 - Generate $\text{skfe.MSK} \leftarrow \text{SKFE.Setup}(1^\lambda; R_0)$.
 - Compute $\text{skfe.sk}_f \leftarrow \text{SKFE.KeyGen}(\text{skfe.MSK}, f; R_1)$.
 - Compute $1\text{keyskfe.ct} \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}, (\text{skfe.MSK}, R_2, 0); R_3)$.
 - Output $(\text{skfe.sk}_f, 1\text{keyskfe.ct})$.
3. Else,
 - Compute $(\text{skfe.sk}_f, 1\text{keyskfe.ct}) \leftarrow \text{SKE.Dec}(\text{ske.SK}, C_{\text{ske}})$.
 - Output $(\text{skfe.sk}_f, 1\text{keyskfe.ct})$.

Figure A.1: The description of the function $g[f, C_E, \tau]$

1. Parse $\text{MSK} = \text{pkfe.MSK}$.
2. Sample $C_{\text{ske}} \leftarrow \{0, 1\}^{\ell_{\text{ske}}(\lambda)}$ where $\ell_{\text{ske}}(\lambda)$ is the length of a SKE ciphertext that encrypts a string of length $\ell_{\text{skfe}}(\lambda) + \ell_{1\text{keyskfe}}(\lambda)$. We denote $\ell_{\text{skfe}}(\lambda)$ by the length of a SKFE secret key and $\ell_{1\text{keyskfe}}(\lambda)$ by the length of a 1KeySKFE ciphertext.
3. Sample $\tau \leftarrow \{0, 1\}^{4\lambda}$
4. Generate $\text{pkfe.sk}_{g[f, C_{\text{ske}}, \tau]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f, C_{\text{ske}}, \tau])$ where $g[f, C_{\text{ske}}, \tau]$ is a function described in Figure A.1.
5. Output $\text{sk}_f = \text{pkfe.sk}_{g[f, C_{\text{ske}}, \tau]}$.

Enc(MPK, m):

1. Parse $\text{MPK} = \text{pkfe.MPK}$.
2. Sample $K \leftarrow \mathcal{K}$.
3. Generate $1\text{keyskfe.MSK} \leftarrow 1\text{KeySKFE.Setup}(1^\lambda)$.
4. Generate $1\text{keyskfe.sk}_{h[m]} \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}, h[m])$ where $h[m]$ is a function described in Figure A.2.
5. Compute $\text{pkfe.ct} \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (1\text{keyskfe.MSK}, K, \perp, 0))$.
6. Output $\text{ct} = (1\text{keyskfe.sk}_{h[m]}, \text{pkfe.ct})$.

Dec(sk_f, ct, y):

1. Parse $\text{sk}_f = \text{pkfe.sk}_{g[f]}$ and $\text{ct} = (1\text{keyskfe.sk}_{h[m]}, \text{pkfe.ct})$.
2. Compute $(\text{skfe.sk}_f, 1\text{keyskfe.ct}) \leftarrow \text{PKFE.Dec}(\text{pkfe.sk}_{g[f]}, \text{pkfe.ct})$.

$$\underline{h[m]}$$

Input: skfe.MSK, R, α

1. If $\alpha = 0$ then
 - Compute $\text{skfe.ct} \leftarrow \text{SKFE.Enc}(\text{skfe.MSK}, m; R)$.
 - Output skfe.ct .
2. Else, output $\perp$.

Figure A.2: The description of the function $h[m]$

3. Compute $\text{skfe.ct} \leftarrow \text{1KeySKFE.Dec}(\text{1keyskfe.sk}_{h[m]}, \text{1keyskfe.ct})$.
4. Compute $m' \leftarrow \text{SKFE.Dec}(\text{skfe.sk}_f, \text{skfe.ct}, y)$
5. Output 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 A.2.21. *If PKFE is selectively secure PKFE for $\mathbf{P}/\text{poly}$, 1KeySKFE is selectively single-key function private SKFE for $\mathbf{P}/\text{poly}$, SKFE is adaptively single-key single-ciphertext public-slot SKFE for $\mathbf{P}/\text{poly}$, PRF is a secure PRF, and SKE is ciphertext pseudorandom, FE is adaptively indistinguishable-secure public-slot PKFE for $\mathbf{P}/\text{poly}$.*

We immediately obtain Theorem 4.3.12 from the thereom above.

Correctness. Let $\text{ct} = (\text{1keyskfe.sk}_{h[m]}, \text{pkfe.ct})$ be an honestly generated ciphertext encrypting a message m and $\text{sk}_f = \text{pkfe.sk}_{g[f]}$ be an honestly generated secret key corresponding to a function f . Firstly, we note that pkfe.ct is an encryption of the message $(\text{1keyskfe.MSK}, K, \perp, 0)$ and $g[f]$ is a function that takes $(\text{1keyskfe.MSK}, K, \perp, 0)$ as input and outputs a SKFE secret key skfe.sk_f corresponding to the function f and a single key SKFE ciphertext 1keyskfe.ct . Therefore, by the correctness of PKFE, the decryption algorithm $\text{PKFE.Dec}(\text{pkfe.sk}_{g[f]}, \text{pkfe.ct})$ yields $g[f](\text{1keyskfe.MSK}, K, \perp, 0) = (\text{skfe.sk}_f, \text{1keyskfe.ct})$. Secondly, we observe that 1keyskfe.ct encrypts a message $(\text{SKFE.MSK}, R_2, 0)$ and $h[m]$ is a function that takes $(\text{SKFE.MSK}, R_2, 0)$ an input and outputs a SKFE ciphertext skfe.ct . Therefore, by the correctness of SKFE, the decryption algorithm $\text{1KeySKFE.Dec}(\text{1keyskfe.sk}_{h[m]}, \text{1keyskfe.ct})$ yields $h[m](\text{SKFE.MSK}, R_2, 0) = \text{skfe.ct}$. Finally, we note that 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 $\text{SKFE.Dec}(\text{skfe.sk}_f, \text{skfe.ct}, y) = m' = f(m, y)$.

Adaptive Security. We prove Theorem A.2.21.

Proof of Theorem A.2.21. 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 $\Pr[\text{Hyb}_i = 1]$ be the winning probability of $\mathcal{A}$ in Hyb_i for all i .

Hyb₀: This is the original adaptive security experiment where the challenge bit set to 0. Specifically, it works as follows:

1. The challenger generates $(\text{pkfe.MPK}, \text{pkfe.MSK}) \leftarrow \text{PKFE.Setup}(1^\lambda)$, sets $\text{MPK} := \text{pkfe.MPK}$ and $\text{MSK} := \text{pkfe.MSK}$, and sends MPK to $\mathcal{A}$.
2. The challenger samples a PRF key $K^* \leftarrow \mathcal{K}$ and generate a master secret key $1\text{keyskfe.MSK}^* \leftarrow 1\text{KeySKFE.Setup}(1^\lambda)$.
3. The challenger computes $\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (1\text{keyskfe.MSK}^*, K^*, \perp, 0))$.
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 $C_{j,\text{ske}}, \tau_j = (\tau_{j,0} \parallel \tau_{j,1} \parallel \tau_{j,2} \parallel \tau_{j,3})$ uniformly at random.
 - (b) Generate $\text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f_j, C_{j,\text{ske}}, \tau_j])$ where $g[f_j, C_{j,\text{ske}}, \tau_j]$ is a function described in Figure A.1.
 - (c) Set $\text{sk}_{f_j} := \text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$.
 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 $1\text{keyskfe.sk}_{h[m_0]}^* \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}^*, h[m_0])$ where $h[m_0]$ is a function described in Figure A.2.
 - (b) Set $\text{ct}^* := (1\text{keyskfe.sk}_{h[m_0]}^*, \text{pkfe.ct}^*)$ where 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 1keyskfe.MSK^* before it answers any secret key query. Moreover, the challenger can also compute the part of the challenge ciphertext pkfe.ct^* before the key query phase.

Hyb₁: This hybrid is identical to Hyb₀ except the challenger samples a SKE key ske.SK^* before it answers any key query and sets $C_{j,\text{ske}}$ 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 $\tau_j = (\tau_{j,0} \parallel \tau_{j,1} \parallel \tau_{j,2} \parallel \tau_{j,3})$ uniformly at random.
- (b) Compute $R_{j,i} = \text{PRF}(K^*, \tau_{j,i})$ for all $i \in \{0, 1, 2, 3\}$.
- (c) Generate $\text{skfe.MSK}_j \leftarrow \text{SKFE.Setup}(1^\lambda; R_{j,0})$.
- (d) Compute $\text{skfe.sk}_{f_j} \leftarrow \text{SKFE.KeyGen}(\text{skfe.MSK}_j, f_j; R_{j,1})$.
- (e) Compute $1\text{keyskfe.ct}_j \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_j, R_{j,2}, 0); R_{j,3})$.
- (f) Compute $C_{j,\text{ske}} \leftarrow \text{SKE.Enc}(\text{ske.SK}^*, u_j)$ where $u_j = (\text{skfe.sk}_{f_j}, 1\text{keyskfe.ct}_j)$.
- (g) Generate $\text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f_j, C_{j,\text{ske}}, \tau_j])$ where $g[f_j, C_{j,\text{ske}}, \tau_j]$ is a function described in Figure A.1.
- (h) Set $\text{sk}_f := \text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$.

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 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 $u_j = (\text{skfe.sk}_{f_j}, 1\text{keyskfe.ct}_j)$ to its challenger. Upon receiving a ciphertext $C_{j,\text{ske}}$ from the challenger, $\mathcal{B}_1$ computes $\text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$ 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 winning probability of $\mathcal{B}_1$ is the same as $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]| \leq \text{negl}$. Hence, by the security of SKE, it holds that

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

Hyb₂: This hybrid is identical to Hyb_1 except the challenger computes

$$\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (\perp, \perp, \text{ske.SK}^*, 1))$$

instead of computing $\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (1\text{keyskfe.MSK}^*, K^*, \perp, 0))$. In particular, the mode of decryption is changed to $\beta = 1$ from $\beta = 0$ meaning that ske.SK^* is used to decrypt $C_{j,\text{ske}}$ to get an output of $(\text{skfe.sk}_{f_j}, 1\text{keyskfe.ct}_j)$ while decryption of ct^* is performed by the j -th secret key $\text{sk}_f := \text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$. 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 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 $((1\text{keyskfe.MSK}^*, K^*, \perp, 0), (\perp, \perp, \text{ske.SK}^*, 1))$ to its challenger and receives the public key pkfe.MPK and a ciphertext 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 its challenger for a function $g[f_j, C_{j,\text{ske}}, \tau_j]$ and returns the output to $\mathcal{A}$. Firstly, $\mathcal{B}_2$ is an admissible adversary as

$$g[f_j, C_{j,\text{ske}}, \tau_j](1\text{keyskfe.MSK}^*, K^*, \perp, 0) = g[f_j, C_{j,\text{ske}}, \tau_j](\perp, \perp, \text{ske.SK}^*, 1)$$

holds for all j . If $\mathcal{B}_2$ receives an encryption of $(1\text{keyskfe.MSK}^*, K^*, \perp, 0)$ then it simulates Hyb_1 , otherwise, if $\mathcal{B}_2$ receives an encryption of $(\perp, \perp, \text{ske.SK}^*, 1)$ then it simulates Hyb_2 . Therefore, the winning probability of $\mathcal{B}_2$ is essentially the same as $|\Pr[\text{Hyb}_1 = 1] - \Pr[\text{Hyb}_2 = 1]| \leq \text{negl}$. Hence, by the security of PKFE, it holds that

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

Hyb₃: 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 $|\Pr[\text{Hyb}_2 = 1] - \Pr[\text{Hyb}_3 = 1]| \leq \text{negl}$. Hence, by the security of PRF, it holds that

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

$$\underline{h[m, m', v]}$$

Input: skfe.MSK, R, α

1. If $\alpha = 0$ then
 - Compute $\text{skfe.ct} \leftarrow \text{SKFE.Enc}(\text{skfe.MSK}, m; R)$.
 - Output skfe.ct .
2. If $\alpha = 1$ then
 - Compute $\text{skfe.ct} \leftarrow \text{SKFE.Enc}(\text{skfe.MSK}, m'; R)$.
 - Output skfe.ct .
3. Else, output v .

Figure A.3: The description of the function $h[m, m', v]$

Hyb₄: This hybrid is identical to Hyb₃ except the challenger generates

$$1\text{keyskfe.sk}_{h[m_0, m_1, v]}^* \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}^*, h[m_0, m_1, v])$$

and sets the challenge ciphertext as $\text{ct}^* := (1\text{keyskfe.sk}_{h[m_0, m_1, v]}^*, \text{pkfe.ct}^*)$ where $h[m_0, m_1, v]$ is a function described in Figure A.3 and v is a random string. The indistinguishability between Hyb₃ and Hyb₄ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of 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 $(M_1, \dots, M_Q)$ where $M_j = (\text{skfe.MSK}_j, R_{j,2}, 0)$ for $j \in [Q]$. More precisely, $\mathcal{B}$ sends (M_j, M_j) for all $j \in [Q]$ and receives the ciphertexts as $\{1\text{keyskfe.ct}_j\}_{j \in [Q]}$ which is used in answering $\mathcal{A}$'s secret key queries as in Hyb₃ or Hyb₄. 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 its challenger. Then, $\mathcal{B}_4$ uses the output from its 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

$$h[m_0](\text{skfe.MSK}_j, R_{j,2}, 0) = h[m_0, m_1, v](\text{skfe.MSK}_j, R_{j,2}, 0)$$

holds for all $j \in [Q]$. If $\mathcal{B}_4$ receives a secret key $1\text{keyskfe.sk}_{h[m_0]}^*$ then it simulates Hyb₃, otherwise, if $\mathcal{B}_4$ receives a secret key $1\text{keyskfe.sk}_{h[m_0, m_1, v]}^*$ then it simulates Hyb₄. Therefore, the winning probability of $\mathcal{B}_4$ is essentially the same as $|\Pr[\text{Hyb}_3 = 1] - \Pr[\text{Hyb}_4 = 1]| \leq \text{negl}$. Hence, by the security of 1KeySKFE, it holds that

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

Hyb₅: This hybrid is identical to Hyb₄ except the challenger computes

$$1\text{keyskfe.ct}_j \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_j, R_{j,2}, 1); R_{j,3})$$

while generating the j -th secret key corresponding to a function f_j for all j . In Lemma A.2.22, we show that

$$|\Pr[\text{Hyb}_4 = 1] - \Pr[\text{Hyb}_5 = 1]| \leq \text{negl}(\lambda).$$

Hyb₆: This hybrid is identical to Hyb₅ except the challenger generates

$$1\text{keyskfe.sk}_{h[m_1, m_1, v]}^* \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}^*, h[m_1, m_1, v])$$

and sets the challenge ciphertext as $\text{ct}^* := (1\text{keyskfe.sk}_{h[m_1, m_1, v]}^*, \text{pkfe.ct}^*)$ where $h[m_1, m_1, v]$ is a function as described in Figure A.3. The indistinguishability between Hyb₅ and Hyb₆ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of 1keyskfe.MSK^* and using the challenger of the security experiment of 1KeySKFE. The simulation strategy is similar to Hyb₄. By the security of 1KeySKFE, it holds that

$$|\Pr[\text{Hyb}_5 = 1] - \Pr[\text{Hyb}_6 = 1]| \leq \text{negl}(\lambda).$$

Hyb₇: This hybrid is identical to Hyb₆ except the challenger computes

$$1\text{keyskfe.ct}_j \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_j, R_{j,2}, 0); R_{j,3})$$

while generating the j -th secret key corresponding to a function f_j for all j . Moreover, the challenger generates

$$1\text{keyskfe.sk}_{h[m_1]}^* \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}^*, h[m_1])$$

and sets the challenge ciphertext as $\text{ct}^* := (1\text{keyskfe.sk}_{h[m_1]}^*, \text{pkfe.ct}^*)$ where $h[m_1]$ is a function as described in Figure A.2. The indistinguishability between Hyb₆ and Hyb₇ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of 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 $((M_1^{(0)}, M_1^{(1)}), \dots, (M_Q^{(0)}, M_Q^{(1)}))$ where

$$M_j^{(0)} = (\text{skfe.MSK}_j, R_{j,2}, 1) \text{ and } M_j^{(1)} = (\text{skfe.MSK}_j, R_{j,2}, 0)$$

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 $\{1\text{keyskfe.ct}_j\}_{j \in [Q]}$ 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 its challenger. Then, $\mathcal{B}_7$ uses the output from its 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

$$h[m_1, m_1, v](\text{skfe.MSK}_j, R_{j,2}, 1) = h[m_1](\text{skfe.MSK}_j, R_{j,2}, 0)$$

holds for all $j \in [Q]$. If $\mathcal{B}_7$ receives ciphertexts for the messages $M_j^{(0)}$ and a secret key $1\text{keyskfe.sk}_{h[m_1, m_1, v]}^*$ then it simulates Hyb₆, otherwise, if $\mathcal{B}_7$ receives ciphertexts for the messages $M_j^{(1)}$ and a secret key $1\text{keyskfe.sk}_{h[m_1]}^*$ then it simulates Hyb₇. Therefore, the winning probability of $\mathcal{B}_7$ is essentially the same as $|\Pr[\text{Hyb}_6 = 1] - \Pr[\text{Hyb}_7 = 1]| \leq \text{negl}$. Hence, by the security of 1KeySKFE, it holds that

$$|\Pr[\text{Hyb}_6 = 1] - \Pr[\text{Hyb}_7 = 1]| \leq \text{negl}(\lambda).$$

Hyb₈: This hybrid is identical to Hyb₇ except the challenger samples

$$R_{j,i} = \text{PRF}(K^*, \tau_{j,i}) \text{ for all } i \in \{0, 1, 2, 3\}$$

instead of sampling these values uniformly at random for all j, i , while answering the secret key queries. The indistinguishability between Hyb₇ and Hyb₈ 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

$$|\Pr[\text{Hyb}_7 = 1] - \Pr[\text{Hyb}_8 = 1]| \leq \text{negl}(\lambda).$$

Hyb₉: This hybrid is identical to Hyb₈ except the challenger computes

$$\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (1\text{keyskfe.MSK}^*, K^*, \perp, 0))$$

instead of computing $\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (\perp, \perp, \text{ske.SK}^*, 1))$. The indistinguishability between Hyb₈ and Hyb₉ follows from the security of PKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of pkfe.MSK and using the challenger of the security experiment of PKFE. In other words, it holds that

$$|\Pr[\text{Hyb}_8 = 1] - \Pr[\text{Hyb}_9 = 1]| \leq \text{negl}(\lambda).$$

Hyb₁₀: This hybrid is identical to Hyb₉ except the challenger does not sample ske.SK^* and chooses $C_{j,\text{ske}}$ 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 $C_{j,\text{ske}}, \tau_j = (\tau_{j,0} \| \tau_{j,1} \| \tau_{j,2} \| \tau_{j,3})$ uniformly at random.
- (b) Generate $\text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f_j, C_{j,\text{ske}}, \tau_j])$ where $g[f_j, C_{j,\text{ske}}, \tau_j]$ is a function described in Figure A.1.
- (c) Set $\text{sk}_{f_j} := \text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$.

The challenger sends sk_{f_j} to $\mathcal{A}$. The indistinguishability between Hyb₉ and Hyb₁₀ follows from the security of SKE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of ske.SK and using the challenger of the security experiment of SKE. In other words, it holds that

$$|\Pr[\text{Hyb}_9 = 1] - \Pr[\text{Hyb}_{10} = 1]| \leq \text{negl}(\lambda).$$

We observe that Hyb₁₀ 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 $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_{10} = 1]| \leq \text{negl}(\lambda)$.

This completes the proof of Theorem A.2.21 if we prove Lemma A.2.22. □

Lemma A.2.22. *If 1KeySKFE is selectively single key function-private secure and public-slot SKFE SKFE is adaptively single-key single-ciphertext secure then for any $\lambda \in [n]$,*

$$|\Pr[\text{Hyb}_4 = 1] - \Pr[\text{Hyb}_5 = 1]| \leq \text{negl}(\lambda).$$

Proof of Lemma A.2.22. We prove this lemma using a sequence of hybrids $\text{Hyb}_{4,q,1}, \text{Hyb}_{4,q,2}, \text{Hyb}_{4,q,3}, \text{Hyb}_{4,q,4}$ for $q \in [Q]$ where Q denotes the total number of secret key queried made by the adversary $\mathcal{A}$. Let us denote $\text{Hyb}_{4,Q+1,1} = \text{Hyb}_5$.

$\text{Hyb}_{4,q,1}$: This is exactly the same as Hyb_4 except the challenger sets v to be the output of $\text{SKFE.Enc}(\text{skfe.MSK}_q, m_0; R_{q,2})$. More precisely, the hybrid works as follows:

1. The challenger generates $(\text{pkfe.MPK}, \text{pkfe.MSK}) \leftarrow \text{PKFE.Setup}(1^\lambda)$, sets $\text{MPK} := \text{pkfe.MPK}$ and $\text{MSK} := \text{pkfe.MSK}$, and sends MPK to $\mathcal{A}$.
2. The challenger generates a master secret key $1\text{keyskfe.MSK}^* \leftarrow 1\text{KeySKFE.Setup}(1^\lambda)$.
3. The challenger generates a secret key $\text{ske.SK}^* \leftarrow \text{SKE.Setup}(1^\lambda)$.
4. The challenger computes $\text{pkfe.ct}^* \leftarrow \text{PKFE.Enc}(\text{pkfe.MPK}, (\perp, \perp, \text{ske.SK}^*, 1))$.
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 $\text{skfe.MSK}_q \leftarrow \text{SKFE.Setup}(1^\lambda; R_{q,0})$.
- (c) Compute $\text{skfe.sk}_{f_q} \leftarrow \text{SKFE.KeyGen}(\text{skfe.MSK}_q, f_q; R_{q,1})$.
- (d) Compute $1\text{keyskfe.ct}_q \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_q, R_{q,2}, 0); R_{q,3})$.
- (e) Set $u_q = (\text{skfe.sk}_{f_q}, 1\text{keyskfe.ct}_q)$.

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 $\text{skfe.MSK}_j \leftarrow \text{SKFE.Setup}(1^\lambda; R_{j,0})$.
- (c) Compute $\text{skfe.sk}_{f_j} \leftarrow \text{SKFE.KeyGen}(\text{skfe.MSK}_j, f_j; R_{j,1})$.
- (d) Compute

$$\begin{aligned} 1\text{keyskfe.ct}_j &\leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_j, R_{j,2}, 1); R_{j,3}) & \text{if } j < q \\ 1\text{keyskfe.ct}_j &\leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.MSK}_j, R_{j,2}, 0); R_{j,3}) & \text{if } j > q \end{aligned}$$

- (e) Compute $C_{j,\text{ske}} \leftarrow \text{SKE.Enc}(\text{ske.SK}^*, u_j)$ where $u_j = (\text{skfe.sk}_{f_j}, 1\text{keyskfe.ct}_j)$.
- (f) Generate $\text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f_j, C_{j,\text{ske}}, \tau_j])$ where $g[f_j, C_{j,\text{ske}}, \tau_j]$ is a function described in Figure A.1.
- (g) Set $\text{sk}_f := \text{pkfe.sk}_{g[f_j, C_{j,\text{ske}}, \tau_j]}$.

If $j = q$:

- (a) Set $C_{q,\text{ske}} \leftarrow \text{SKE.Enc}(\text{ske.SK}^*, u_q)$.
- (b) Generate $\text{pkfe.sk}_{g[f_q, C_{q,\text{ske}}, \tau_q]} \leftarrow \text{PKFE.KeyGen}(\text{pkfe.MSK}, g[f_q, C_{q,\text{ske}}, \tau_q])$ where $g[f_q, C_{q,\text{ske}}, \tau_q]$ is a function described in Figure A.1.

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 $v := \text{SKFE.Enc}(\text{skfe.MSK}_q, m_0; R_{q,2})$

- (b) Generate $1\text{keyskfe.sk}_{h[m_0, m_1, v]}^* \leftarrow 1\text{KeySKFE.KeyGen}(1\text{keyskfe.MSK}^*, h[m_0, m_1, v])$ where $h[m_0, m_1, v]$ is a function described in Figure A.3.
- (c) Set $\text{ct}^* := (1\text{keyskfe.sk}_{h[m_0, m_1, v]}^*, \text{pkfe.ct}^*)$ where 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 $\text{Hyb}_{4,1,1}$ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of 1keyskfe.MSK^* and using the challenger of the security experiment of 1KeySKFE . Let us consider an adversary $\mathcal{B}_{4,1}$ against the security of 1KeySKFE .

At first, $\mathcal{B}_{4,1}$ prepares a list of Q challenge message pairs $(M_1, \dots, M_Q)$ where

$$\begin{aligned} M_j &= (\text{skfe.MSK}_j, R_{j,2}, 1) & \text{if } 1 \leq j < q \\ M_j &= (\text{skfe.MSK}_j, R_{j,2}, 0) & \text{if } q \leq j \leq Q. \end{aligned}$$

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 $\{1\text{keyskfe.ct}_j\}_{j \in [Q]}$ 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 its challenger where v is a random string of appropriate length and $v' = \text{SKFE.Enc}(\text{skfe.MSK}_q, m_0; R_{q,2})$. Then, $\mathcal{B}_{4,1}$ uses the output from its 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 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 $1\text{keyskfe.sk}_{h[m_0, m_1, v]}^*$ then it simulates Hyb_4 , otherwise, if $\mathcal{B}_{4,1}$ receives a secret key $1\text{keyskfe.sk}_{h[m_0, m_1, v']}^*$ then it simulates $\text{Hyb}_{4,1,1}$. Therefore, the winning probability of $\mathcal{B}_{4,1}$ is essentially the same as $|\Pr[\text{Hyb}_4 = 1] - \Pr[\text{Hyb}_{4,1,1} = 1]| \leq \text{negl}$. Hence, by the security of 1KeySKFE , it holds that

$$|\Pr[\text{Hyb}_4 = 1] - \Pr[\text{Hyb}_{4,1,1} = 1]| \leq \text{negl}(\lambda).$$

$\text{Hyb}_{4,q,2}$: This is exactly the same as $\text{Hyb}_{4,q,1}$ 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 1keyskfe.ct_q as follows:

$$1\text{keyskfe.ct}_q \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (0, 0, 2); R_{q,3}).$$

The indistinguishability between $\text{Hyb}_{4,q,1}$ and $\text{Hyb}_{4,q,2}$ follows from the security of 1KeySKFE since the view of the adversary $\mathcal{A}$ can be simulated without the knowledge of 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

$$h[m_0, m_1, v](\text{skfe.MSK}_q, R_{q,2}, 0) = h[m_0, m_1, v](0, 0, 2)$$

holds where $v = \text{SKFE.Enc}(\text{skfe.MSK}_q, m_0; R_{q,2})$. Hence, by the security of 1KeySKFE , it holds that

$$|\Pr[\text{Hyb}_{4,q,1} = 1] - \Pr[\text{Hyb}_{4,q,2} = 1]| \leq \text{negl}(\lambda).$$

Hyb_{4,q,3} : This is exactly the same as Hyb_{4,q,2} except the challenger changes v to be the encryption of m_1 , that is, it sets v as follows:

$$v := \text{SKFE.Enc}(\text{skfe.MSK}_q, m_1; R_{q,2})$$

The indistinguishability between Hyb_{4,q,2} and 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 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 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 skfe.sk_{f_q} at any point whenever $\mathcal{A}$ asks for a secret key for f_q and sets $u_q = (\text{skfe.sk}_{f_q}, 1\text{keyskfe.ct}_q)$. 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 an arbitrary input y to the public slot of f . If $\mathcal{B}_{4,3}$ receives a ciphertext $v = \text{SKFE.Enc}(\text{skfe.MSK}_q, m_0)$ then it simulates Hyb_{4,q,2}, otherwise, if $\mathcal{B}_{4,3}$ receives a ciphertext $v = \text{SKFE.Enc}(\text{skfe.MSK}_q, m_1)$ then it simulates Hyb_{4,q,3}. Therefore, the winning probability of $\mathcal{B}_{4,3}$ is essentially the same as $\left| \Pr[\text{Hyb}_{4,q,2} = 1] - \Pr[\text{Hyb}_{4,q,3} = 1] \right| \leq \text{negl}$. Hence, by the security of SKFE, it holds that

$$\left| \Pr[\text{Hyb}_{4,q,2} = 1] - \Pr[\text{Hyb}_{4,q,3} = 1] \right| \leq \text{negl}(\lambda).$$

Hyb_{4,q,4} : This is exactly the same as 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 1keyskfe.ct_q as follows:

$$1\text{keyskfe.ct}_q \leftarrow 1\text{KeySKFE.Enc}(1\text{keyskfe.MSK}^*, (\text{skfe.ct}_q, R_{q,2}; 1); R_{q,3}).$$

The indistinguishability between Hyb_{4,q,3} and 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 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

$$h[m_0, m_1, v](0, 0, 2) = h[m_0, m_1, v](\text{skfe.MSK}_q, R_{q,2}, 1)$$

holds where $v = \text{SKFE.Enc}(\text{skfe.MSK}_q, m_1; R_{q,2})$. Hence, by the security of 1KeySKFE, it holds that

$$\left| \Pr[\text{Hyb}_{4,q,3} = 1] - \Pr[\text{Hyb}_{4,q,4} = 1] \right| \leq \text{negl}(\lambda).$$

Combining the advantage of $\mathcal{A}$ in all the consecutive hybrids and applying the triangular inequality, we have $|\Pr[\text{Hyb}_4 = 1] - \Pr[\text{Hyb}_5 = 1]| \leq \text{negl}(\lambda)$. This completes the proof of Lemma A.2.22. $\square$

A.3 Secret and Public Key Encryption with Certified Everlasting Deletion

In Appendix A.3.1, we define SKE and PKE with certified everlasting deletion. In Appendix A.3.2 and Appendix A.3.3, we construct a certified everlasting secure SKE scheme with and without QROM, respectively. In Appendix A.3.4 and Appendix A.3.5, we construct a certified everlasting secure PKE scheme with and without QROM, respectively.

A.3.1 Definition

Definition A.3.1 (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 $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ 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 $\mathcal{D} := \mathcal{Q}^{\otimes s(\lambda)}$.

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

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

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

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

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

Remark A.3.2. Although we consider quantum certificates in Appendix A.3.3, we consider classical certificates by default. In the quantum certificate case, we need to use *cert* and *Vrfy* in the syntax.

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

Definition A.3.3 (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 $m \in \mathcal{M}$,

$$\Pr \left[m' \neq m \mid \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m) \\ m' \leftarrow \text{Dec}(\text{sk}, \text{ct}) \end{array} \right] \leq \text{negl}(\lambda).$$

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

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

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 A.5.2, and verification correctness with QOTP for the construction of FE in Section 4.4.3.

Definition A.3.4 (Special Correctness). There exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,

$$\Pr \left[\text{Dec}(\text{sk}_2, \text{ct}) \neq \perp \mid \begin{array}{l} \text{sk}_2, \text{sk}_1 \leftarrow \text{KeyGen}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}_1, m) \end{array} \right] \leq \text{negl}(\lambda).$$

Definition A.3.5 (Verification Correctness with QOTP). *There exists a negligible function negl and a PPT algorithm Recover such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}^*) = \perp \mid \begin{array}{l} \text{sk} \leftarrow \text{KeyGen}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m) \\ a, b \leftarrow \{0, 1\}^{p(\lambda)} \\ \widetilde{\text{cert}} \leftarrow \text{Del}(Z^b X^a \text{ct} X^a Z^b) \\ \text{cert}^* \leftarrow \text{Recover}(a, b, \widetilde{\text{cert}}) \end{array} \right] \leq \text{negl}(\lambda).$$

As security, we consider two definitions, Definition A.3.6 and Definition A.3.7 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 A.3.6 (IND-CPA Security for SKE with Certified Everlasting Deletion). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be an SKE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *$\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m)$, 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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m_b)$, and sends ct to $\mathcal{A}$.*
5. *$\mathcal{A}$ sends an encryption query m to the challenger. The challenger computes $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m)$, 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

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

Definition A.3.7 (Certified Everlasting IND-CPA Security for SKE). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a certified everlasting SKE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$.*

1. *The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.*
2. *$\mathcal{A}_1$ sends an encryption query m_i to the challenger. The challenger computes $(\text{vk}_i, \text{ct}_i) \leftarrow \text{Enc}(\text{sk}, m_i)$, and returns $(\text{vk}_i, \text{ct}_i)$ 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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{sk}, m_b)$, and sends ct to $\mathcal{A}_1$.*

5. $\mathcal{A}_1$ sends an encryption query m_i to the challenger. The challenger computes $(vk_i, ct_i) \leftarrow \text{Enc}(\text{sk}, m_i)$, 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 $\text{Vrfy}(vk, \text{cert})$. If the output is $\perp$, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\top$, and sends sk to $\mathcal{A}_2$.
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 $\perp$.

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

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

Definition A.3.8 (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 $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ 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 $\mathcal{D} := \mathcal{Q}^{\otimes t(\lambda)}$.

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

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

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

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

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

Remark A.3.9. Although we consider quantum certificates in Appendix A.3.5, we consider classical certificates by default. In the quantum certificate case, we need to use cert and Vrfy in the syntax.

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

Definition A.3.10 (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 $m \in \mathcal{M}$,

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

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

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}) = \perp \mid \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m) \\ \text{cert} \leftarrow \text{Del}(\text{ct}) \end{array} \right] \leq \text{negl}(\lambda).$$

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 4.4.3.

Definition A.3.11 (Verification Correctness with QOTP). *There exists a negligible function negl and a PPT algorithm Recover such that for any $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$,*

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}^*) = \perp \mid \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m) \\ a, b \leftarrow \{0, 1\}^{p(\lambda)} \\ \widetilde{\text{cert}} \leftarrow \text{Del}(Z^b X^a \text{ct} X^a Z^b) \\ \text{cert}^* \leftarrow \text{Recover}(a, b, \widetilde{\text{cert}}) \end{array} \right] \leq \text{negl}(\lambda).$$

As security, we consider two definitions, Definition A.3.12 and Definition A.3.13 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 A.3.12 (IND-CPA Security for PKE with Certified Everlasting Deletion). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a PKE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.*

1. *The challenger generates $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$, 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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m_b)$, 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

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

Definition A.3.13 (Certified Everlasting IND-CPA Security for PKE). *Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a PKE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$.*

1. *The challenger computes $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$, 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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m_b)$, 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 $\text{Vrfy}(\text{vk}, \text{cert})$. If the output is $\perp$, the challenger outputs $\perp$, and sends $\perp$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\top$, and sends sk to $\mathcal{A}_2$.
6. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
7. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

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

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

A.3.2 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 $\Sigma_{\text{cesk}} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ from the following primitives.

- A one-time SKE with certified deletion scheme (Definition 2.2.32) $\Sigma_{\text{skcd}} = \text{CD}(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$.
- A SKE scheme (Definition 2.2.7) $\Sigma_{\text{sk}} = \text{SKE}(\text{KeyGen}, \text{Enc}, \text{Dec})$ with plaintext space $\{0, 1\}^\lambda$.
- A hash function H modeled as a quantum random oracle.

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

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

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

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

$\text{Dec}(\text{sk}, ct)$:

- Parse $\text{sk} = \text{ske.sk}$ and $ct = (h, \text{ske.ct}, \text{cd.ct})$.
- Compute $R' \text{ or } \perp \leftarrow \text{SKE.Dec}(\text{ske.sk}, \text{ske.ct})$. If it outputs $\perp$, $\text{Dec}(\text{sk}, ct)$ outputs $\perp$.
- Compute $\text{cd.sk}' := H(R') \oplus h$.

- Compute $m' \leftarrow \text{CD}.\text{Dec}(\text{cd.sk}', \text{cd.ct})$.
- Output m' .

$\text{Del}(ct)$:

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

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

- Parse $\text{vk} = \text{cd.sk}$ and $\text{cert} = \text{cd.cert}$.
- Compute $b \leftarrow \text{CD}.\text{Vrfy}(\text{cd.sk}, \text{cd.cert})$.
- 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} . Verification correctness with QOTP holds due to that of Σ_{skcd} .

Security: The following two theorems hold.

Theorem A.3.14. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.2.10) and Σ_{skcd} satisfies the OT-CD security (Definition 2.2.32), Σ_{cesk} satisfies the IND-CPA security (Definition A.3.6).*

Its proof is similar to that of Theorem A.3.15, and therefore we omit it.

Theorem A.3.15. *If Σ_{sk} satisfies the OW-CPA security (Definition 2.2.10) and Σ_{skcd} satisfies the OT-CD security (Definition 2.2.32), Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition A.3.7).*

Its proof is similar to that of [HMNY22b, Theorem 5.8].

A.3.3 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 A.3.2), in this construction the plaintext space is of constant size. However, the size can be extended to the polynomial size via the standard hybrid argument. Our construction is similar to that of revocable quantum timed-release encryption in [Unr15]. The difference is that we use SKE instead of timed-release encryption.

Our certified everlasting 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 $\Sigma_{\text{cesk}} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ with plaintext space $\mathcal{M} = C_1/C_2$ (isomorphic to $\{0, 1\}^{k_1-k_2}$), ciphertext space $\mathcal{C} = \mathbb{Q}^{\otimes(p(\lambda)+q(\lambda))} \times \{0, 1\}^{r(\lambda)} \times \{0, 1\}^{q(\lambda)}/C_1 \times C_1/C_2$, 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 $\mathcal{D} = \mathbb{Q}^{\otimes(p(\lambda)+q(\lambda))}$ from the following primitive.

- An SKE scheme (Definition 2.2.7) $\Sigma_{\text{sk}} = \text{SKE}(\text{KeyGen}, \text{Enc}, \text{Dec})$ 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.)

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

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

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

- Parse $\text{sk} = \text{ske.sk}$.
- 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 $\text{ske.ct} \leftarrow \text{SKE.Enc}(\text{ske.sk}, (B, Q, r, y))$.
- Let U_Q be the unitary that permutes the qubits in Q into the first half of the system. (I.e., $U_Q |x_1 x_2 \cdots x_{p+q}\rangle = |x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\rangle$ with $Q := \{a_1, a_2, \dots, a_p\}$ and $\{1, 2, \dots, p+q\} \setminus Q := \{b_1, b_2, \dots, b_q\}$.)
- 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 $\text{ct} := (|\Psi\rangle, \text{ske.ct}, u, h)$ and $\text{vk} := (B, Q, r)$.

$\text{Dec}(\text{sk}, \text{ct})$:

- Parse $\text{sk} = \text{ske.sk}$, $\text{ct} = (|\Psi\rangle, \text{ske.ct}, u, h)$.
- Compute $(B, Q, r, y) / \perp \leftarrow \text{SKE.Dec}(\text{ske.sk}, \text{ske.ct})$. If $\perp \leftarrow \text{SKE.Dec}(\text{ske.sk}, \text{ske.ct})$, $\text{Dec}(\text{sk}, \text{ct})$ outputs $\perp$ 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 $m' := h \oplus x \oplus y$.

$\text{Del}(\text{ct})$:

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

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

- Parse $\text{vk} = (B, Q, r)$ and $\text{cert} = |\Psi\rangle$.
- 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 $\perp$ otherwise.

Correctness. Correctness easily follows from that of Σ_{sk} . Special correctness holds due to that of Σ_{sk} . Verification correctness with QOTP holds since Recover is the decryption algorithm of QOTP.

Security. The following two theorems hold.

Theorem A.3.16. *If Σ_{sk} is IND-CPA secure (Definition 2.2.11), then Σ_{cesk} is IND-CPA secure (Definition A.3.6).*

Its proof is straightforward, so we omit it.

Theorem A.3.17. *If Σ_{sk} is IND-CPA secure (Definition 2.2.11) and $tp/(p+q) - 4(k_1 - k_2)\ln 2$ is superlogarithmic, then Σ_{cesk} is certified everlasting IND-CPA secure (Definition A.3.7).*

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.

A.3.4 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 $\Sigma_{cepk} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ from a one-time SKE with certified deletion scheme $\Sigma_{skcd} = \text{SKE}(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ (Definition 2.2.32), a PKE scheme $\Sigma_{pk} = \text{PKE}(\text{KeyGen}, \text{Enc}, \text{Dec})$ with plaintext space $\{0, 1\}^\lambda$ (Definition 2.2.1) and a hash function H modeled as quantum random oracle.

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

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

$\text{Enc}(\text{pk}, m)$:

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

$\text{Dec}(\text{sk}, ct)$:

- Parse $\text{sk} = \text{pke.sk}$ and $ct = (h, \text{ske.ct}, \text{pke.ct})$.
- Compute $R' \leftarrow \text{PKE.Dec}(\text{pke.sk}, \text{pke.ct})$.
- Compute $\text{ske.sk}' := h \oplus H(R')$.

- Compute $m' \leftarrow \text{SKE.Dec}(\text{ske.sk}', \text{ske.ct})$.
- Output m' .

$\text{Del}(ct)$:

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

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

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

Correctness: Correctness easily follows from those of Σ_{pk} and Σ_{skcd} . Verification correctness with QOTP holds due to that of Σ_{skcd} .

Security: The following two theorems hold. Their proofs are similar to those of Theorems A.3.14 and A.3.15, and therefore we omit them.

Theorem A.3.18. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.2.3) and Σ_{skcd} satisfies the OT-CD security (Definition 2.2.35), Σ_{cepk} is IND-CPA secure (Definition A.3.12).*

Theorem A.3.19. *If Σ_{pk} satisfies the OW-CPA security (Definition 2.2.3) and Σ_{skcd} satisfies the OT-CD security (Definition 2.2.35), Σ_{cepk} is certified everlasting IND-CPA secure (Definition A.3.13).*

A.3.5 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 $\Sigma_{\text{cepk}} = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$, 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 2.2.1) $\Sigma_{\text{pk}} = \text{PKE}(\text{KeyGen}, \text{Enc}, \text{Dec})$ 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.)

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

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

$\text{Enc}(\text{pk}, m)$:

- Parse $\text{pk} = \text{pke.pk}$.
- 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 $\text{pke.ct} \leftarrow \text{PKE.Enc}(\text{pke.pk}, (B, Q, r, y))$.
- Let U_Q be the unitary that permutes the qubits in Q into the first half of the system. (I.e., $U_Q |x_1 x_2 \cdots x_{p+q}\rangle = |x_{a_1} x_{a_2} \cdots x_{a_p} x_{b_1} x_{b_2} \cdots x_{b_q}\rangle$ with $Q := \{a_1, a_2, \dots, a_p\}$ and $\{1, 2, \dots, p+q\} \setminus Q := \{b_1, b_2, \dots, b_q\}$.)
- 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 $ct := (|\Psi\rangle, \text{pke.ct}, u, h)$ and $\text{vk} := (B, Q, r)$.

$\text{Dec}(\text{sk}, ct)$:

- Parse $\text{sk} = \text{pke.sk}$ and $ct = (|\Psi\rangle, \text{pke.ct}, u, h)$.
- Compute $(B, Q, r, y) \leftarrow \text{PKE.Dec}(\text{pke.sk}, \text{pke.ct})$.
- 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 $m' := h \oplus x \oplus y$.

$\text{Del}(ct)$:

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

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

- Parse $\text{vk} = (B, Q, r)$ and $\text{cert} = |\Psi\rangle$.
- 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 $\perp$ otherwise.

Correctness. Correctness easily follows from that of Σ_{pk} . Verification correctness with QOTP holds since Recover is the decryption algorithm of QOTP.

Security. The following two theorems hold.

Theorem A.3.20. *If Σ_{pk} is IND-CPA secure (Definition 2.2.4), then Σ_{cepk} is IND-CPA secure (Definition A.3.12).*

Its proof is straightforward, and therefore we omit it.

Theorem A.3.21. *If Σ_{pk} is IND-CPA secure (Definition 2.2.4) and $tp/(p+q) - 4(k_1 - k_2)\ln 2$ is superlogarithmic, then Σ_{cepk} is certified everlasting IND-CPA secure (Definition A.3.13).*

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.

A.4 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 A.4.1, we define RNCE with certified everlasting deletion. In Appendix A.4.2, we construct a certified everlasting RNCE scheme from certified everlasting secure PKE (Appendix A.3).

A.4.1 Definition

Definition A.4.1 (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 $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal}, \text{Del}, \text{Vrfy})$ 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 $\mathcal{AUX} := \{0, 1\}^{v(\lambda)}$.*

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

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

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

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

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

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

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

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

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

Definition A.4.2 (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 $m \in \mathcal{M}$,*

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

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

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}) = \perp \mid \begin{array}{l} (\text{pk}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda) \\ (\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m) \\ \text{cert} \leftarrow \text{Del}(\text{ct}) \end{array} \right] \leq \text{negl}(\lambda).$$

As security, we consider two definitions, Definition A.4.3 and Definition A.4.4 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 A.4.3 (Receiver Non-Committing Security for RNCE with Certified Everlasting Deletion).

Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal}, \text{Del}, \text{Vrfy})$ be an RNCE with certified everlasting deletion scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{rec-nc}}(\lambda, b)$ against a QPT adversary $\mathcal{A}$.

1. The challenger runs $(\text{pk}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ 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 $(\text{vk}, \text{ct}) \leftarrow \text{Enc}(\text{pk}, m)$ and $\text{sk} \leftarrow \text{KeyGen}(\text{MSK})$, and sends (ct, sk) to $\mathcal{A}$.
 - If $b = 1$, the challenger generates $(\text{vk}, \tilde{\text{ct}}, \text{aux}) \leftarrow \text{Fake}(\text{pk})$ and $\tilde{\text{sk}} \leftarrow \text{Reveal}(\text{pk}, \text{MSK}, \text{aux}, m)$, and sends $(\tilde{\text{ct}}, \tilde{\text{sk}})$ 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

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

Definition A.4.4 (Certified Everlasting RNC Security for RNCE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal}, \text{Del}, \text{Vrfy})$ be a certified everlasting RNCE scheme. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-rec-nc}}(\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$.

1. The challenger runs $(\text{pk}, \text{MSK}) \leftarrow \text{Setup}(1^\lambda)$ 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 $(vk, ct) \leftarrow \mathcal{Enc}(pk, m)$ and $sk \leftarrow \text{KeyGen}(\text{MSK})$, and sends (ct, sk) to $\mathcal{A}_1$.
 - If $b = 1$, the challenger generates $(vk, \tilde{ct}, \text{aux}) \leftarrow \mathcal{Fake}(pk)$ and $\tilde{sk} \leftarrow \text{Reveal}(pk, \text{MSK}, \text{aux}, m)$, and sends $(\tilde{ct}, \tilde{sk})$ 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 $\text{Vrfy}(vk, \text{cert})$. If the output is $\top$, the challenger outputs $\top$ and sends MSK to $\mathcal{A}_2$. If the output is $\perp$, the challenger outputs $\perp$ and sends $\perp$ to $\mathcal{A}_2$.
 6. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
 7. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

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

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

A.4.2 Construction

In this section, we construct a certified everlasting RNCE scheme from a certified everlasting PKE scheme (Definition A.3.8). 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 $\Sigma_{\text{cence}} = (\text{Setup}, \text{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Fake}, \text{Reveal}, \mathcal{Del}, \text{Vrfy})$ from a certified everlasting secure PKE scheme $\Sigma_{\text{cepk}} = \text{PKE}(\text{KeyGen}, \mathcal{Enc}, \mathcal{Dec}, \mathcal{Del}, \text{Vrfy})$, which was introduced in Definition A.3.8.

Setup(1^λ):

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

KeyGen(MSK):

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

Enc(pk, m):

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

Dec(sk, ct):

- Parse $\text{sk} = (x, \{\text{pke.sk}_i\}_{i \in [n]})$ and $\text{ct} = \{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}$.

- Compute $m[i] \leftarrow \text{PKE.Dec}(\text{pke.sk}_i, \text{pke.ct}_{i,x[i]})$ for all $i \in [n]$.
- Output $m := m[1] || m[2] || \dots || m[n]$.

$\text{Fake}(\text{pk})$:

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

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

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

$\text{Del}(ct)$:

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

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

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

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

Security: The following two theorems hold.

Theorem A.4.5. *If Σ_{cepk} is IND-CPA secure (Definition A.3.12), Σ_{cence} is RNC secure (Definition A.4.3).*

Its proof is similar to that of Theorem A.4.6, and therefore we omit it.

Theorem A.4.6. *If Σ_{cepk} is certified everlasting IND-CPA secure (Definition A.3.13), Σ_{cence} is certified everlasting RNC secure (Definition A.4.4).*

Proof of Theorem A.4.6. To prove the theorem, let us introduce the sequence of hybrids.

Hyb_0 : This is identical to $\text{Exp}_{\Sigma_{\text{cence}}, \mathcal{A}}^{\text{cert-ever-rec-nc}}(\lambda, 0)$. 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 $(\text{pke.pk}_{i,\alpha}, \text{pke.sk}_{i,\alpha}) \leftarrow \text{PKE.KeyGen}(1^\lambda)$ for all $i \in [n]$ and $\alpha \in \{0,1\}$.
2. The challenger sends $\{\text{pke.pk}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}$ 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 $(\text{pke.vk}_{i,\alpha}, \text{pke.ct}_{i,\alpha}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,\alpha}, m[i])$ for all $i \in [n]$ and $\alpha \in \{0, 1\}$, and sends $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}, (x, \{\text{pke.sk}_{i,x[i]}\}_{i \in [n]}))$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ sends $\{\text{pke.cert}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}$ to the challenger and its internal state to $\mathcal{A}_2$.
6. The challenger computes $\text{PKE.Vrfy}(\text{pke.vk}_{i,\alpha}, \text{pke.cert}_{i,\alpha})$ for all $i \in [n]$ and $\alpha \in \{0, 1\}$. If all results are $\top$, the challenger outputs $\top$ and sends $\{\text{pke.sk}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$ and sends $\perp$ to $\mathcal{A}_2$.
7. $\mathcal{A}_2$ outputs $b' \in \{0, 1\}$.
8. If the challenger outputs $\top$, then the output of the experiment is b' . Otherwise, the output of the experiment is $\perp$.

Hyb₁: This is identical to Hyb₀ except that the challenger generates $(\text{pke.vk}_{i,x[i] \oplus 1}, \text{pke.ct}_{i,x[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x[i] \oplus 1}, m[i] \oplus 1)$ for all $i \in [n]$ instead of computing $(\text{pke.vk}_{i,x[i] \oplus 1}, \text{pke.ct}_{i,x[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x[i] \oplus 1}, m[i])$ for all $i \in [n]$.

Hyb₂: This is identical to Hyb₁ 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 $(\text{pke.vk}_{i,x^*[i]}, \text{pke.ct}_{i,x^*[i]}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x^*[i]}, 0)$ and $(\text{pke.vk}_{i,x^*[i] \oplus 1}, \text{pke.ct}_{i,x^*[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x^*[i] \oplus 1}, 1)$ instead of computing $(\text{pke.vk}_{i,x[i]}, \text{pke.ct}_{i,x[i]}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x[i]}, m[i])$ and $(\text{pke.vk}_{i,x[i] \oplus 1}, \text{pke.ct}_{i,x[i] \oplus 1}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x[i] \oplus 1}, m[i] \oplus 1)$.
3. The challenger sends $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}, (x^* \oplus m, \{\text{pke.sk}_{i,x^*[i] \oplus m[i]}\}_{i \in [n]}))$ to $\mathcal{A}_1$ instead of sending $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}, (x, \{\text{pke.sk}_{i,x[i]}\}_{i \in [n]}))$ to $\mathcal{A}_1$.

It is clear that Hyb₀ is identical to $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-rec-nc}}(\lambda, 0)$ and Hyb₂ is identical to $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{cert-ever-rec-nc}}(\lambda, 1)$. Hence, Theorem A.4.6 easily follows from the following Propositions A.4.7 and A.4.8 (whose proof is given later.). $\square$

Proposition A.4.7. *If Σ_{cepk} is certified everlasting IND-CPA secure, it holds that $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]| \leq \text{negl}(\lambda)$.*

Proposition A.4.8. $|\Pr[\text{Hyb}_1 = 1] - \Pr[\text{Hyb}_2 = 1]| \leq \text{negl}(\lambda)$.

Proof of Proposition A.4.7. For the proof, we use Lemma A.4.9. We assume that $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$ defined in Lemma A.4.9. This contradicts the certified everlasting IND-CPA security of Σ_{cepk} from Lemma A.4.9. Let us describe how $\mathcal{B}$ works below.

1. $\mathcal{B}$ receives $\{\text{pke.pk}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}$ from the challenger of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$.
2. $\mathcal{B}$ sends $\{\text{pke.pk}_{i,\alpha}\}_{i \in [n], \alpha \in \{0, 1\}}$ 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 $(x, m[1], \dots, m[n], m[1] \oplus 1, \dots, m[n] \oplus 1)$ to the challenger of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$.

5. $\mathcal{B}$ receives $(\{\text{pke.sk}_{i,x[i]}\}_{i \in [n]}, \{\text{pke.ct}_{i,x[i] \oplus 1}\}_{i \in [n]})$ from the challenger of $\text{Exp}_{\Sigma_{\text{cepk}}, \mathcal{B}}^{\text{multi-cert-ever}}(\lambda, b)$.
6. $\mathcal{B}$ computes $(\{\text{pke.vk}_{i,x[i]}\}_{i \in [n]}, \{\text{pke.ct}_{i,x[i]}\}_{i \in [n]}) \leftarrow \text{PKE.Enc}(\text{pke.pk}_{i,x[i]}, m[i])$ for $i \in [n]$.
7. $\mathcal{B}$ sends $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}, (x, \{\text{pke.sk}_{i,x[i]}\}_{i \in [n]}))$ to $\mathcal{A}_1$.
8. $\mathcal{A}_1$ sends $\{\text{pke.cert}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}$ to $\mathcal{B}$ and the internal state to $\mathcal{A}_2$.
9. $\mathcal{B}$ sends $\{\text{pke.cert}_{i,x[i] \oplus 1}\}_{i \in [n]}$ to the challenger, and receives $\{\text{pke.sk}_{i,x[i] \oplus 1}\}_{i \in [n]}$ or $\perp$. If $\mathcal{B}$ receives $\perp$, it outputs $\perp$ and aborts.
10. $\mathcal{B}$ sends $\{\text{pke.sk}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}$ to $\mathcal{A}_2$.
11. $\mathcal{A}_2$ outputs b' .
12. $\mathcal{B}$ computes $\text{PKE.Vrfy}(\text{pke.vk}_{i,x[i]}, \text{pke.cert}_{i,x[i]})$ for all $i \in [n]$. If all results are $\top$, $\mathcal{B}$ outputs b' . Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_0 = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_1 = 1]$. By assumption, $|\Pr[\text{Hyb}_0 = 1] - \Pr[\text{Hyb}_1 = 1]|$ 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 A.4.9. $\square$

Proof of Proposition A.4.8. It is obvious that the joint probability distribution that $\mathcal{A}_1$ receives $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}, (x, \{\text{pke.sk}_{i,x[i]}\}_{i \in [n]}))$ in Hyb_1 is identical to the joint probability distribution that $\mathcal{A}_1$ receives $(\{\text{pke.ct}_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}, (x^* \oplus m, \{\text{pke.sk}_{i,x^*[i] \oplus m[i]}\}_{i \in [n]}))$ in Hyb_2 . Hence, Proposition A.4.8 follows. $\square$

We use the following lemma for the proof of Theorem A.4.6 and Theorem A.5.7. The proof is shown with the standard hybrid argument. It is also easy to see that a similar lemma holds for IND-CPA security.

Lemma A.4.9. *Let s be some polynomial of the security parameter λ . Let $\Sigma := (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a certified everlasting secure PKE scheme. Let us consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{multi-cert-ever}}(\lambda, b)$ against $\mathcal{A}$ consisting of any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.*

1. *The challenge generates $(\text{pk}_{i,\alpha}, \text{sk}_{i,\alpha}) \leftarrow \text{KeyGen}(1^\lambda)$ for all $i \in [s]$ and $\alpha \in \{0,1\}$, and sends $\{\text{pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ to $\mathcal{A}_1$.*
2. *$\mathcal{A}_1$ chooses $f \in \{0,1\}^s$ and $(m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s])$ to the challenger.*
3. *The challenger computes $(\text{vk}_{i,f[i] \oplus 1}, \text{ct}_{i,f[i] \oplus 1}) \leftarrow \text{Enc}(\text{pk}_{i,f[i] \oplus 1}, m_b[i])$ for all $i \in [s]$, and sends $(\{\text{sk}_{i,f[i]}\}_{i \in [s]}, \{\text{ct}_{i,f[i] \oplus 1}\}_{i \in [s]})$ to $\mathcal{A}_1$.*
4. *At some point, $\mathcal{A}_1$ sends $\{\text{cert}_{i,f[i] \oplus 1}\}_{i \in [s]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.*
5. *The challenger computes $\text{Vrfy}(\text{vk}_{i,f[i] \oplus 1}, \text{cert}_{i,f[i] \oplus 1})$ for every $i \in [s]$. If all results are $\top$, the challenger outputs $\top$, and sends $\{\text{sk}_{i,f[i] \oplus 1}\}_{i \in [s]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ 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 $\perp$.

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

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

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

Proof of Lemma A.4.9. Let us consider the following hybrids for $j \in \{0, 1, \dots, s\}$.

Hyb_j:

1. The challenger generates $(\text{pk}_{i,\alpha}, \text{sk}_{i,\alpha}) \leftarrow \text{KeyGen}(1^\lambda)$ for every $i \in [s]$ and $\alpha \in \{0, 1\}$, and sends $\{\text{pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0, 1\}}$ to $\mathcal{A}_1$.
2. $\mathcal{A}_1$ chooses $f \in \{0, 1\}^s$ and $(m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s])$ to the challenger.
3. The challenger computes

$$(\text{vk}_{i,f[i] \oplus 1}, \text{ct}_{i,f[i] \oplus 1}) \leftarrow \text{Enc}(\text{pk}_{i,f[i] \oplus 1}, m_1[i])$$

for $i \in [j]$ and

$$(\text{vk}_{i,f[i] \oplus 1}, \text{ct}_{i,f[i] \oplus 1}) \leftarrow \text{Enc}(\text{pk}_{i,f[i] \oplus 1}, m_0[i])$$

for $i \in \{j+1, j+2, \dots, s\}$, and sends $(\{\text{sk}_{i,f[i]}\}_{i \in [s]}, \{\text{ct}_{i,f[i] \oplus 1}\}_{i \in [s]})$ to $\mathcal{A}_1$.

4. At some point, $\mathcal{A}_1$ sends $\{\text{cert}_{i,f[i] \oplus 1}\}_{i \in [s]}$ to the challenger, and sends its internal state to $\mathcal{A}_2$.
5. The challenger computes $\text{Vrfy}(\text{vk}_{i,f[i] \oplus 1}, \text{cert}_{i,f[i] \oplus 1})$ for every $i \in [s]$. If all results are $\top$, the challenger outputs $\top$, and sends $\{\text{sk}_{i,f[i] \oplus 1}\}_{i \in [s]}$ to $\mathcal{A}_2$. Otherwise, the challenger outputs $\perp$, and sends $\perp$ 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 $\perp$.

It is clear that $\Pr[\text{Hyb}_0 = 1] = \Pr[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{multi-cert-ever}}(\lambda, 0) = 1]$ and $\Pr[\text{Hyb}_s = 1] = \Pr[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{multi-cert-ever}}(\lambda, 1) = 1]$. Furthermore, we can show

$$\left| \Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1] \right| \leq \text{negl}(\lambda)$$

for each $j \in \{0, 1, \dots, s-1\}$. (Its proof is given below.) From these facts, we obtain Lemma A.4.9.

Let us show the remaining one. To show it, let us assume that $\left| \Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1] \right|$ 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 $\text{Exp}_{\Sigma, \mathcal{B}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$.
2. $\mathcal{B}$ generates $\beta \leftarrow \{0, 1\}$ and sets $\text{pk}_{j+1, \beta} := \text{pk}$.

3. $\mathcal{B}$ generates $(\text{pk}_{i,\alpha}, \text{sk}_{i,\alpha}) \leftarrow \text{KeyGen}(1^\lambda)$ for $i \in \{1, \dots, j, j+2, \dots, s\}$ and $\alpha \in \{0, 1\}$, and $(\text{pk}_{j+1,\beta \oplus 1}, \text{sk}_{j+1,\beta \oplus 1}) \leftarrow \text{KeyGen}(1^\lambda)$.
4. $\mathcal{B}$ sends $\{\text{pk}_{i,\alpha}\}_{i \in [s], \alpha \in \{0,1\}}$ to $\mathcal{A}_1$.
5. $\mathcal{A}_1$ chooses $f \in \{0, 1\}^s$ and $(m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s]) \in \mathcal{M}^{2s}$, and sends $(f, m_0[1], m_0[2], \dots, m_0[s], m_1[1], m_1[2], \dots, m_1[s])$ to the challenger.
6. If $f[j+1] = \beta$, $\mathcal{B}$ aborts the experiment, and outputs $\perp$.
7. $\mathcal{B}$ computes

$$(\text{vk}_{i,f[i] \oplus 1}, \text{ct}_{i,f[i] \oplus 1}) \leftarrow \text{Enc}(\text{pk}_{i,f[i] \oplus 1}, m_1[i])$$

for $i \in [j]$ and

$$(\text{vk}_{i,f[i] \oplus 1}, \text{ct}_{i,f[i] \oplus 1}) \leftarrow \text{Enc}(\text{pk}_{i,f[i] \oplus 1}, m_0[i])$$

for $i \in \{j+2, \dots, s\}$.

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

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

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

If $b = 1$ and $f[j+1] = \beta \oplus 1$, $\mathcal{B}$ simulates Hyb_{j+1} . Similarly, we have $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \frac{1}{2} \Pr[\text{Hyb}_{j+1} = 1]$. By assumption, $|\Pr[\text{Hyb}_j = 1] - \Pr[\text{Hyb}_{j+1} = 1]|$ 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 . $\square$

A.5 Garbling Scheme with Certified Everlasting Deletion

In Appendix A.5.1, we define a garbling scheme with certified everlasting deletion. In Appendix A.5.2, we construct a certified everlasting secure garbling scheme from a certified everlasting secure SKE scheme.

A.5.1 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 $\tilde{C}$ (i.e., an output of Grbl) is a quantum state.

Definition A.5.1 (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 $\Sigma = (\text{Setup}, \text{Garble}, \text{Eval}, \text{Del}, \text{Vrfy})$ 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 $\mathcal{D} := \mathcal{Q}^{\otimes s(\lambda)}$.

$\text{Setup}(1^\lambda) \rightarrow \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}$: 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 $\alpha \in \{0, 1\}$.

$\text{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}) \rightarrow (\tilde{C}, \text{vk})$: 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 $\tilde{C} \in \mathcal{C}$ and a verification key $\text{vk} \in \mathcal{VK}$.

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

$\text{Del}(\tilde{C}) \rightarrow \text{cert}$: The deletion algorithm takes $\tilde{C}$ as input, and outputs a certificate $\text{cert} \in \mathcal{D}$.

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

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

Definition A.5.2 (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 $x \in \{0, 1\}^n$,

$$\Pr \left[y \neq C(x) \mid \begin{array}{l} \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}} \leftarrow \text{Setup}(1^\lambda) \\ (\tilde{C}, \text{vk}) \leftarrow \text{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}) \\ y \leftarrow \text{Eval}(\tilde{C}, \{L_{i,x_i}\}_{i \in [n]}) \end{array} \right] \leq \text{negl}(\lambda).$$

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

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}) = \perp \mid \begin{array}{l} \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}} \leftarrow \text{Setup}(1^\lambda) \\ (\tilde{C}, \text{vk}) \leftarrow \text{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}) \\ \text{cert} \leftarrow \text{Del}(\tilde{C}) \end{array} \right] \leq \text{negl}(\lambda).$$

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 4.4.3.

Definition A.5.3 (Verification Correctness with QOTP). *There exists a negligible function negl and a PPT algorithm Recover such that for any $\lambda \in \mathbb{N}$,*

$$\Pr \left[\text{Vrfy}(\text{vk}, \text{cert}^*) = \perp \mid \begin{array}{l} \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}} \leftarrow \text{Setup}(1^\lambda) \\ (\tilde{C}, \text{vk}) \leftarrow \text{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}}) \\ a, b \leftarrow \{0,1\}^{q(\lambda)} \\ \widetilde{\text{cert}} \leftarrow \text{Del}(Z^b X^a \tilde{C} X^a Z^b) \\ \text{cert}^* \leftarrow \text{Recover}(a, b, \widetilde{\text{cert}}) \end{array} \right] \leq \text{negl}(\lambda).$$

As security, we consider two definitions, Definition A.5.4 and Definition A.5.5 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 $\tilde{C}$ 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 A.5.4 (Selective Security for Garbling Scheme with Certified Everlasting Deletion). *Let $\Sigma = (\text{Setup}, \text{Garble}, \text{Eval}, \text{Del}, \text{Vrfy})$ be a garbling scheme with certified everlasting deletion. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{sel-gbl}}(1^\lambda, b)$ against a QPT adversary $\mathcal{A}$. Let Sim be a QPT algorithm.*

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

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

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

Definition A.5.5 (Selective Certified Everlasting Security for Garbling Scheme). *Let $\Sigma = (\text{Setup}, \text{Garble}, \text{Eval}, \text{Del}, \text{Vrfy})$ be a garbling scheme with certified everlasting deletion. We consider the following security experiment $\text{Exp}_{\mathcal{A}_1, \Sigma}^{\text{cert-ever-sel-gbl}}(1^\lambda, b)$ against a QPT adversary $\mathcal{A}_1$ and an unbounded adversary $\mathcal{A}_2$. Let Sim be a QPT algorithm.*

1. $\mathcal{A}_1$ sends a circuit $C \in \mathcal{C}_n$ and an input $x \in \{0,1\}^n$ to the challenger.
2. The challenger computes $\{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}} \leftarrow \text{Setup}(1^\lambda)$.
3. If $b = 0$, the challenger computes $(\tilde{C}, \text{vk}) \leftarrow \text{Garble}(1^\lambda, C, \{L_{i,\alpha}\}_{i \in [n], \alpha \in \{0,1\}})$, and returns $(\tilde{C}, \{L_{i,x_i}\}_{i \in [n]})$ to $\mathcal{A}_1$. If $b = 1$, the challenger computes $(\tilde{C}, \text{vk}) \leftarrow \text{Sim}(1^\lambda, 1^{|C|}, C(x), \{L_{i,x_i}\}_{i \in [n]})$, and returns $(\tilde{C}, \{L_{i,x_i}\}_{i \in [n]})$ 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 $\text{Vrfy}(\text{vk}, \text{cert})$. If the output is $\perp$, then the challenger outputs $\perp$, and sends $\perp$ 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 $\perp$.

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

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

A.5.2 Construction

In this section, we construct a certified everlasting secure garbling scheme from a certified everlasting secure SKE scheme (Definition A.3.1). 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, $\text{gate}_1, \text{gate}_2, \dots, \text{gate}_q$, where q is the number of gates in the circuit. Here, $\text{gate}_i := (g, w_a, w_b, w_c)$, where $g : \{0, 1\}^2 \rightarrow \{0, 1\}$ 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 $\text{out}_1, \text{out}_2, \dots, \text{out}_m$ 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 $\text{gate}_1, \text{gate}_2, \dots, \text{gate}_q$ are arranged in the ascending order of the level.

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

Setup(1^λ):

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

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

- For each $i \in \{n+1, \dots, p\}$ and $\sigma \in \{0, 1\}$, generate $\text{ske.sk}_i^\sigma \leftarrow \text{SKE.KeyGen}(1^\lambda)$.

- For each $i \in [q]$, compute

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

where $\text{gate}_i = (g, w_a, w_b, w_c)$ and GateGrbl is described in Fig A.4.

- For each $i \in [m]$, set $\tilde{d}_i := [(\text{ske.sk}_{\text{out}_i}^0, 0), (\text{ske.sk}_{\text{out}_i}^1, 1)]$.
- Output $\tilde{C} := (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$ and $\text{vk} := \{\text{vk}_i\}_{i \in [q]}$.

$\text{Eval}(\tilde{C}, \{L_{i,x_i}\}_{i \in [n]}):$

- Parse $\tilde{C} = (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$ and $\{L_{i,x_i}\}_{i \in [n]} = \{\text{ske.sk}'_i\}_{i \in [n]}$.
- For each $i \in [q]$, compute $\text{ske.sk}'_c \leftarrow \text{GateEval}(\tilde{g}_i, \text{ske.sk}'_a, \text{ske.sk}'_b)$ in the ascending order of the level, where GateEval is described in Fig A.5. If $\text{ske.sk}'_c = \perp$, output $\perp$ and abort.
- For each $i \in [m]$, set $y[i] = \sigma$ if $\text{ske.sk}'_{\text{out}_i} = \text{ske.sk}_{\text{out}_i}^\sigma$. Otherwise, set $y[i] = \perp$, and abort.
- Output $y := y[1] || y[2] || \dots || y[m]$.

$\text{Del}(\tilde{C}):$

- Parse $\tilde{C} = (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$.
- For each $i \in [q]$, compute $\text{cert}_i \leftarrow \text{GateDel}(\tilde{g}_i)$, where GateDel is described in Fig A.6.
- Output $\text{cert} := \{\text{cert}_i\}_{i \in [q]}$.

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

- Parse $\text{vk} = \{\text{vk}_i\}_{i \in [q]}$ and $\text{cert} = \{\text{cert}_i\}_{i \in [q]}$.
- For each $i \in [q]$, compute $\perp / \top \leftarrow \text{GateVrfy}(\text{vk}_i, \text{cert}_i)$, where GateVrfy is described in Fig A.7.
- If $\top \leftarrow \text{GateVrfy}(\text{vk}_i, \text{cert}_i)$ for all $i \in [q]$, then output $\top$. Otherwise, output $\perp$.

Gate Garbling Circuit GateGrbl

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

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

1. Parse $\text{gate}_i = (g, w_a, w_b, w_c)$.
2. Sample $\gamma_i \leftarrow S_4$.^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 $(\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_a^{\sigma_a, \sigma_b}) \leftarrow \text{SKE.Enc}(\text{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a, \sigma_b})$ and $(\text{ske.vk}_b^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}) \leftarrow \text{SKE.Enc}(\text{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a, \sigma_b} \oplus \text{ske.sk}_c^g(\sigma_a, \sigma_b))$.
5. Output $\tilde{g}_i := \{\text{ske.ct}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}\}_{\sigma_a, \sigma_b \in \{0,1\}}$ in the permuted order of γ_i and $\text{vk}_i := \{\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.vk}_b^{\sigma_a, \sigma_b}\}_{\sigma_a, \sigma_b \in \{0,1\}}$ in the permuted order of γ_i .

^a S_4 is the symmetric group of order 4.

Figure A.4: The description of GateGrbl

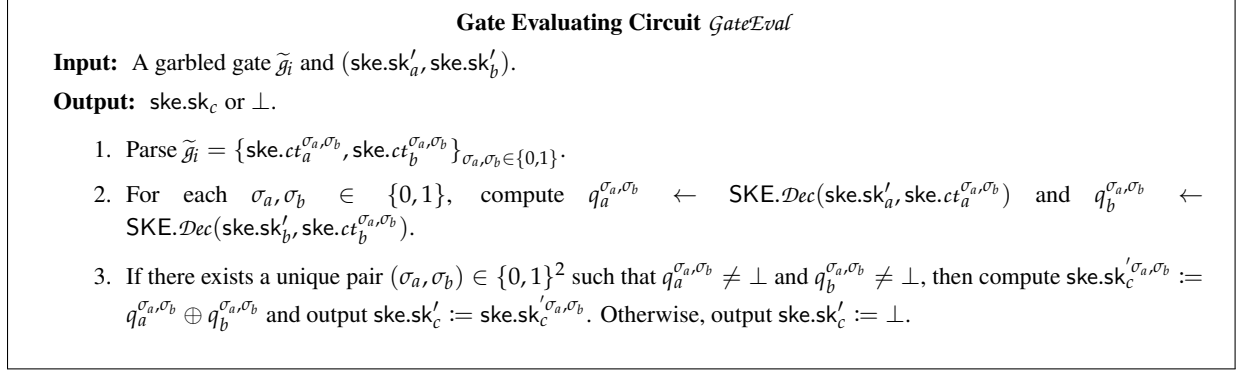


Figure A.5: The description of $GateEval$

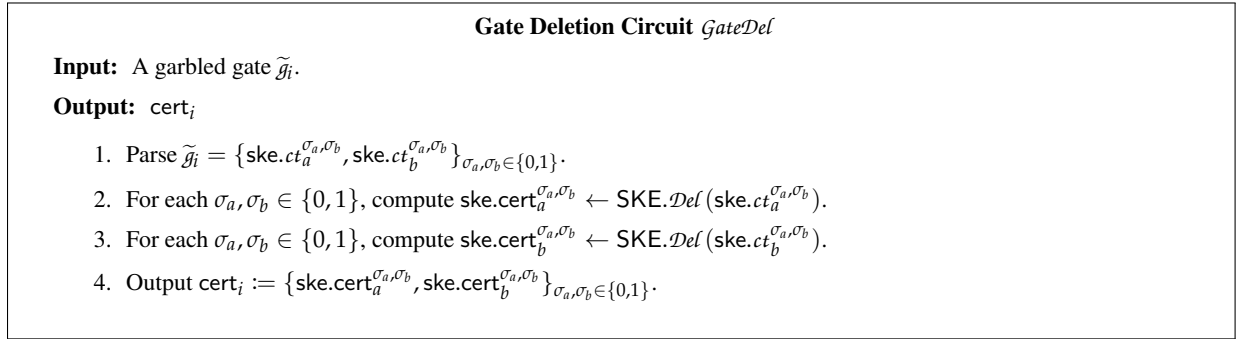


Figure A.6: The description of $GateDel$

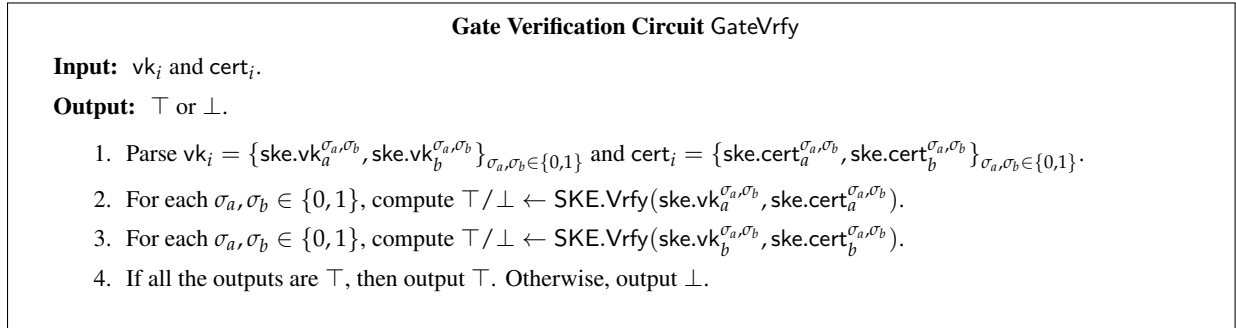


Figure A.7: The description of $GateVrfy$

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

Security: The following two theorems hold.

Theorem A.5.6. *If Σ_{cesk} satisfies the IND-CPA security (Definition A.3.6), Σ_{cegc} satisfies the selective security (Definition A.5.4).*

Its proof is similar to that of Theorem A.5.7, and therefore we omit it.

Theorem A.5.7. *If Σ_{cesk} satisfies the certified everlasting IND-CPA security (Definition A.3.7), Σ_{cegc} satisfies the selective certified everlasting security (Definition A.5.5).*

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

Proof of Theorem A.5.7. First, let us define a simulator Sim as follows.

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

1. Parse $\{L_{i,x_i}\}_{i \in [n]} := \{\text{ske.sk}_i^{x_i}\}_{i \in [n]}$.
2. For $i \in [n]$, generate $\text{ske.sk}_i^{x_i \oplus 1} \leftarrow \text{SKE.KeyGen}(1^\lambda)$.
3. For $i \in \{n+1, n+2, \dots, p\}$ and $\sigma \in \{0, 1\}$, generate $\text{ske.sk}_i^\sigma \leftarrow \text{SKE.KeyGen}(1^\lambda)$.
4. For each $i \in [q]$, compute $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{Sim.GateGrbl}(\widehat{\text{gate}}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0,1\}})$, where Sim.GateGrbl is described in Fig A.8 and $\widehat{\text{gate}}_i = (\perp, w_a, w_b, w_c)$.
5. For each $i \in [m]$, generate $\tilde{d}_i := \left[(\text{ske.sk}_{\text{out}_i}^0, C(x)_i), (\text{ske.sk}_{\text{out}_i}^1, C(x)_i \oplus 1) \right]$.
6. Output $\tilde{C} := (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$ and $\text{vk} := \{\text{vk}_i\}_{i \in [q]}$.

Simulation Gate Garbling Circuit Sim.GateGrbl

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

Output: $\tilde{g}_i$ 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 $\gamma_i \leftarrow \mathcal{S}_4$.
3. For each $\sigma_a, \sigma_b \in \{0, 1\}$, compute $(\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_a^{\sigma_a, \sigma_b}) \leftarrow \text{SKE.Enc}(\text{ske.sk}_a^{\sigma_a}, p_{a,b}^{\sigma_a, \sigma_b})$ and $(\text{ske.vk}_b^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}) \leftarrow \text{SKE.Enc}(\text{ske.sk}_b^{\sigma_b}, p_{c,b}^{\sigma_a, \sigma_b} \oplus \text{ske.sk}_c^0)$.
4. Output $\tilde{g}_i := \{\text{ske.ct}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}\}_{\sigma_a, \sigma_b \in \{0,1\}}$ in permuted order of γ_i and $\text{vk}_i := \{\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.vk}_b^{\sigma_a, \sigma_b}\}_{\sigma_a, \sigma_b \in \{0,1\}}$ in permuted order of γ_i .

Figure A.8: The description of Sim.GateGrbl

For each $j \in [q]$, we define a QPT algorithm (a simulator) InputDep.Sim_j as follows.

The simulator $\text{InputDep.Sim}_j(1^\lambda, C, x, \{L_{i,x_i}\}_{i \in [n]})$:

1. Parse $\{L_{i,x_i}\}_{i \in [n]} = \{\text{ske.sk}_i^{x_i}\}_{i \in [n]}$.
2. For $i \in [n]$, generate $\text{ske.sk}_i^{x_i \oplus 1} \leftarrow \text{SKE.KeyGen}(1^\lambda)$.
3. For $i \in \{n+1, n+2, \dots, p\}$ and $\sigma \in \{0, 1\}$, generate $\text{ske.sk}_i^\sigma \leftarrow \text{SKE.KeyGen}(1^\lambda)$.
4. For $i \in [j]$, compute $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{InputDep.GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0,1\}})$, where InputDep.GateGrbl is described in Fig. A.9 and $\text{gate}_i = (g, w_a, w_b, w_c)$.
5. For each $i \in \{j+1, j+2, \dots, q\}$, compute $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0,1\}})$, where GateGrbl is described in Fig A.4 and $\text{gate}_i = (g, w_a, w_b, w_c)$.
6. For each $i \in [m]$, generate $\tilde{d}_i := \left[(\text{ske.sk}_{\text{out}_i}^0, 0), (\text{ske.sk}_{\text{out}_i}^1, 1) \right]$.
7. Output $\tilde{C} := (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$ and $\text{vk} := \{\text{vk}_i\}_{i \in [q]}$.

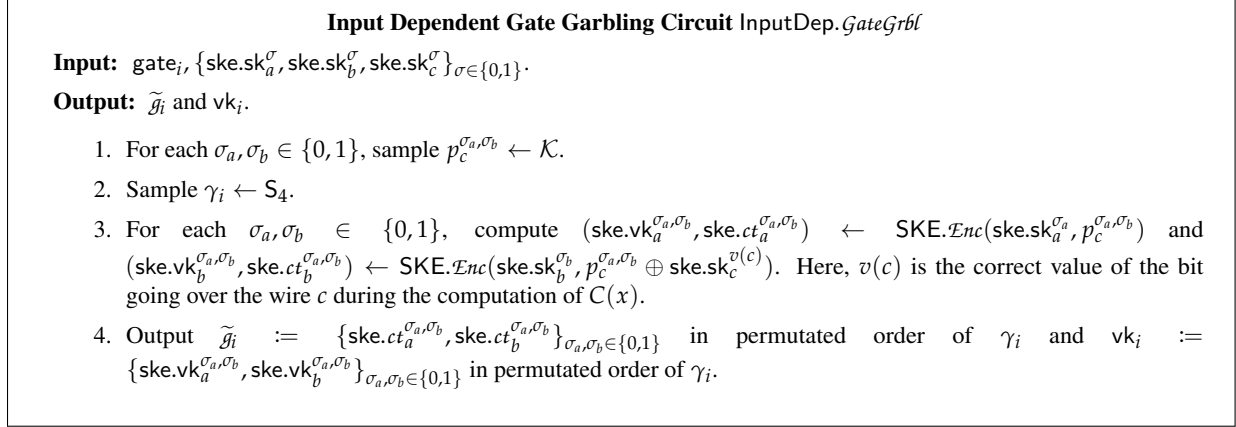


Figure A.9: The description of InputDep.GateGrbl

For each $j \in \{0, 1, \dots, q\}$, let us define a sequence of hybrid games $\{\text{Hyb}_j\}_{j \in \{0, 1, \dots, q\}}$ 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

$$\text{InputDep.Sim}_0(1^\lambda, C, x, \{L_{i, x_i}\}_{i \in [n]}) = \text{Garble}(1^\lambda, C, \{L_{i, \alpha}\}_{i \in [n], \alpha \in \{0,1\}}).$$

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 $\{L_{i, \alpha}\}_{i \in [n], \alpha \in \{0,1\}} \leftarrow \text{Smp}(1^\lambda)$.
3. The challenger computes $(\tilde{C}, \text{vk}) \leftarrow \text{GC.InputDep.Sim}_j(1^\lambda, C, x, \{L_{i, x_i}\}_{i \in [n]})$, and sends $(\tilde{C}, \{L_{i, x_i}\}_{i \in [n]})$ 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 $\text{Vrfy}(\text{vk}, \text{cert}) \rightarrow \top / \perp$. If the output is $\perp$, then the challenger outputs $\perp$ and sends $\perp$ 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 $\perp$.

Note that Hyb_0 is identical to $\text{Exp}_{\Sigma_{\text{cegc}}, \mathcal{A}}^{\text{cert-ever-sel-gbl}}(1^\lambda, 0)$ by definition. Therefore, Theorem A.5.7 easily follows from the following Propositions A.5.8 and A.5.9 (whose proofs are given later). $\square$

Proposition A.5.8. *If Σ_{cesk} satisfies the certified everlasting IND-CPA security, it holds that $\left| \Pr[\text{Hyb}_{j-1} = 1] - \Pr[\text{Hyb}_j = 1] \right| \leq \text{negl}(\lambda)$ for all $j \in [q]$.*

Proposition A.5.9. $\left| \Pr[\text{Hyb}_q = 1] - \Pr[\text{Exp}_{\Sigma_{\text{cegc}}, \mathcal{A}}^{\text{cert-ever-sel-gbl}}(1^\lambda, 1) = 1] \right| \leq \text{negl}(\lambda)$.

Proof of Proposition A.5.8. For the proof, we use Lemma A.5.10 whose statement and proof are given later. We construct an adversary $\mathcal{B}$ that breaks the security experiment of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$, which

is described in Lemma A.5.10, assuming that $\left| \Pr[\text{Hyb}_{j-1} = 1] - \Pr[\text{Hyb}_j = 1] \right|$ is non-negligible. This contradicts the certified everlasting IND-CPA security of Σ_{cesk} from Lemma A.5.10. 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 $\text{gate}_j = (g, w_\alpha, w_\beta, w_\gamma)$.
2. The challenger of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$ generates $\text{ske.sk}_\alpha^{v(\alpha) \oplus 1} \leftarrow \text{SKE.KeyGen}(1^\lambda)$ and $\text{ske.sk}_\beta^{v(\beta) \oplus 1} \leftarrow \text{SKE.KeyGen}(1^\lambda)$ ².
3. For each $i \in [p] \setminus \{\alpha, \beta\}$ and $\sigma \in \{0, 1\}$, $\mathcal{B}$ generates $\text{ske.sk}_i^\sigma \leftarrow \text{SKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ generates $\text{ske.sk}_\alpha^{v(\alpha)} \leftarrow \text{SKE.KeyGen}(1^\lambda)$ and $\text{ske.sk}_\beta^{v(\beta)} \leftarrow \text{SKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ sets $\{L_{i, x_i}\}_{i \in [n]} := \{\text{ske.sk}_i^{x_i}\}_{i \in [n]}$.
4. For each $i \in [j-1]$, $\mathcal{B}$ computes $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{InputDep.GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0, 1\}})$, where InputDep.GateGrbl is described in Fig A.9 and $\text{gate}_i = (g, w_a, w_b, w_c)$. $\mathcal{B}$ calls the encryption query of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$ if it needs to use $\text{ske.sk}_\alpha^{v(\alpha) \oplus 1}$ or $\text{ske.sk}_\beta^{v(\beta) \oplus 1}$ to run $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{InputDep.GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0, 1\}})$.
5. $\mathcal{B}$ samples $p_\gamma^{v(\alpha), v(\beta)} \leftarrow \mathcal{K}$. $\mathcal{B}$ computes

$$\begin{aligned} (\text{ske.vk}_\alpha^{v(\alpha), v(\beta)}, \text{ske.ct}_\alpha^{v(\alpha), v(\beta)}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_\alpha^{v(\alpha)}, p_\gamma^{v(\alpha), v(\beta)}), \\ (\text{ske.vk}_\beta^{v(\alpha), v(\beta)}, \text{ske.ct}_\beta^{v(\alpha), v(\beta)}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_\beta^{v(\beta)}, p_\gamma^{v(\alpha), v(\beta)} \oplus \text{ske.sk}_\gamma^{v(\gamma)}). \end{aligned}$$

6. $\mathcal{B}$ sets

$$\begin{aligned} (x_0, y_0, z_0) &:= (\text{ske.sk}_\gamma^{g(v(\alpha), v(\beta) \oplus 1)}, \text{ske.sk}_\gamma^{g(v(\alpha) \oplus 1, v(\beta))}, \text{ske.sk}_\gamma^{g(v(\alpha) \oplus 1, v(\beta) \oplus 1)}), \\ (x_1, y_1, z_1) &:= (\text{ske.sk}_\gamma^{v(\gamma)}, \text{ske.sk}_\gamma^{v(\gamma)}, \text{ske.sk}_\gamma^{v(\gamma)}), \end{aligned}$$

and sends $(\text{ske.sk}_\alpha^{v(\alpha)}, \text{ske.sk}_\beta^{v(\beta)}, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma \in \{0, 1\}})$ to the challenger of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$.

7. The challenger samples $(x, y, z) \leftarrow \mathcal{K}^3$ and $(\text{ske.sk}_\alpha^{v(\alpha) \oplus 1}, \text{ske.sk}_\beta^{v(\beta) \oplus 1}) \leftarrow \text{KeyGen}(1^\lambda)$. The challenger computes

$$\begin{aligned} (\text{ske.vk}_\alpha^{v(\alpha), v(\beta) \oplus 1}, \text{ske.ct}_\alpha^{v(\alpha), v(\beta) \oplus 1}) &\leftarrow \text{Enc}(\text{ske.sk}_\alpha^{v(\alpha)}, x), \\ (\text{ske.vk}_\beta^{v(\alpha), v(\beta) \oplus 1}, \text{ske.ct}_\beta^{v(\alpha), v(\beta) \oplus 1}) &\leftarrow \text{Enc}(\text{ske.sk}_\beta^{v(\beta) \oplus 1}, x \oplus x_b), \\ (\text{ske.vk}_\alpha^{v(\alpha) \oplus 1, v(\beta)}, \text{ske.ct}_\alpha^{v(\alpha) \oplus 1, v(\beta)}) &\leftarrow \text{Enc}(\text{ske.sk}_\alpha^{v(\alpha) \oplus 1}, y), \\ (\text{ske.vk}_\beta^{v(\alpha) \oplus 1, v(\beta)}, \text{ske.ct}_\beta^{v(\alpha) \oplus 1, v(\beta)}) &\leftarrow \text{Enc}(\text{ske.sk}_\beta^{v(\beta)}, y \oplus y_b), \\ (\text{ske.vk}_\alpha^{v(\alpha) \oplus 1, v(\beta) \oplus 1}, \text{ske.ct}_\alpha^{v(\alpha) \oplus 1, v(\beta) \oplus 1}) &\leftarrow \text{Enc}(\text{ske.sk}_\alpha^{v(\alpha) \oplus 1}, z), \\ (\text{ske.vk}_\beta^{v(\alpha) \oplus 1, v(\beta) \oplus 1}, \text{ske.ct}_\beta^{v(\alpha) \oplus 1, v(\beta) \oplus 1}) &\leftarrow \text{Enc}(\text{ske.sk}_\beta^{v(\beta) \oplus 1}, z \oplus z_b), \end{aligned}$$

²Recall that $v(\alpha)$ is the correct value of the bit going over the wire α during the computation of $C(x)$.

and sends

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

to $\mathcal{B}$.

8. $\mathcal{B}$ samples $\gamma_j \leftarrow S_4$. $\mathcal{B}$ sets $\tilde{g}_j := \{\text{ske.ct}_\alpha^{\sigma_\alpha, \sigma_\beta}, \text{ske.ct}_\beta^{\sigma_\alpha, \sigma_\beta}\}_{\sigma_\alpha, \sigma_\beta \in \{0,1\}}$ in the permuted order of γ_j .
9. For each $i \in \{j+1, j+2, \dots, q\}$, $\mathcal{B}$ computes $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0,1\}})$, where $\mathcal{B}$ calls the encryption query of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$ if $\mathcal{B}$ needs to use $\text{ske.sk}_\alpha^{v(\alpha)\oplus 1}$ or $\text{ske.sk}_\beta^{v(\beta)\oplus 1}$ to run $(\text{vk}_i, \tilde{g}_i) \leftarrow \text{GateGrbl}(\text{gate}_i, \{\text{ske.sk}_a^\sigma, \text{ske.sk}_b^\sigma, \text{ske.sk}_c^\sigma\}_{\sigma \in \{0,1\}})$.
10. $\mathcal{B}$ computes $\tilde{d}_i := [(\text{ske.sk}_{\text{out}_i}^0, 0), (\text{ske.sk}_{\text{out}_i}^1, 1)]$ for $i \in [m]$, sets $\tilde{\mathcal{C}} := (\{\tilde{g}_i\}_{i \in [q]}, \{\tilde{d}_i\}_{i \in [m]})$, and sends $(\tilde{\mathcal{C}}, \{L_{i,x_i}\}_{i \in [n]})$ to $\mathcal{A}_1$.
11. At some point, $\mathcal{A}_1$ sends $\text{cert} := \{\text{cert}_i\}_{i \in [q]}$ to $\mathcal{B}$ and the internal state to $\mathcal{A}_2$, respectively.
12. $\mathcal{B}$ re-sorts $\text{cert}_j = \{\text{ske.cert}_\alpha^{\sigma_\alpha, \sigma_\beta}, \text{ske.cert}_\beta^{\sigma_\alpha, \sigma_\beta}\}_{\sigma_\alpha, \sigma_\beta \in \{0,1\}}$ according to γ_j . $\mathcal{B}$ sends

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

to the challenger of $\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, b)$ and receives $\perp$ or $(\text{ske.sk}_\alpha^{v(\alpha)\oplus 1}, \text{ske.sk}_\beta^{v(\beta)\oplus 1})$ from the challenger. $\mathcal{B}$ computes $\text{SKE.Vrfy}(\text{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \text{ske.cert}_\alpha^{v(\alpha),v(\beta)})$ and $\text{SKE.Vrfy}(\text{ske.vk}_\beta^{v(\alpha),v(\beta)}, \text{ske.cert}_\beta^{v(\alpha),v(\beta)})$. $\mathcal{B}$ computes $\text{GateVrfy}(\text{vk}_i, \text{cert}_i)$ for each $i \in \{1, 2, \dots, j-1, j+1, j+2, \dots, q\}$, where GateVrfy is described in Fig A.7. If $\mathcal{B}$ receives $(\text{ske.sk}_\alpha^{v(\alpha)\oplus 1}, \text{ske.sk}_\beta^{v(\beta)\oplus 1})$ from the challenger, $\top \leftarrow \text{SKE.Vrfy}(\text{ske.vk}_\alpha^{v(\alpha),v(\beta)}, \text{ske.cert}_\alpha^{v(\alpha),v(\beta)})$, $\top \leftarrow \text{SKE.Vrfy}(\text{ske.vk}_\beta^{v(\alpha),v(\beta)}, \text{ske.cert}_\beta^{v(\alpha),v(\beta)})$, and $\top \leftarrow \text{GateVrfy}(\text{cert}_i, \text{vk}_i)$ for all $i \in \{1, 2, \dots, j-1, j+1, j+2, \dots, q\}$, then $\mathcal{B}$ sends $\top$ to $\mathcal{A}_2$. Otherwise, $\mathcal{B}$ sends $\perp$ to $\mathcal{A}_2$, and aborts.

13. $\mathcal{B}$ outputs the output of $\mathcal{A}_2$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Hyb}_{j-1} = 1]$ and $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_j = 1]$. Therefore, if for an adversary $\mathcal{A}$, $|\Pr[\text{Hyb}_{j-1} = 1] - \Pr[\text{Hyb}_j = 1]|$ is non-negligible, then

$$|\Pr[\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1] - \Pr[\text{Exp}_{\Sigma_{\text{cesk}}, \mathcal{B}}^{\text{parallel-cert-ever}}(\lambda, 1) = 1]|$$

is non-negligible. From Lemma A.5.10, it contradicts the certified everlasting IND-CPA security of Σ_{cesk} , which completes the proof. $\square$

Proof of Proposition A.5.9. To show Proposition A.5.9, it is sufficient to prove that the probability distribution of $\tilde{\mathcal{C}}$ in $\text{Exp}_{\Sigma_{\text{ceg}}, \mathcal{A}}^{\text{cert-ever-select}}(1^\lambda, 1)$ is statistically identical to that of $\tilde{\mathcal{C}}$ in Hyb_q .

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

$$\begin{aligned} (\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_a^{\sigma_a, \sigma_b}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a, \sigma_b}), \\ (\text{ske.vk}_b^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a, \sigma_b} \oplus \text{ske.sk}_c^{v(c)}), \end{aligned}$$

and $\tilde{d}_i$ is

$$[(\text{ske.sk}_{\text{out}_i}^0, 0), (\text{ske.sk}_{\text{out}_i}^1, 1)].$$

In $\text{Exp}_{\Sigma_{\text{cegc}}, \mathcal{A}}^{\text{cert-ever-select}}(1^\lambda, 1)$, $\tilde{g}_i$ consists of $(\text{ske.ct}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b})$ where

$$\begin{aligned} (\text{ske.vk}_a^{\sigma_a, \sigma_b}, \text{ske.ct}_a^{\sigma_a, \sigma_b}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_a^{\sigma_a}, p_c^{\sigma_a, \sigma_b}), \\ (\text{ske.vk}_b^{\sigma_a, \sigma_b}, \text{ske.ct}_b^{\sigma_a, \sigma_b}) &\leftarrow \text{SKE.Enc}(\text{ske.sk}_b^{\sigma_b}, p_c^{\sigma_a, \sigma_b} \oplus \text{ske.sk}_c^0), \end{aligned}$$

and $\tilde{d}_i$ is

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

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

We use the following lemma for the proof of Proposition A.5.8. 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 A.5.10. *Let $\Sigma := (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a certified everlasting secure SKE scheme. Let us consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, b)$ against $\mathcal{A}$ consisting of any QPT adversary $\mathcal{A}_1$ and any unbounded adversary $\mathcal{A}_2$.*

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

$\mathcal{A}_1$ can repeat this process polynomially many times.

3. $\mathcal{A}_1$ generates $(\text{sk}^0, \text{sk}^1) \leftarrow \text{KeyGen}(1^\lambda)$ 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 $(\text{sk}^0, \text{sk}^1, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma \in \{0,1\}})$ to the challenger.

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

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

and sends $\{\text{ct}_x^\sigma, \text{ct}_y^\sigma, \text{ct}_z^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.

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

$\mathcal{A}_1$ can repeat this process polynomially many times.

6. $\mathcal{A}_1$ sends $\{\text{cert}_x^\sigma, \text{cert}_y^\sigma, \text{cert}_z^\sigma\}_{\sigma \in \{0,1\}}$ to the challenger, and sends the internal state to $\mathcal{A}_2$.

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

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

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

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

Proof of Lemma A.5.10. We define the following hybrid experiment.

Hyb₁: This is identical to $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0)$ except that the challenger encrypts (x_0, y_0, z_1) instead of encrypting (x_0, y_0, z_0) .

Hyb₂: This is identical to Hyb₁ except that the challenger encrypts (x_0, y_1, z_1) instead of encrypting (x_0, y_0, z_1) .

Lemma A.5.10 easily follows from the following Propositions A.5.11 to A.5.13 (whose proof is given later.). $\square$

Proposition A.5.11. *If Σ is certified everlasting IND-CPA secure, it holds that*

$$\left| \Pr \left[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1 \right] - \Pr[\text{Hyb}_1 = 1] \right| \leq \text{negl}(\lambda).$$

Proposition A.5.12. *If Σ is certified everlasting IND-CPA secure, it holds that*

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

Proposition A.5.13. *If Σ is certified everlasting IND-CPA secure, it holds that*

$$\left| \Pr[\text{Hyb}_2 = 1] - \Pr \left[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 1) = 1 \right] \right| \leq \text{negl}(\lambda).$$

Proof of Proposition A.5.11. We assume that $\left| \Pr \left[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1 \right] - \Pr[\text{Hyb}_1(1) = 1] \right|$ is non-negligible, and construct an adversary $\mathcal{B}$ that breaks the security experiment of $\text{Exp}_{\Sigma, \mathcal{B}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$. This contradicts the certified everlasting IND-CPA security of Σ . Let us describe how $\mathcal{B}$ works.

1. The challenger of $\text{Exp}_{\Sigma, \mathcal{B}}^{\text{cert-ever-ind-cpa}}(\lambda, b)$ generates $\text{sk}'^0 \leftarrow \text{KeyGen}(1^\lambda)$, and $\mathcal{B}$ generates $\text{sk}'^1 \leftarrow \text{KeyGen}(1^\lambda)$.
2. $\mathcal{A}_1$ chooses $\beta \in \{0, 1\}$, $\text{sk} \in \mathcal{K}$ 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 $(\text{vk}_m^0, \text{ct}_m^0)$ from the challenger, computes $(\text{vk}_m^1, \text{ct}_m^1) \leftarrow \text{Enc}(\text{sk}, m \oplus m^*)$, and sends $\{\text{vk}_m^\sigma, \text{ct}_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.
 - If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(\text{vk}_m^1, \text{ct}_m^1) \leftarrow \text{Enc}(\text{sk}'^1, m \oplus m^*)$ and $(\text{vk}_m^0, \text{ct}_m^0) \leftarrow \text{Enc}(\text{sk}, m^*)$ and sends $\{\text{vk}_m^\sigma, \text{ct}_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.

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

3. $\mathcal{B}$ receives $(\text{sk}^0, \text{sk}^1, \{x_\sigma, y_\sigma, z_\sigma\}_{\sigma \in \{0,1\}})$ from $\mathcal{A}_1$.
4. $\mathcal{B}$ generates $(x, y, z) \leftarrow \mathcal{M}^3$. $\mathcal{B}$ computes

$$\begin{aligned} (\text{vk}_x^0, \text{ct}_x^0) &\leftarrow \text{Enc}(\text{sk}^0, x), (\text{vk}_x^1, \text{ct}_x^1) \leftarrow \text{Enc}(\text{sk}'^1, x \oplus x_0), \\ (\text{vk}_y^1, \text{ct}_y^1) &\leftarrow \text{Enc}(\text{sk}^1, y \oplus y_0), \\ (\text{vk}_z^1, \text{ct}_z^1) &\leftarrow \text{Enc}(\text{sk}'^1, z \oplus z_0). \end{aligned}$$

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 $(\text{vk}_z^0, \text{ct}_z^0) \leftarrow \text{Enc}(\text{sk}^0, m_b)$, and sends ct_z^0 to $\mathcal{B}$.
7. $\mathcal{B}$ sends an encryption query y to the challenger, and receives $(\text{vk}_y^0, \text{ct}_y^0)$.
8. $\mathcal{B}$ sends $\{\text{ct}_x^\sigma, \text{ct}_y^\sigma, \text{ct}_z^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.
9. $\mathcal{A}_1$ chooses $\beta \in \{0, 1\}$, $\text{sk} \in \mathcal{K}$ 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 $(\text{vk}_m^0, \text{ct}_m^0)$ from the challenger, computes $(\text{vk}_m^1, \text{ct}_m^1) \leftarrow \text{Enc}(\text{sk}, m \oplus m^*)$, and sends $\{\text{vk}_m^\sigma, \text{ct}_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.

- If $\beta = 1$, $\mathcal{B}$ generates $m^* \leftarrow \mathcal{M}$, computes $(vk_m^1, ct_m^1) \leftarrow \text{Enc}(\text{sk}'^1, m \oplus m^*)$ and $(vk_m^0, ct_m^0) \leftarrow \text{Enc}(\text{sk}, m^*)$ and sends $\{vk_m^\sigma, ct_m^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_1$.

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

10. $\mathcal{A}_1$ sends $\{\text{cert}_x^\sigma, \text{cert}_y^\sigma, \text{cert}_z^\sigma\}_{\sigma \in \{0,1\}}$ 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 $\{\text{sk}'^\sigma\}_{\sigma \in \{0,1\}}$ to $\mathcal{A}_2$.
13. $\mathcal{A}_2$ outputs b' .
14. $\mathcal{B}$ computes $\text{Vrfy}(vk_x^\sigma, \text{cert}_x^\sigma)$ and $\text{Vrfy}(vk_y^\sigma, \text{cert}_y^\sigma)$ for each $\sigma \in \{0,1\}$, and $\text{Vrfy}(vk_z^1, \text{cert}_z^1)$. If all results are $\top$, $\mathcal{B}$ outputs b' . Otherwise, $\mathcal{B}$ outputs $\perp$.

It is clear that $\Pr[1 \leftarrow \mathcal{B} \mid b = 0] = \Pr[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1]$. 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 $\Pr[1 \leftarrow \mathcal{B} \mid b = 1] = \Pr[\text{Hyb}_1 = 1]$. By assumption, $|\Pr[\text{Exp}_{\Sigma, \mathcal{A}}^{\text{parallel-cert-ever}}(\lambda, 0) = 1] - \Pr[\text{Hyb}_1 = 1]|$ 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_{cesk} . $\square$

Proof of Proposition A.5.12. The proof is very similar to that of Proposition A.5.11. Therefore we skip the proof. $\square$

Proof of Proposition A.5.13. The proof is very similar to that of Proposition A.5.11. Therefore, we skip the proof. $\square$

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Tomoyuki Morimae, for introducing me to the fascinating field of quantum cryptography. His passion and integrity in research have served as an exemplary role model, demonstrating an ideal approach to academic inquiry and the joy of tackling challenging problems. His insightful advice, constructive feedback, and unwavering support have been instrumental throughout my doctoral journey. This research would not have been possible without his exceptional guidance and encouragement. I feel incredibly fortunate to have had him as my supervisor and will always admire his intellectual curiosity and breadth of expertise.

I am also sincerely grateful to my co-researchers, Dr. Ryo Nishimaki, Dr. Takashi Yamakawa, Dr. Fuyuki Kitagawa, Dr. Tapas Pal, Dr. Min-Hsiu Hsieh, and Dr. Yuki Takeuchi. Their expertise, invaluable discussions, and thoughtful feedback have significantly advanced my work. Collaborating with such outstanding researchers has been a truly enriching experience, and I am deeply thankful for the opportunities to learn from them.

I also appreciate NTT Social Informatics Laboratories for hosting me as an intern and providing a stimulating environment to deepen my research. I owe special thanks to Dr. Takashi Yamakawa for his insightful guidance, constructive suggestions, and generosity with his time throughout my internship. It was an inspiring experience to work under such an extraordinary researcher, and it has had a profound impact on my academic development.

I am also deeply thankful to my family for their unwavering support and encouragement throughout this journey. Their belief in me and their understanding during the most challenging times have been a pillar of strength, without which I could not have reached this milestone.

Additionally, I am grateful to all current and former members of Morimae Laboratory, including Ryu Hayakawa, Misaki Yonekawa, Takumi Kagitani, Takeru Utsumi, Yuki Shirakawa, Keigo Masaki, Shogo Yamada, and Yusuke Inoue, and Saku Mizuhara for their camaraderie and support. The countless discussions, shared experiences, and collaborative spirit within the lab have made my doctoral journey both productive and enjoyable.

I would also like to acknowledge the financial support provided by the JSPS Research Fellowship and JSPS KAKENHI No. JP22J21864. This support has been crucial in facilitating the progress of my research, allowing me to focus on my studies and pursue ambitious goals. I am sincerely grateful for this opportunity to contribute to the field of quantum cryptography.

Lastly, I want to express my heartfelt thanks to all the people who have interacted with and supported me in various aspects of my life.