Quantum Encryption with Certified Deletion:
Public Key and Attribute-Based

Ryo Nishimaki\(^1\) and Takashi Yamakawa\(^1\)

\(^1\)NTT Secure Platform Laboratories, Tokyo, Japan
{ryo.nishimaki.zk,takashi.yamakawa.ga}@hco.ntt.co.jp

March 24, 2021

Abstract

Broadbent and Islam (TCC ’20) proposed a quantum cryptographic primitive called quantum encryption with certified deletion. In this primitive, a receiver in possession of a quantum ciphertext can generate a classical certificate that the encrypted message is deleted. Though they proved that their construction is information theoretically secure, a drawback is that the construction is limited to the setting of one-time symmetric key encryption (SKE) where a sender and receiver have to share a common key in advance and the key can be used only once.

In this paper, we construct a (reusable-key) public key encryption (PKE) and attribute-based encryption (ABE) with certified deletion. Our PKE with certified deletion is constructed assuming the existence of IND-CPA secure PKE, and our ABE with certified deletion is constructed assuming the existence of indistinguishability obfuscation and one-way function.
## Contents

1 Introduction 1
  1.1 Our Result 1
  1.2 Related work 2

2 Preliminaries 2
  2.1 Notations 2
  2.2 Cryptographic Tools 2

3 Public Key Encryption with Certified Deletion 7
  3.1 Definition of PKE with Certified Deletion 7
  3.2 PKE with Certified Deletion from PKE and SKE with Certified Deletion 8

4 Attribute-Based Encryption with Certified Deletion 10
  4.1 Definition of ABE with Certified Deletion 10
  4.2 Non-Committing ABE from IO 12
  4.3 ABE with Certified Deletion from NCABE and SKE with Certified Deletion 16
1 Introduction

The no-cloning theorem, which means that a quantum state cannot be copied in general, is one of the most fundamental principles in quantum physics. As any classical information can be trivially copied, this indicates a fundamental difference between classical and quantum information. The no-cloning theorem has been a basis of many quantum cryptographic protocols including quantum money [Wie83] and quantum key distribution [BB84]. Recently, Broadbent and Islam [BI20] used the principle to construct quantum encryption with certified deletion. In this primitive, a sender encrypts a classical message to generate a quantum ciphertext. A receiver in possession of the quantum ciphertext and a classical decryption key can either decrypt the ciphertext or “delete” the encrypted message by generating a classical certificate. After generating a valid certificate of deletion, the receiver (or any other party) can no longer recover the message even if the decryption key is given. We remark that this functionality is classically impossible to achieve since one can simply copy a classical ciphertext and keep it so that s/he can decrypt it in any later time. They prove security of their construction without relying on any computational assumption, which ensures the information theoretical security. On the other hand, a drawback is that the construction is limited to the setting of one-time symmetric key encryption (SKE) where a sender and receiver have to share a common key in advance and the key can be used only once.

A possible application scenario of quantum encryption with certified deletion is the following. A user upload encrypted data on quantum cloud. Whenever the user wishes to delete the data, the cloud generates the certificate of the deletion and sends it to the user. After the user verifies the validity of the certificate, s/he is convinced that the data cannot be recovered even if the decryption key is accidentally leaked later. In this scenario, one-time SKE is quite inconvenient. By the one-time restriction, the user has to locally keep as many decryption keys as the number of encrypted data in the cloud, in which case there seems no advantage of uploading the data to cloud: If the user has such a large storage, s/he could have just locally kept the messages rather than uploading encryption of them to the cloud. Also, in some cases, a party other than the decryptor may want to upload data to the cloud. This would be possible if we can extend the quantum encryption with certified deletion to public key encryption (PKE). Remark that the one-time restriction is automatically resolved for PKE, which can be seen by a simple hybrid argument. Even more flexibly, a single encrypted data on the cloud may be supposed to be decrypted by multiple users according to some access control policy. Such an access control has been realized by attribute-based encryption (ABE) [SW05, GPSW06] in classical cryptography. Thus, it would be useful if we have ABE with certified deletion.

1.1 Our Result

We give formal definitions of PKE and ABE with certified deletion, and give constructions of them:

- We construct a PKE scheme with certified deletion assuming the existence of (classical) IND-CPA secure PKE. We also observe that essentially the same construction gives a reusable SKE scheme with certified deletion if we use IND-CPA secure SKE, which exists under the existence of one-way function (OWF), instead of PKE.

- We construct a (public-key) ABE scheme with certified deletion assuming the existence of indistinguishability obfuscation (iO) [BGI+12] and OWF. This construction satisfies the collusion-resistance, i.e., it is secure against adversaries that obtain arbitrarily many decryption keys.

We note that our constructions rely on computational assumptions and thus not information theoretically secure unlike the construction in [BI20]. This is unavoidable since even plain PKE or ABE cannot be information theoretically secure.

Our main technical insight is that we can combine the one-time secure SKE with certified deletion of [BI20] and plain PKE to construct PKE with certified deletion by a simple hybrid encryption if the latter satisfies receiver non-committing (RNC) security [CFGN96, JL00, CHK05]. Since it is known that PKE/SKE with RNC security can be constructed from any IND-CPA secure PKE/SKE [CHK05, KNTY19], our first result follows.

For the second result, we first give a suitable definition of RNC security for ABE that suffices for our purpose. Then we construct an ABE scheme with RNC security based on the existence of iO and OWF. By combining this with one-time SKE with certified deletion by hybrid encryption, we obtain an ABE scheme with certified deletion.

1
1.2 Related work

Before the work of [BI20], Fu and Miller [FM18] and Coiteux-Roy and Wolf [CRW19] also studied the concept of certifying deletion of information in different settings. (See [BI20] for the comparison with these works.)

The construction of quantum encryption with certified deletion in [BI20] is based on Wiesner’s conjugate coding, which is the backbone of quantum money [Wie83] and quantum key distribution [BB84]. A similar idea has been used in many constructions in quantum cryptography that include (but not limited to) revocable quantum timed-release encryption [Unr15], uncloneable quantum encryption [BL20], single-decryptor encryption [GZ20], and copy protection/secure software leasing [CMP20]. Among them, revocable quantum timed-release encryption is conceptually similar to quantum encryption with certified deletion. In this primitive, a receiver can decrypt a quantum ciphertext only after spending a certain amount of time $T$. The receiver can also choose to return the ciphertext before the time $T$ is over, in which case it is ensured that the message can no longer be recovered. As observed in [BI20], an important difference from quantum encryption with certified deletion is that the revocable quantum timed-release encryption does not have a mechanism to generate a classical certificate of deletion. Moreover, the construction in [Unr15] heavily relies on the random oracle heuristic [BR97, BDF+11], and there is no known construction without random oracles.

Kundu and Tan [KT20] constructed (one-time symmetric key) quantum encryption with certified deletion with the device-independent security, i.e., the security holds even if quantum devices are untrusted. Moreover, they show that their construction satisfies composable security.

2 Preliminaries

2.1 Notations

We introduce basic notations used in this paper.

In this paper, $x \leftarrow X$ denotes selecting an element from a finite set $X$ uniformly at random, and $y \leftarrow A(x)$ denotes assigning to $y$ the output of a 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 from $D$. Let $[\ell]$ denote the set of integers $\{1, \cdots, \ell\}$, $\lambda$ denote a security parameter, and $y := z$ denote that $y$ is set, defined, or substituted by $z$. QPT stands for quantum polynomial time. A function $f : \mathbb{N} \to \mathbb{R}$ is a negligible function if for any constant $c$, there exists $\lambda_0 \in \mathbb{N}$ such that for any $\lambda > \lambda_0$, $f(\lambda) < \lambda^{-c}$. We write $f(\lambda) \leq \text{negl}(\lambda)$ to denote $f(\lambda)$ being a negligible function.

2.2 Cryptographic Tools

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

Public key encryption.

Definition 2.1. A public key encryption scheme $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ is a triple of algorithm: a key generation KeyGen, an encryption algorithm Enc and a decryption algorithm Dec.

KeyGen$(1^\lambda) \rightarrow (pk, sk)$: The key generation algorithm takes as input the security parameter and outputs a public key $pk$ and a secret key $sk$.

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

Dec$(sk, CT) \rightarrow m'$: The decryption algorithm takes as input $sk$ and $CT$, and output a plaintext $m'$ or $\bot$.

Definition 2.2 (Correctness for PKE). For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$,

$$\Pr \left[ \text{Dec}(sk, CT) \neq m \left| \begin{array}{c} (pk, sk) \leftarrow \text{KeyGen}(1^\lambda) \\
CT \leftarrow \text{Enc}(pk, m) \end{array} \right. \right] \leq \text{negl}(\lambda).$$
Definition 2.3 (IND-CPA security). Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec})$ be a PKE scheme. For QPT adversaries $\mathcal{A}$, we define the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-CPA}}(\lambda, b)$.

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

Let $\text{Adv}_{\Sigma, \mathcal{A}}^{\text{ind-CPA}}(\lambda)$ be the advantage of the game. We say that the $\Sigma$ is $\epsilon$-IND-CPA secure if for any QPT $\mathcal{A}$, it holds that

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

When $\epsilon$ is negligible, we omit $\epsilon$ and say $\Sigma$ is IND-CPA secure.

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 the references for the LWE assumption and constructions of post-quantum secure PKE [Reg09, GPV08].

Attribute-based encryption. We review the notion of *(key-policy) attribute-based encryption (ABE) [SW05, GPSW06].

Definition 2.4 (Attribute-Based Encryption (Syntax)). An ABE scheme is a tuple of algorithms (Setup, KeyGen, Enc, Dec) with plaintext space $\mathcal{M}$, attribute space $\mathcal{X}$, and policy space $\mathcal{P}$.

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

KeyGen$(msk, P) \rightarrow sk_P$: The key generation algorithm takes as $msk$ and a policy $P \in \mathcal{P}$, and outputs a secret key $sk_P$.

Enc$(pk, X, m) \rightarrow CT_X$: The encryption algorithm takes as $pk$, an attribute $X \in \mathcal{X}$, and a plaintext $m \in \mathcal{M}$, and outputs a ciphertext $CT$.

Dec$(sk_P, CT_X) \rightarrow m^\prime$: The decryption algorithm takes as input $sk_P$ and $CT_X$, and outputs a plaintext $m^\prime$ or ⊥.

Definition 2.5 (Correctness for ABE). For any $\lambda \in \mathbb{N}$, $m \in \mathcal{M}$, $P \in \mathcal{P}$, and $X \in \mathcal{X}$ such that $P(X) = \top$,

$$\Pr\left[\text{Dec}(sk_P, CT_X) \neq m \mid (pk, msk) \leftarrow \text{Setup}(1^\lambda), \quad (sk_P \leftarrow \text{KeyGen}(msk, P), \quad CT_X \leftarrow \text{Enc}(pk, X, m))\right] \leq \negl(\lambda).$$

Definition 2.6 (IND-sel-CPA Security for ABE). Let $\Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec})$ be an ABE scheme with plaintext space $\mathcal{M}$, attribute space $\mathcal{X}$, and policy space $\mathcal{P}$. We consider the following security experiment $\text{Exp}_{\Sigma, \mathcal{A}}^{\text{ind-sel-CPA}}(\lambda, b)$.

1. $\mathcal{A}$ declare the target attribute $X^* \in \mathcal{X}$ and sends it to the challenger.
2. The challenger computes $(pk, msk) \leftarrow \text{Setup}(1^\lambda)$ and sends $pk$ to $\mathcal{A}$.
3. $\mathcal{A}$ sends a query $P \in \mathcal{P}$ to the challenger and it returns $sk_P \leftarrow \text{KeyGen}(msk, P)$ to $\mathcal{A}$. This process can be repeated polynomially many times.
4. $\mathcal{A}$ sends $(m_0, m_1) \in \mathcal{M}^2$ to the challenger.
5. The challenger computes $CT_b \leftarrow \text{Enc}(pk, X^*, m_b)$ and sends $CT_b$ to the $A$.

6. Again, $A$ can send key queries.

7. At some point, $A$ outputs the $b' \in \{0, 1\}$.

Let $\text{Adv}^{\text{ind-sel-cpa}}_{\Sigma, A}(\lambda)$ be the advantage of the experiment above. We say that the $\Sigma$ is IND-sel-CPA secure if for any QPT adversary $A$, it holds that

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

Theorem 2.7 ([GVW15]). If the LWE assumption holds against QPT adversaries, there exists IND-sel-CPA secure ABE scheme with $P = \mathbb{P}/\text{poly}$ against QPT adversaries.

Encryption with certified deletion. Broadbent and Islam introduced the notion of encryption with certified deletion [BI20]. Their notion is for secret key encryption (SKE).

Definition 2.8 (Secret Key Encryption with Certified Deletion (Syntax)). A secret key encryption scheme with certified deletion is a tuple of quantum algorithms $(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ with plaintext space $M$ and key space $K$.

- **KeyGen**$(1^\lambda) \rightarrow \text{sk}$: The key generation algorithm takes as input the security parameter and outputs a secret key $\text{sk} \in K$.
- **Enc**$(pk, m) \rightarrow CT$: The encryption algorithm takes as input the public key and a plaintext $m \in M$ and outputs a ciphertext $CT$.
- **Dec**$(\text{sk}, CT) \rightarrow m'$: The decryption algorithm takes as input the secret key and a ciphertext and output a plaintext $m' \in M$ or $\perp$.
- **Del**$(CT) \rightarrow \text{cert}$: The deletion algorithm takes as input a ciphertext and outputs a certification $\text{cert}$.
- **Vrfy**$(\text{sk}, \text{cert}) \rightarrow \top$ or $\perp$: The verification algorithm takes the secret key and a certification and outputs $\top$ or $\perp$.

Definition 2.9 (Correctness for SKE with Certified Deletion). There are two types of correctness. One is decryption correctness and the other is verification correctness.

**Decryption correctness:** For any $\lambda \in \mathbb{N}$, $m \in M$,

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

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

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

Broadbent and Islam consider a setting where a secret key is used only once (that is, one time SKE), but it is easy to extend the definition to reusable secret key setting.

Definition 2.10 (Certified Deletion Security for SKE). Let $\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ be a secret key encryption with certified deletion. We consider the following the security experiment $\text{Exp}_{\Sigma, A}^{\text{sk-cert-del}}(\lambda, b)$.

1. The challenger computes $\text{sk} \leftarrow \text{KeyGen}(1^\lambda)$.
2. $A$ sends an encryption query $m$ to the challenger and the it returns $CT \leftarrow \text{Enc}(\text{sk}, m)$ to $A$. This process can be repeated polynomially many times.
3. A sends \((m_0, m_1)\) to the challenger.

4. The challenger computes \(CT_b \leftarrow Enc(sk, m_b)\) and sends \(CT_b\) to the A.

5. Again, A can send encryption queries.

6. At some point, A sends cert to the challenger.

7. The challenger computes \(Vrfy(sk, cert)\). If the output is \(\perp\), then abort. Otherwise (the output is \(\top\)), the challenger sends the \(sk\) to the A.

8. A receives \(sk\) and outputs the \(b' \in \{0, 1\}\).

Let \(Adv_{\Sigma, A}^{sk-cert-del}(\lambda)\) be the advantage of the experiment above. We say that the \(\Sigma\) is \(\epsilon\)-IND-CPA-CD secure if for any QPT \(A\), it holds that

\[
Adv_{\Sigma, A}^{sk-cert-del}(\lambda) := |Pr[Exp_{\Sigma, A}^{sk-cert-del}(\lambda, 0) = 1] - Pr[Exp_{\Sigma, A}^{sk-cert-del}(\lambda, 1) = 1]| \leq \epsilon.
\]

If \(A\) is not allowed to send any encryption query at the second and fifth items of \(Exp_{\Sigma, A}^{sk-cert-del}(\lambda, b)\), we say that \(\Sigma\) is \(\epsilon\)-OT-CD secure. When \(\epsilon\) is negligible, we omit \(\epsilon\) and say \(\Sigma\) is IND-CPA-CD (or OT-CD) secure.

We sometimes say reusable (resp. one-time) SKE with certified deletion if it satisfies IND-CPA-CD (resp. OT-CD) security.

We emphasize that in the existing construction of SKE with certified deletion, a secret key is a classical string though a ciphertext must be a quantum state. Broadbent and Islam prove the following theorem.

**Theorem 2.11 ([B120]).** There exists OT-CD secure SKE with certified deletion with \(M = \{0, 1\}^{\ell_m}\) and \(K = \{0, 1\}^{\ell_k}\) where \(\ell_m\) and \(\ell_k\) are some polynomials, unconditionally.

**Receiver Non-Committing Encryption.** We introduce the notion of receiver non-committing encryption (RNCE) [CFGN96, JL00, CHK05], which is used in Section 3.2. We sometimes simply write NCE to mean RNCE since we consider only RNCE in this paper.

**Definition 2.12 (RNCE (syntax)).** An NCE scheme is a tuple of algorithms \((\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal})\) with plaintext space \(\mathcal{M}\).

- **KeyGen**\((1^\lambda) \rightarrow (pk, sk, aux)\): The key generation algorithm takes as input the security parameter and outputs a key pair \((pk, sk)\) and an auxiliary information \(aux\).

- **Enc**\((pk, m) \rightarrow CT\): The encryption algorithm takes as input \(pk\) and a plaintext \(m\) and outputs a ciphertext \(CT\).

- **Dec**\((sk, CT) \rightarrow m'/\perp\): The decryption algorithm takes as input \(sk\) and \(CT\) and outputs a plaintext \(m'\) or \(\perp\).

- **Fake**\((pk, sk, aux) \rightarrow \tilde{CT}\): The fake encryption algorithm takes \(pk, sk,\) and an auxiliary input \(aux\) and outputs a fake ciphertext \(\tilde{CT}\).

- **Reveal**\((pk, sk, aux, \tilde{CT}, m) \rightarrow \tilde{sk}\): The reveal algorithm takes \(pk, sk, aux,\) a fake ciphertext \(\tilde{CT}\), and a plaintext \(m\), and outputs a fake secret key \(\tilde{sk}\).

Correctness is the same as that of PKE.

**Definition 2.13 (Receiver Non-Committing (RNC) Security).** An NCE scheme is RNC secure if it satisfies the following. Let \(\Sigma = (\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal})\) be an NCE scheme. We consider the following the security experiment \(Exp_{\Sigma, A}^{rec-ne}(\lambda, b)\).

1. The challenger computes \((pk, sk, aux) \leftarrow \text{KeyGen}(1^\lambda)\).
2. A sends a query \( m \in \mathcal{M} \) to the challenger.

3. The challenger does the following.
   - If \( b = 0 \), the challenger generates \( CT \leftarrow \text{Enc}(pk, m) \) and returns \((CT, sk)\) to \( A \).
   - If \( b = 1 \), the challenger generates \( CT \leftarrow \text{Fake}(pk, sk, aux) \) and \( sk \leftarrow \text{Reveal}(pk, sk, aux, \overline{CT}, m) \) and returns \((\overline{CT}, sk)\) to \( A \).

4. \( A \) outputs the \( b' \in \{0, 1\} \).

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

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

**Theorem 2.14 ([KNTY19]).** If there exists an IND-CPA secure SKE/PKE scheme (against QPT adversaries), there exists an RNC secure secret/public key NCE scheme (against QPT adversaries) with plaintext space \( \{0, 1\}^\ell \), where \( \ell \) is some polynomial, respectively.

Note that Kitagawa, Nishimaki, Tanaka, and Yamakawa [KNTY19] prove the theorem above for the SKE case, but it is easy to extend their theorem to the PKE setting. We also note that the core idea of Kitagawa et al. is based on the observation by Canetti, Halevi, and Katz [CHK05].

**Non-interactive zero-knowledge.** We review non-interactive zero-knowledge (NIZK) which is used in Section 4.2.

**Definition 2.15 (Non-Interactive Zero-Knowledge Proofs).** A non-interactive zero-knowledge (NIZK) proof for an NP language \( \mathcal{L} \) consists of PPT algorithms \( (\text{Setup}, \text{Prove}, \text{Vrfy}) \).

- \( \text{Setup}(1^\lambda) \rightarrow \text{crs} \): The setup algorithm takes as input the security parameter \( 1^\lambda \) and outputs a common reference string \( \text{crs} \).
- \( \text{Prove}(\text{crs}, x, w) \rightarrow \pi \): The prover’s algorithm takes as input a common reference string \( \text{crs} \), a statement \( x \), and a witness \( w \) and outputs a proof \( \pi \).
- \( \text{Vrfy}(\text{crs}, x, \pi) \rightarrow \top \) or \( \bot \): The verifier’s algorithm takes as input a common reference string \( \text{crs} \), a statement \( x \), and a proof \( \pi \) and outputs \( \top \) to indicate acceptance of the proof and \( \bot \) otherwise.

A non-interactive proof must satisfy the following requirements.

**Completeness:** For all \( \lambda \in \mathbb{N} \) and all pairs \( (x, w) \in \mathcal{R} \), we have

\[
\Pr[\text{Vrfy}(\text{crs}, x, \pi) = \top | \text{crs} \leftarrow \text{Setup}(1^\lambda), \pi \leftarrow \text{Prove}(\text{crs}, x, w)] = 1.
\]

**Statistical Soundness.** For all unbounded time adversaries \( A \), if we run \( \text{crs} \leftarrow \text{Setup}(1^\lambda) \), then we have\[ \Pr[x \not\in \mathcal{L} \land \text{Vrfy}(\text{crs}, x, \pi) = \top | (x, \pi) \leftarrow A(\text{crs}, \cdot, \cdot)(1^\lambda, \text{crs})] \leq \text{negl}(\lambda). \]

**Computational Zero-Knowledge:** If there exists a PPT simulator \( \text{Sim} = (\text{Sim}_1, \text{Sim}_2) \) such that for all QPT adversaries \( A \) and for all \( (x, w) \in \mathcal{R} \), we have

\[
\Pr\left[A(1^\lambda, \text{crs}, x, \pi) = 1 \mid \text{crs} \leftarrow \text{Setup}(1^\lambda), \pi \leftarrow \text{Prove}(\text{crs}, x, w)\right] - \Pr\left[A(1^\lambda, \overline{\text{crs}}, x, \pi) = 1 \mid \overline{\text{crs}} \leftarrow \text{Sim}_1(1^\lambda, x), \pi \leftarrow \text{Sim}_2(\overline{\text{crs}}, td, x)\right] \leq \text{negl}(\lambda).
\]

**Theorem 2.16 ([PS19]).** If the LWE assumption holds against QPT adversaries, then there exists computational NIZK proof for NP against QPT adversaries.
**Definition 2.17 (Statistical Simulation-Sound NIZK).** A NIZK proof system $\Pi_{nizk}$ is statistical simulation-sound if it is hard to generate a convincing proof for a false statement even if an adversary is given a simulated proof. That is, for all statements $x$ and all unbounded time adversaries $A$, we have

$$\Pr[x^* \neq x \land x^* \notin \mathcal{L} \land \text{Vrfy}(\text{crs}, x^*, \pi^*) = \top \mid (\text{crs}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda, x), \pi \leftarrow \text{Sim}_2(\text{crs}, \text{td}, x), (x^*, \pi^*) \leftarrow A(\text{crs}, x, \pi) \leq \text{negl}(\lambda).$$

**Theorem 2.18 ([GGH+16]).** If there exist computational NIZK proof systems for $\mathcal{L}$ and non-interactive perfectly binding commitment schemes, there exists a statistical simulation-sound NIZK proof system for $\mathcal{L}$.

Non-interactive perfectly binding commitment can be constructed from IND-CPA PKE with perfect correctness [LS19]. We can obtain such PKE schemes by slightly modifying IND-CPA PKE with imperfect correctness. For example, we can obtain a variant of Regev PKE scheme that has perfect correctness from Regev PKE scheme. Thus, we obtain the following corollary from Theorems 2.16 and 2.18.

**Corollary 2.19.** If the LWE assumption holds against QPT adversaries, there exists a statistical simulation-sound NIZK proof system for $\text{NP}$.

**Definition 2.20 (Indistinguishability Obfuscator [BGI+12]).** A PPT algorithm $iO$ is an IO for a circuit class $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ if it satisfies the following two conditions.

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

$$\Pr[C'(x) = C(x) \mid C' \leftarrow iO(C)] = 1.$$

**Indistinguishability:** For any QPT distinguisher $D$ and for any pair of circuits $C_0, C_1 \in C_\lambda$ such that for any input $x$, $C_0(x) = C_1(x)$ and $|C_0| = |C_1|$, it holds that

$$\text{Adv}_{iO,\text{A}}^D(\lambda) := |\Pr[D(iO(C_0)) = 1] - \Pr[D(iO(C_1)) = 1]| \leq \text{negl}(\lambda).$$

There exist candidate constructions of IO against QPT adversaries [GP20, WW20, BDGM20].

3 Public Key Encryption with Certified Deletion

In this section, we define the notion of PKE with certified deletion, which is a natural extension of SKE with certified deletion and present how to achieve PKE with certified deletion from OT-CD secure SKE and IND-CPA (standard) PKE.

3.1 Definition of PKE with Certified Deletion

The definition of PKE with certified deletion is an extension of SKE with certified deletion. Note that a verification key for verifying a certification is generated in the encryption algorithm.

**Definition 3.1 (PKE with Certified Deletion (Syntax)).** A certified deletion public-key encryption scheme is a tuple of quantum algorithms $(\text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy})$ with plaintext space $\mathcal{M}$.

- $\text{KeyGen}(1^\lambda) \rightarrow (\text{pk}, \text{sk})$ The key generation algorithm takes as input the security parameter and outputs a key pair $(\text{pk}, \text{sk})$.
- $\text{Enc}(\text{pk}, m) \rightarrow (\text{vk}, \text{CT})$ The encryption algorithm takes as input the public key and a plaintext and outputs a verification key $\text{vk}$ and a ciphertext $\text{CT}$.
- $\text{Dec}(\text{sk}, \text{CT}) \rightarrow m'$ The decryption algorithm takes as input the secret key and a ciphertext and outputs a plaintext $m'$ or $\bot$.
- $\text{Del}(\text{CT}) \rightarrow \text{cert}$ The deletion algorithm takes as input a ciphertext and outputs a certification $\text{cert}$. 

7
Vrfy(vk, cert) → ⊤ or ⊥. The verification algorithm takes the verification key and a certificate and outputs ⊤ or ⊥.

**Definition 3.2 (Correctness for PKE with Certified Deletion).** There are two types of correctness. One is decryption correctness and the other is verification correctness.

**Decryption correctness:** For any λ ∈ N, m ∈ M,

\[
\Pr \left[ Dec(sk, CT) \neq m \bigg| (pk, sk) \leftarrow KeyGen(1^\lambda), (vk, CT) \leftarrow Enc(pk, m) \right] \leq \text{negl}(\lambda).
\]

**Verification correctness:** For any λ ∈ N, m ∈ M,

\[
\Pr \left[ Vrfy(vk, cert) = ⊥ \bigg| (pk, sk) \leftarrow KeyGen(1^\lambda), (vk, CT) \leftarrow Enc(pk, m), cert \leftarrow Del(CT) \right] \leq \text{negl}(\lambda).
\]

**Definition 3.3 (Certified Deletion Security for PKE).** Let \( \Sigma = (\text{KeyGen}, Enc, Dec, Del, Vrfy) \) be a PKE with certified deletion scheme. We consider the following the security experiment \( \text{Exp}_{\Sigma, A}^{pk\text{-cert-de}l}(\lambda, b) \).

1. The challenger computes \((pk, sk) \leftarrow \text{KeyGen}(1^\lambda)\) and sends \( pk \) to \( A \).
2. \( A \) sends \((m_0, m_1) \in M^2\) to the challenger.
3. The challenger computes \((vk_b, CT_b) \leftarrow Enc(sk, m_b)\) and sends \( CT_b \) to the \( A \).
4. At some point, \( A \) sends \( cert \) to the challenger.
5. The challenger computes \( Vrfy(vk_b, cert) \). If the output is \( ⊥ \), then abort. Otherwise (the output is \( ⊤ \)), the challenger sends the \( sk \) to the \( A \).
6. \( A \) receives \( sk \) and outputs the \( b' \in \{0, 1\} \).

Let \( \text{Adv}_{\Sigma, A}^{pk\text{-cert-de}l}(\lambda) \) be the advantage of the experiment above. We say that the \( \Sigma \) is ε-IND-CPA-CD secure if for any QPT adversary \( A \), it holds that

\[
\text{Adv}_{\Sigma, A}^{pk\text{-cert-de}l}(\lambda) := \left| \Pr[\text{Exp}_{\Sigma, A}^{pk\text{-cert-de}l}(\lambda, 0) = 1] - \Pr[\text{Exp}_{\Sigma, A}^{pk\text{-cert-de}l}(\lambda, 1) = 1] \right| \leq \varepsilon.
\]

When \( \varepsilon \) is negligible, we omit \( \varepsilon \) and say \( \Sigma \) is IND-CPA-CD secure.

### 3.2 PKE with Certified Deletion from PKE and SKE with Certified Deletion

In this section, we present how to construct a PKE scheme with certified deletion from an SKE scheme with certified deletion and NCE scheme, which can be constructed from standard IND-CPA PKE schemes.

**Our PKE Scheme.** We construct \( \Sigma_{\text{pkcd}} = (\text{KeyGen}, Enc, Dec, Del, Vrfy) \) with plaintext space \( M \) from an SKE with certified deletion scheme \( \Sigma_{\text{skcd}} = SKE(\text{Gen}, Enc, Dec, Del, Vrfy) \) with plaintext space \( M \) and key space \( K \) and a public key NCE scheme \( \Sigma_{\text{nce}} = NCE(\text{KeyGen}, Enc, Dec, Fake, Reveal) \) with plaintext space \( K \).

1. **KeyGen\((1^\lambda)\):**
   - Generate \((nce.pk, nce.sk, nce.aux) \leftarrow NCE.\text{KeyGen}(1^\lambda)\) and output \((pk, sk) \leftarrow nce.pk, nce.sk\).

2. **Enc\((pk, m)\):**
   - Parse \( pk = nce.pk \).
   - Generate \( ske.sk \leftarrow SKE.\text{Gen}(1^\lambda) \).
Theorem 3.4.

Proof. We prove the following theorem.

Security. We construct an adversary

Proposition 3.5.

Proposition 3.6.

Thus, if

Correctness. The decryption and verification correctness easily follows from the correctness of $\Sigma_{nce}$ and $\Sigma_{skcd}$.

Security. We prove the following theorem.

Theorem 3.4. If $\Sigma_{nce}$ is RNC secure and $\Sigma_{skcd}$ is OT-CD secure, $\Sigma_{pkcd}$ is IND-CPA-CD secure.

Proof. We define the following hybrid game $Hyb(b)$.

$Hyb(b)$: This is the same as $Exp_{\Sigma_{pkcd},A}^{sk-cert-del}(\lambda, b)$ except that the challenger generate the target ciphertext as follows. It generates $sk \leftarrow SKE.Gen(1^\lambda)$ and computes $nce.CT^* \leftarrow NCE.Fake(nce.pk, nce.sk, aux)$ and $ske.CT^* \leftarrow SKE.Enc(ske.sk, m_b)$. The target ciphertext is $CT^* := (nce.CT^*, ske.CT^*)$. In addition, we reveal $sk \leftarrow Reveal(nce.pk, nce.sk, nce.aux, nce.CT^*, ske.sk)$ instead of $nce.sk$.

Proposition 3.5. If $\Sigma_{nce}$ is RNC secure, $|Pr[Exp_{\Sigma_{pkcd},A}^{sk-cert-del}(\lambda, b) = 1] - Pr[Hyb(b) = 1]| \leq Adv_{\Sigma_{nce},B_1}^{pk-cert}(\lambda)$.

Proof. We construct an adversary $B_1$ that breaks the RNC security of $\Sigma_{nce}$ by using the distinguisher $D$ for these two games. First, $B_1$ is given $nce.pk$. $B_1$ generates $sk_b \leftarrow SKE.Gen(1^\lambda)$ and sends $nce.pk$ to $D$. When $D$ sends $(m_0, m_1)$, $B_1$ sends $sk_b$ to the challenger of $NCE$, receives $(nce.CT^*, sk)$ and generates $ske.CT \leftarrow SKE.Enc(ske.sk, m_b)$. $B_1$ sends $(nce.CT^*, ske.CT)$ to $D$ as the challenge ciphertext. At some point, $D$ outputs $cert$. If $SKE.Vrfy(cert) = 1$, $B_1$ sends $sk$ to $D$.

- If $(nce.CT^*, \tilde{sk}) = (NCE.Enc(nce.pk, ske.sk), nce.sk)$, $B_1$ perfectly simulates $Exp_{\Sigma_{pkcd},A}^{sk-cert-del}(\lambda, b)$.

- If $(nce.CT^*, \tilde{sk}) = (NCE.Fake(nce.pk, nce.sk, nce.aux), NCE.Reveal(nce.pk, nce.sk, nce.aux, nce.CT^*, ske.sk))$, $B_1$ perfectly simulates $Hyb(b)$.

Thus, if $D$ distinguishes the two games, we can break the RNC security. This completes the proof.

Proposition 3.6. If $\Sigma_{skcd}$ is OT-CD secure, $|Pr[Hyb(0) = 1] - Pr[Hyb(1) = 1]| \leq Adv_{\Sigma_{skcd},B_2}^{sk-cert}(\lambda)$.

Proof. We construct an adversary $B_2$ that breaks the OT-CD security of $\Sigma_{skcd}$ by using the distinguisher $D$ for these two games. First, $B_2$ generates $(nce.pk, nce.sk, nce.aux)$ via $NCE.KeyGen(1^\lambda)$ and sends $nce.pk$ to $D$. When $D$ sends $(m_0, m_1)$, $B_2$ sends $(m_0, m_1)$ to the challenger of OT-CD $SKE$, receives $ske.CT^*$, and generates $nce.CT \leftarrow NCE.Fake(nce.pk, nce.sk, nce.aux)$. $B_2$ sends $(nce.CT, ske.CT^*)$ to $D$ as the challenge ciphertext. At some point, $D$ outputs $cert$. $B_2$ passes $cert$ to the challenger of OT-CD $SKE$. If the challenger returns $ske.sk$, $B_2$ generates $sk \leftarrow NCE.Reveal(nce.pk, nce.sk, nce.aux, nce.CT, ske.sk)$ and sends $sk$ to $D$. 

9
• If \( \text{ske.CT}^* = \text{SKE.Enc(ske.sk, m_0)} \), \( B_2 \) perfectly simulates \( \text{Hyb}(0) \).

• If \( \text{ske.CT}^* = \text{SKE.Enc(ske.sk, m_1)} \), \( B_2 \) perfectly simulates \( \text{Hyb}(1) \).

Thus, if \( D \) distinguishes the two games, we can break the OT-CD security. This completes the proof.

By Propositions 3.5 and 3.6, we immediately obtain Theorem 3.4.

By Theorems 2.11, 2.14 and 3.4, we immediately obtain the following corollary.

**Corollary 3.7.** If there exists IND-CPA secure PKE against QPT adversaries, there exists IND-CPA-CD secure PKE with certified deletion.

**Reusable SKE with certified deletion.** We can construct a secret key variant of \( \Sigma_{\text{pkcd}} \) above (that is, reusable SKE with certified deletion) by replacing \( \Sigma_{\text{nce}} \) with a secret key NCE scheme. We omit the proof since it is almost the same as that of Theorem 3.4. By Theorem 2.14 and the fact that OWFs imply (reusable) SKE [HILL99, GGM86], we also obtain the following theorem.

**Theorem 3.8.** If there exists OWF against QPT adversaries, there exists IND-CPA-CD secure SKE with certified deletion.

### 4 Attribute-Based Encryption with Certified Deletion

In this section, we define the notion of attribute-based encryption (ABE) with certified deletion, which is a natural extension of ABE and PKE with certified deletion and present how to achieve ABE with certified deletion from OT-CD secure SKE, IO, and OWFs. In Section 4.1, we present the definition of ABE with certified deletion and non-committing ABE (NCABE), which is a crucial tool to achieve ABE with certified deletion. In Section 4.2, we present how to achieve NCABE from IO and standard ABE. In Section 4.3, we present how to achieve ABE with certified deletion from NCABE and OT-CD secure SKE with certified deletion.

#### 4.1 Definition of ABE with Certified Deletion

The definition of ABE with certified deletion is a natural combination of ABE and PKE with certified deletion.

**Definition 4.1 (Attribute-Based Encryption with Certified Deletion (Syntax)).** A certified deletion ABE scheme is a tuple of quantum algorithms (Setup, KeyGen, Enc, Dec, Del, Vrfy) with plaintext space \( \mathcal{M} \), attribute space \( \mathcal{X} \), and policy space \( \mathcal{P} \).

\[
\text{Setup}(1^\lambda) \rightarrow (\text{pk}, \text{msk}): \text{The setup algorithm takes as input the security parameter and outputs a public key pk and a master secret key msk.}
\]

\[
\text{KeyGen}(\text{msk}, P) \rightarrow \text{sk}_P: \text{The key generation algorithm takes as msk and a policy } P \in \mathcal{P}, \text{and outputs a secret key sk}_P.
\]

\[
\text{Enc}(\text{pk}, X, m) \rightarrow (\text{vk}, \text{CT}_X): \text{The encryption algorithm takes as input pk, an attribute } X \in \mathcal{X}, \text{and a plaintext } m \in \mathcal{M}, \text{and outputs a verification key vk and ciphertext CT}_X.
\]

\[
\text{Dec}(\text{sk}_P, \text{CT}_X) \rightarrow m': \text{The decryption algorithm takes as input sk}_P \text{ and CT}, \text{and output a plaintext } m' \in \mathcal{M} \text{ or } \perp.
\]

\[
\text{Del}(\text{CT}) \rightarrow \text{cert}: \text{The deletion algorithm takes as input CT and outputs a certification cert.}
\]

\[
\text{Vrfy}(\text{vk}, \text{cert}) \rightarrow \top \text{ or } \perp: \text{The verification algorithm takes msk and cert and outputs } \top \text{ or } \perp.
\]

**Definition 4.2 (Correctness for ABE with Certified Deletion).** There are two types of correctness. One is decryption correctness and the other is verification correctness.
**Definition 4.4 (Non-Committing Attribute-Based Encryption (Syntax)).** A non-committing (key policy) attribute-based encryption (NCABE) is a tuple of algorithms \(\langle \text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy} \rangle\) with plaintext space \(\mathcal{M}\), attribute space \(\mathcal{X}\), and policy space \(\mathcal{P}\).

\[
\text{Setup}(1^{\lambda}) \rightarrow (pk, msk): \text{The setup algorithm takes as input the security parameter and outputs a public key } pk \text{ and a master secret key } msk.
\]

\[
\text{KeyGen}(msk, P) \rightarrow sk_P: \text{The key generation algorithm takes as input msk and a policy } P \in \mathcal{P}, \text{ and outputs a secret key } sk_P.
\]

\[
\text{Enc}(pk, X, m) \rightarrow CT: \text{The encryption algorithm takes as input pk, an attribute } X \in \mathcal{X}, \text{ and a plaintext } m \in \mathcal{M}, \text{ and outputs a ciphertext } CT.
\]

\[
\text{Dec}(sk, CT) \rightarrow m'/\bot: \text{The decryption algorithm takes as input sk and CT and outputs a plaintext } m' \in \mathcal{M} \text{ or } \bot.
\]
FakeSetup($1^\lambda, \text{aux}') \rightarrow (\text{pk}, \text{aux})$: The setup algorithm takes as input the security parameter and a pre-auxiliary information aux' and outputs a public key pk, a post-auxiliary information aux.

FakeCT(pk, aux, X) \rightarrow \tilde{CT}$: The fake encryption algorithm takes pk, aux, and X \in \mathcal{X}$, and outputs a fake ciphertext CT.

FakeSK(pk, aux, P) \rightarrow \tilde{sk}$: The fake key generation algorithm takes pk, aux, and P \in \mathcal{P}$, and outputs a fake secret key sk.

Reveal(pk, aux, \tilde{CT}, m) \rightarrow \tilde{msk}$: The reveal algorithm takes pk, aux, a fake ciphertext \tilde{CT}$, and a plaintext m \in \mathcal{M}$, and outputs a fake master secret key \tilde{msk}.

Correctness is the same as that of ABE.

**Definition 4.5 (RNC Security for ABE).** An NCABE scheme is RNC secure if it satisfies the following. Let \Sigma = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Fake}, \text{Reveal})$ be an NCE scheme. We consider the following the security experiment Exp_{\Sigma, A}^{\text{nc-cpa}}(\lambda, b).

1. \text{A}$ declared the target attribute X^* \in \mathcal{X}$ and sends it to the challenger.

2. The challenger sets aux' := X$^*$.

   - If b = 0, The challenger computes (pk, msk) \leftarrow \text{Setup}(1^\lambda)$ and sends pk to A.
   - If b = 1, The challenger computes (pk, aux) \leftarrow \text{FakeSetup}(1^\lambda, \text{aux}')$ and sends pk to A.

3. \text{A}$ sends a query P_1 \in \mathcal{P}$ to the challenger. If P_1(X^*) = \top, returns nothing. Else:

   - If b = 0, the challenger returns a secret key \text{sk}_{P_1} \leftarrow \text{KeyGen}(\text{msk}, P_1).
   - If b = 1, the challenger returns a fake secret key \tilde{\text{sk}}_{P_1} \leftarrow \text{FakeSK}(pk, aux, P_1)$.

4. At some point, \text{A}$ sends a query m \in \mathcal{M}$ to the challenger. The challenger does the following.

   - If b = 0, the challenger generates CT^* \leftarrow \text{Enc}(pk, X^*, m)$ and returns (CT, msk) to A.
   - If b = 1, the challenger generates \tilde{CT}^* \leftarrow \text{FakeCT}(pk, aux, X^*)$ and \tilde{msk} \leftarrow \text{Reveal}(pk, aux, \tilde{CT}, m)$ and returns (\tilde{CT}, \tilde{msk}) to A.

5. Again \text{A}$ can send key queries.

6. \text{A}$ outputs the b' \in \{0, 1\}$.

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

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

If \text{A}$ also declares the target plaintext m at the beginning of the game above, we say that \Sigma$ is selective-message RNC secure and write its advantages as \text{Adv}_{\Sigma, A}^{\text{nc-cpa}}(\lambda)$.

Note that a selective-message RNC secure scheme is sufficient for our purpose (ABE with certified deletion) since we use the hybrid encryption technique as in Section 3.2.

**4.2 Non-Committing ABE from IO**

In this section, we construct NCABE scheme with plaintext space \{0, 1\}^{\ell_m}$, attribute space \mathcal{X} = \{0, 1\}^{\ell_\mathcal{X}}$ where \ell_m and \ell_\mathcal{X}$ are some polynomials, and policy space \mathcal{P} = \mathcal{P}/\text{poly}$ from IO for \mathcal{P}/\text{poly}$ and PKE scheme with plaintext space \{0, 1\}.
Our NCABE scheme. We present an NCABE scheme based on any IND-CPA-secure PKE scheme and IO. Let \( \Sigma_{abe} = ABE.(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}) \) be an IND-set-CPA-secure ABE scheme on the message space \( \{0, 1\} \) and \( \Pi_{msk} \) a statistically simulation-sound NIZK protocol. Let \( pk := \{\text{abe.pk}_{t,0}, \text{abe.pk}_{t,1}\}_{i \in \ell_m} \). We define an NP relation \( R_L \) defined as follows.

\[
R_L := \{(pk, \{ct_{t,0}, ct_{t,1}\}_{i \in [\ell_m]}, X), \{(m[i], r_{t,0}, r_{t,1})\}_{i \in [\ell_m]} \mid \forall i \forall b \text{ct}_{i,b} = \text{abe.Enc}(\text{abe.pk}_{i,b}, X, m[i]; r_{i,b})\}.
\]

We construct an NCABE scheme \( \Sigma_{nce} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{FakeSetup}, \text{FakeCT}, \text{FakeSK}, \text{Reveal}) \) as follows.

Setup(1\(^\lambda\), 1\(^{\ell_m}\)):
1. Generate \( (\text{abe.pk}_{t,b}, \text{abe.msk}_{t,b}) \leftarrow \text{ABE.Setup}(1\(^\lambda\)) \) for every \( i \in [\ell_m] \) and \( b \in \{0, 1\} \).
2. Choose \( z \leftarrow \{0, 1\}^{\ell_m} \).
3. Computes \( crs \leftarrow \text{NIZK.Setup}(1\(^\lambda\)) \).
4. Output \( pk := (\{\text{abe.pk}_{t,b}\}_{i \in [\ell_m], b \in \{0, 1\}}, \text{crs}) \) and \( msk := (\{\text{abe.msk}_{i,z[i]}\}_{i \in [\ell_m], z}) \).

KeyGen(msk, P):
1. Parse \( msk = (\{\text{abe.msk}_{i}\}_{i \in [\ell_m]}, z) \).
2. Generate \( sk_i \leftarrow \text{ABE.KeyGen}(\text{abe.msk}_{i}, P) \) for every \( i \in [\ell_m] \).
3. Generate and output \( sk_{P_i} := iO(D[crs, \{sk_i\}_{i}], \pi) \), where circuit \( D \) is described in Figure 1.

Enc(pk, X, m):
1. Parse \( pk = (\{\text{abe.pk}_{t,b}\}_{i \in [\ell_m], b \in \{0, 1\}}, \text{crs}) \).
2. Generate \( \text{ct}_{i,b} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{t,b}, X, m[i]) \) for every \( i \in [\ell_m] \) and \( b \in \{0, 1\} \).
3. Generate \( \pi \leftarrow \text{NIZK.Prove}(\text{crs}, d, w) \) where \( d = (\{\text{abe.pk}_{t,0}, \text{abe.pk}_{t,1}, \text{ct}_{t,0}, \text{ct}_{t,1}\}_{i \in [\ell_m]}), X \) and \( w = \langle m, \{r_{t,0}, r_{t,1}\}_{i \in [\ell_m]} \rangle \).
4. Output \( \text{ct}_{X} := (\{\text{ct}_{t,0}, \text{ct}_{t,1}\}_{i \in [\ell_m]}, \pi) \).

Dec(sk_{P}, CT_{X}):
1. Parse \( sk_{P} = D \) and \( \text{ct}_{X} = (\{\text{ct}_{t,0}, \text{ct}_{t,1}\}_{i \in [\ell_m]}, \pi) \).
2. If \( \text{NIZK.Vrfy}(\text{crs}, d, \pi) \neq \top \), output \( \bot \).
3. Compute and output \( m := D(\text{ct}_{X}) \).

FakeSetup(1\(^\lambda\), 1\(^{\ell_m}\), aux):
1. Parse aux' = \( X^* \).
2. Generate \( (\text{abe.pk}_{t,b}, \text{abe.msk}_{t,b}) \leftarrow \text{ABE.Setup}(1\(^\lambda\)) \) for every \( i \in [\ell_m] \) and \( b \in \{0, 1\} \).
3. Choose \( z^* \leftarrow \{0, 1\}^{\ell_m} \).
4. Compute \( \text{ct}_{t,z^*[i]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{t,z^*[i]}, X^*, 0) \) and \( \text{ct}_{t,1-z^*[i]} \leftarrow \text{ABE.Enc}(\text{pk}_{t,1-z^*[i]}, X^*, 1) \) for every \( i \in [\ell_m] \).
5. Computes \( (\text{ct}_{X}, d^*) \) where \( d^* = (\{\text{abe.pk}_{t,0}, \text{abe.pk}_{t,1}, \text{ct}_{t,0}^*, \text{ct}_{t,1}^*\}_{i \in [\ell_m]}, X^*) \).
6. Output \( pk := (\{\text{abe.pk}_{t,b}\}_{i \in [\ell_m], b \in \{0, 1\}}, \text{ct}_{X}) \) and aux := (\{\text{abe.msk}_{i,b}\}_{i \in [\ell_m], b \in \{0, 1\}}, \{\text{ct}_{t,b}\}_{i \in [\ell_m], z^*}) .

FakeSK(pk, aux, P):
1. Parse aux = (\{\text{abe.msk}_{i,b}\}_{i \in [\ell_m], b \in \{0, 1\}}, \{\text{ct}_{t,b}^*\}_{i \in [\ell_m], z^*}) .
2. Generate \( sk_{i}^0 \leftarrow \text{ABE.KeyGen}(\text{abe.msk}_{i,0}, P) \) for every \( i \in [\ell_m] \) and set \( sk_{P_i}^0 := \{sk_{i}^0\}_{i} \).
3. Generate and output \( \tilde{\text{sk}} := iO(D_0[\tilde{\text{crs}}, \text{sk}^0_\Lambda]) \), where circuit \( D_0 \) is described in Figure 2.

\[
\text{FakeCT}(pk, aux, X): \\
\quad 1. \ \text{Parse} \ pk = (\{\text{abe.pk}_{i,b}\}_{i \in [\ell_m], b \in \{0,1\}}, \tilde{\text{crs}}) \text{ and } aux = (\{\text{abe.msk}_{i,b}\}_{i \in [\ell_m], b \in \{0,1\}}, \{CT^*_i\}_{i \in [\ell_m]}, z^*). \\
\quad 2. \ \text{Compute} \ \tilde{\pi} \leftarrow \text{Sim}_2(\text{crs}, td, d^*). \\
\quad 3. \ \text{Outputs} \ \tilde{\text{CT}}_X := (\{CT^*_i\}_{i \in [\ell_m], b \in \{0,1\}}, \tilde{\pi}).
\]

Reveal\((pk, aux, \tilde{\text{CT}}_X, m) : \\
\quad 1. \ \text{ Parses } aux = (\{\text{abe.msk}_{i,b}\}_{i \in [\ell_m], b \in \{0,1\}}, \{CT^*_i\}_{i \in [\ell_m]}, z^*). \\
\quad 2. \ \text{Outputs} \ \tilde{\text{msk}} := (\{\text{abe.msk}_{i,z^*[i] \oplus m[i]}\}_{i \in [\ell_m]}, z^* \oplus m).
\]

**Left-or-Right Decryption Circuit \( D \)**

**Input:** A ciphertext \( CT_X \).

**Hardwired value:** \( \text{crs}, z, \) and \( \{\text{sk}_i\}_i \).

\[
\quad 1. \ \text{Parse} \ CT_X = (\{CT_{i,0}, CT_{i,1}\}_i, \pi) \\
\quad 2. \ \text{If} \ \text{NIZK.Vrfy(crs}, d, \pi) \neq \top, \text{output} \bot. \\
\quad 3. \ \text{Compute} \ m[i] \leftarrow \text{ABE.Dec(sk}_i, CT_{i,z[i]} \} \text{ for } i \in [\ell_m]. \\
\quad 4. \ \text{Output} \ m := m[1] || \cdots || m[\ell_m].
\]

**Figure 1:** The description of the left-or-right decryption circuit

**Left Decryption Circuit \( D_0 \)**

**Input:** A ciphertext \( CT_X \).

**Hardwired value:** \( \tilde{\text{crs}} \) and \( \text{sk}^0_\Lambda = \{\text{sk}^0_i\}_i \).

\[
\quad 1. \ \text{Parse} \ CT_X = (\{CT_{i,0}, CT_{i,1}\}_i, \pi) \\
\quad 2. \ \text{If} \ \text{NIZK.Vrfy(crs}, d, \pi) \neq \top, \text{output} \bot. \\
\quad 3. \ \text{Compute} \ m[i] \leftarrow \text{ABE.Dec(sk}_i, CT_{i,0} \} \text{ for } i \in [\ell_m]. \\
\quad 4. \ \text{Output} \ m := m[1] || \cdots || m[\ell_m].
\]

**Figure 2:** The description of the left decryption circuit

**Correctness.** Correctness of \( \Sigma \text{nce} \) easily follows from correctness of \( \Sigma \text{abe} \) and completeness of \( \Pi \text{nizk} \).

**Security.** We prove the following theorem.

**Theorem 4.6.** If \( \Sigma \text{abe} \) is IND-sel-CPA and \( iO \) is secure IO for \( P/poly \) and \( \Pi \text{nizk} \) is an SSS-NIZK proof system for \( \text{NP} \), \( \Sigma \text{nce} \) is selective-message RNC secure NCABE.

**Proof.** We define a sequence of hybrid games.

- **Hyb_0:** This is the same as \( \text{Exp}^{\Sigma \text{nce}-sel-\text{cpa}}_{\Sigma \text{abe}}(\Lambda, 0) \). Let \( X^*, m^*, \) and \( d^* = (\{(\text{abe.pk}_{i,0}, \text{abe.pk}_{i,1}, CT^*_{i,0}, CT^*_{i,1})\}_i, X^*) \) be the target attribute, target message, and statement of NIZK used in the target ciphertext, respectively.
We prove Propositions 4.7 to 4.10.

Proposition 4.7.

**We define more hybrid games. Recall $q$ is the total number of key queries.**

Hyb$: This is the same as Hyb except that the challenger generates a simulated NIZK proof for the target ciphertext instead of real one. That is, it generates $\hat{\pi} \leftarrow \hat{\text{Sim}}_2(\text{crs}, \text{td}, d^*)$ where $(\text{crs}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda, d^*)$, $d^* = \{(\text{abe}, \text{pk}_{i,0}, \text{abe}, \text{pk}_{i,1}, \text{CT}_i^*, \text{CT}_{i,1}^*), X^*)$, and $X^*$ is the target attribute that the adversary declares at the beginning of the game. In addition, it uses $\bar{\text{crs}}$ instead of crs as a part of pk. This change is indistinguishable by the computational ZK property of $\Pi_{\text{nizk}}$.

Hyb$: This is the same as Hyb except that the challenger uses circuit $D_0[\bar{\text{crs}}, \text{sk}_0]$ instead of $D[\text{crs}, \{\text{sk}_i\}, z]$ to generate secret keys for key queries. That is, it returns $\text{sk} = iO(D_0[\bar{\text{crs}}, \text{sk}_0])$ instead of $\text{sk} = iO(D[\text{crs}, \{\text{sk}_i\}, z])$. This change is indistinguishable by the IO security and statistical simulation-soundness of $\Pi_{\text{nizk}}$. Note that secret keys do not depend on $z$ in this game.

Hyb$: This is the same as Hyb except that the challenger generates an inconsistent target ciphertext. That is, it generates $\text{CT}_{i,1-z[i]} \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,1-z[i]}, X^*, 1 - m^*[i])$ and $\text{CT}_{i,z[i]} \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,z[i]}, X^*, m^*[i])$ instead of double encryption of $m[i]$ for all $i$. Note that the NIZK proof in the target ciphertext is simulated in this game.

Hyb$: This is the same as Hyb except that the challenger chooses $z^* \leftarrow \{0, 1\}^m$, computes $\text{CT}_{i,z[i]}^*, \text{abe} \leftarrow \text{ABE.Setup}(1^\lambda)$, $\text{CT}_{i,b}^* \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,b}, X^*, 0)$ and $\text{CT}_{i,1-z[i]}^* \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,1-z[i]}, X^*, 1)$, and sets $\text{msk} := (z^* \oplus m^*, \{\text{msk}_{i,z[i]}^* \oplus m^*[i]\})$ as a master secret key.

We prove Propositions 4.7 to 4.10.

*Proposition 4.7.* \Pr[H_{\text{Hyb}} = 1] = \Pr[H_{\text{Hyb}} = 1] \leq \text{Adv}_{\text{crypt}}^{\text{crypt}}(\lambda).

*Proof of Proposition 4.7.* The only difference of these two games is the NIZK proof in the target ciphertext. The distinguisher declares the target attribute $X^*$ and message $m^*$ at the beginning of the game since we consider the selective setting. We can construct $B_1$ for computational ZK as follows by using the distinguisher $D$. $B_1$ generates $(\text{abe}, \text{pk}_{i,b}, \text{abe}, \text{msk}_{i,b}) \leftarrow \text{ABE.Setup}(1^\lambda)$, $\text{CT}_{i,b}^* \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,b}, X^*, 0)$ and $\text{CT}_{i,1-z[i]}^* \leftarrow \text{ABE.Enc}(\text{abe}, \text{pk}_{i,1-z[i]}, X^*, 1)$, and sends $(d^*, w^*)$ to the challenger where $w^* = (m^*, \{\text{msk}_{i,b}^*, \text{crs}\})$ is the witness for $d^*$ and receives $(\text{crs}, \pi^*)$. $B_1$ sets $\text{pk} := (\{\text{abe}, \text{pk}_{i,b}\}, \text{crs})$ and $\text{CT} := (\{\text{CT}_{i,b}^*\})$. $B_1$ passes $\text{pk}$ to $D$ and simulates secret keys by using $\{\text{msk}_{i,b}^*\}$.

If $(\text{crs}, \pi^*)$ consists of $\text{crs}^* \leftarrow \text{NIZK.Setup}(1^\lambda)$ and $\pi^* \leftarrow \text{NIZK.Prove}(\text{crs}, d^*, w^*)$, $B_1$ perfectly simulates $\text{Hyb}_0$.

If $(\text{crs}, \pi^*)$ consists of $(\text{crs}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda, d^*)$ and $\pi^* \leftarrow \text{Sim}_2(\text{crs}, \text{td}, d^*)$, $B_1$ perfectly simulates $\text{Hyb}_1$.

Thus, if $D$ distinguishes these two games, $B_1$ breaks the computational ZK property of $\Pi_{\text{nizk}}$. We complete the proof.

*Proposition 4.8.* \Pr[H_{\text{Hyb}} = 1] = \Pr[H_{\text{Hyb}} = 1] \leq q \cdot \text{Adv}_{\text{crypt}}^{\text{crypt}}(\lambda) + \text{negl}(\lambda).

*Proof of Proposition 4.8.* We define more hybrid games. Recall $q$ is the total number of key queries.

$\text{Hyb}_0^1$ is the same as $\text{Hyb}_0$ except that

- for $j < k \leq q$, the challenger generates $\text{sk} = iO(D[\text{crs}, \{\text{sk}_i\}, z])$ for the $k$-th key query.
- for $1 \leq k \leq j$, the challenger generates $\text{sk} = iO(D_0[\bar{\text{crs}}, \text{sk}_0])$ for the $k$-th key query.

Clearly, $\text{Hyb}_0^1 = \text{Hyb}_0$ and $\text{Hyb}_0^q = \text{Hyb}_0$.

Let $\text{Invalid}$ be an event that the adversary generates $\text{CT}_{i,0}^* = \text{ABE.Enc}(\text{abe}, \text{pk}_{i,0}, X^*, m_0[i])$, $\text{CT}_{i,1}^* = \text{ABE.Enc}(\text{abe}, \text{pk}_{i,1}, X^*, m_1[i])$, and $\pi^*$ such that $\text{NIZK.Vrfy}(\text{crs}, d^*, \pi^*) = \top$ where $d^* = (\{\text{abe}, \text{pk}_{i,0}, \text{abe}, \text{pk}_{i,1}, \text{CT}_{i,0}^*, \text{CT}_{i,1}^*\})$, $X^*$, $m_0[i] \neq m_1[i]$ for some $i \in [\ell_m]$, and $(\text{crs}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda, d^*)$ such that $d^* \neq d^!$. By the definition of $D$ and $D_0$, their functionalities are equivalent for all inputs as long as $\text{Invalid}$ does not happen. By the statistical simulation-soundness, there is no false statement $d^!$ such that $d^! \neq d^*$ and a proof for $d^!$ can pass the verification except negligible probability.
Note that if \( d^t = d^* \), \( X^t = X^* \) and the decryption algorithm must output \( \perp \) since \( P(X^*) = \perp \) by the admissible property of the ABE security. This means both \( D \) and \( D_0 \) output \( \perp \) for the target ciphertext. That is, Invalid happens with negligible probability by the statistical simulation-soundness. Thus, \( D \) and \( D_0 \) are functionally equivalent except negligible probability.

The difference between \( \text{Hyb}^{j,1}_1 \) and \( \text{Hyb}^{j,1}_2 \) is that the \( j \)-th key query answer is generated by \( D_0 \) instead of \( D \). These are indistinguishable by the IO security. That is, \( | \Pr[\text{Hyb}^{j,1}_1 = 1] - \Pr[\text{Hyb}^{j,1}_2 = 1] | \leq \text{Adv}_{\text{Hyb}^{j,1},\Sigma_{\text{abe}}}^\text{Ind} \). \( \square \)

**Proposition 4.9.** \( | \Pr[\text{Hyb} = 1] - \Pr[\text{Hyb}_3 = 1] | \leq \ell_m \cdot \text{Adv}_{\text{Hyb},\Sigma_{\text{abe}}}^\text{Ind-sel-cpa} \). 

**Proof of Proposition 4.9.** We define more hybrid games. Recall \( \ell_m \) is the length of plaintexts.

\( \text{Hyb}^{0}_2 \): This is the same as \( \text{Hyb}_2 \) except that

- for \( j < i \leq \ell_m \), the challenger generates \( \text{CT}^*_{i,b} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{i,b}, X^*, m[i]) \) for \( b \in \{0,1\} \).
- for \( 1 \leq i \leq j \), the challenger generates \( \text{CT}^*_{i,1-z[i]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{i,1-z[i]}, X^*, 1-m[i]) \) and \( \text{CT}^*_{i,z[i]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{i,z[i]}, m[i]) \).

Clearly, \( \text{Hyb}^{0}_2 = \text{Hyb}_2 \) and \( \text{Hyb}^{\ell_m}_2 = \text{Hyb}_3 \).

The difference between \( \text{Hyb}^{0}_2 \) and \( \text{Hyb}^{\ell_m}_2 \) is the \( j \)-th component of the target ciphertext is valid or invalid. However, in these two games, \( \text{msk} := \{ \{ \text{abe.msks}_{i,z[i]} \} \}_{i \in \{\ell_m\}} \). That is, \( \{ \text{msks}_{1,1-z[j]} \} \) is never revealed to the adversary.

We can construct an adversary \( B_3 \) that breaks IND-sel-CPA security of \( \Sigma_{\text{abe}} \) under key \( \text{abe.pk}_{j,1-z[j]} \) by using the distinguisher \( D \) of these two games. \( D \) first declares the target attribute \( X^* \) and target message \( m^* \). \( B_3 \) passes \( X^* \) to the challenger, receives \( \text{abe.pk} \), and sets \( \text{abe.pk}_{j,1-z[j]} := \text{abe.pk} \). For other public keys (that is, \( \{ \text{abe.pk}_{i,b} \}_{i,b} \setminus \{ \text{abe.pk}_{j,1-z[j]} \} \)), \( B_3 \) generates them by itself. \( B_3 \) sends \( \{ \text{abe.pk}_{i,b} \}_{i,b} \) to the distinguisher. When the distinguisher sends a key query \( P \), \( B_3 \) passes \( P \) to the challenger and receives \( \text{sk}_{j,1-z[j]} \leftarrow \text{ABE.KeyGen}(\text{msk}_{j,1-z[j]}, P) \). For other secret keys for \( P \) (that is, \( \{ \text{sk}_{i,b} \leftarrow \text{ABE.KeyGen}(\text{msk}_{i,b}, P) \}_{i,b} \setminus \{ \text{sk}_{j,1-z[j]} \} \)), \( B_3 \) generates them by itself since it has \( \{ \text{msks}_{i,b} \}_{i,b} \) except \( \text{msks}_{j,1-z[j]} \). Thus, \( B_3 \) can compute \( \text{sk} = iO(D_0[\text{crs}, \text{sk}^*]) \).

At some point, \( B_3 \) sends \( (m^*[j], 1-m^*[j]) \) to the challenger and receives \( \text{CT}^*_{j,1-z[j]} \). For \( (i,b) \in \{\ell_m\} \times \{0,1\} \) \( (j, 1-z[j]) \), \( B_3 \) generates \( \text{CT}^*_{j,z[j]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{j,z[j]}, X^*, m^*[j]) \) and \( \{ \text{CT}_{i,b} \}_{i \in \{\ell_m\} \setminus \{j\}} \) as in \( \text{Hyb}^{j,1}_2 \) and \( \text{Hyb}^{j,1}_3 \). Note that the difference between two games is the \( j \)-th component (and in particular \( (j, 1-z[j]) \) part) of the target ciphertext. Again, \( B_3 \) simulates answers for secret key queries as above. \( B_3 \) outputs whatever \( D \) outputs.

- If \( \text{CT}^*_{j,1-z[j]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{j,1-z[j]}, X^*, m^*[j]) \), \( B_3 \) perfectly simulates \( \text{Hyb}^{j,1}_2 \).
- If \( \text{CT}^*_{j,1-z[j]} \leftarrow \text{ABE.Enc}(\text{abe.pk}_{j,1-z[j]}, X^*, 1-m^*[j]) \), \( B_3 \) perfectly simulates \( \text{Hyb}^{j,1}_2 \).

Thus, if \( D \) distinguishes these two games, \( B_3 \) breaks IND-sel-CPA security of \( \Sigma_{\text{abe}} \). This completes the proof. \( \square \)

**Proposition 4.10.** \( \text{Hyb}_3 = \text{Hyb}_4 \).

**Proof of Proposition 4.10.** This is a conceptual change. The advantage of distinguishing these two games is 0 since we can see that these two games are identical if we set \( z := z^* \oplus m^* \). Note that secret keys do not depend on \( z \) in these games. \( \square \)

4.3 ABE with Certified Deletion from NCABE and SKE with Certified Deletion

In this section, we construct ABE with certified deletion from NCABE and OT-CD secure SKE with certified deletion.
Our ABE with certified deletion scheme. We construct an ABE with certified deletion scheme \( \Sigma_{\text{cd}} = (\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy}) \) with plaintext space \( M \), attribute space \( X \), policy space \( \mathcal{P} \) from an NCABE scheme \( \Sigma_{\text{nce}} = \text{NCE}(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{FakeSetup}, \text{FakeSK}, \text{FakeCT}, \text{Reveal}) \) with plaintext space \( \{0, 1\}^\ell \), attribute space \( X \), policy space \( \mathcal{P} \) and an SKE with certified deletion scheme \( \Sigma_{\text{skecd}} = \text{SKE}(\text{Gen}, \text{Enc}, \text{Dec}, \text{Del}, \text{Vrfy}) \) with plaintext space \( M \) and key space \( \{0, 1\}^\ell \).

Setup(1^\ell):
- Generate \( (\text{nce}.pk, \text{nce}.msk, \text{nce}.aux) \leftarrow \text{NCE.Setup}(1^\ell) \).
- Output \( (\text{pk}, \text{msk}) := (\text{nce}.pk, \text{nce}.msk) \).

KeyGen(msk, P):
- Generate \( \text{nce}.sk_P \leftarrow \text{NCE.KeyGen}(\text{nce}.msk, P) \) and output \( \text{sk}_P := \text{nce}.sk_P \).

Enc(pk, X, m):
- Parse \( \text{pk} = \text{nce}.pk \).
- Generate \( \text{ske}.sk := \text{SKE.Gen}(1^\ell) \).
- Compute \( \text{nce}.CT_X \leftarrow \text{NCE.Enc}(\text{nce}.pk, X, \text{ske}.sk) \) and \( \text{ske}.CT \leftarrow \text{SKE.Enc}(\text{ske}.sk, m) \).
- Output \( \text{CT}_X := (\text{nce}.CT_X, \text{ske}.CT) \) and \( \text{vk} := \text{ske}.sk \).

Dec(sk_P, CT_X):
- Parse \( \text{sk}_P = \text{nce}.sk_P \) and \( \text{CT}_X = (\text{nce}.CT_X, \text{ske}.CT) \).
- Compute \( \text{sk}' \leftarrow \text{NCE.Dec}(\text{nce}.sk_P, \text{nce}.CT_X) \).
- Compute and output \( m' \leftarrow \text{SKE.Dec}(\text{sk}', \text{ske}.CT) \).

Del(CT):
- Parse \( \text{CT}_X = (\text{nce}.CT_X, \text{ske}.CT) \).
- Generate \( \text{ske}.cert := \text{SKE.Del}(\text{ske}.CT) \).
- Output \( \text{cert} := \text{ske}.cert \).

Vrfy(vk, cert):
- Parse \( \text{vk} = \text{ske}.sk \) and \( \text{cert} = \text{ske}.cert \).
- Output \( \text{b} := \text{SKE.Vrfy}(\text{ske}.sk, \text{ske}.cert) \).

Correctness. Correctness easily follows from correctness of \( \Sigma_{\text{skecd}} \) and \( \Sigma_{\text{nce}} \).

**Theorem 4.11.** If \( \Sigma_{\text{nce}} \) is selective-message RNC secure NCABE and \( \Sigma_{\text{skecd}} \) is OT-CD secure, \( \Sigma_{\text{cd}} \) is IND-sel-CPA-CD secure ABE.

**Proof.** We define the following hybrid game \( \text{Hyb}(b) \).

\( \text{Hyb}(b) \): This is the same as \( \text{Exp}_{\Sigma_{\text{cd}}}^{\text{ind-sel-cpa-cd}}(\lambda, b) \) except that following two differences: (1) the challenger generates the target ciphertext as follows. It generates \( \text{ske}.sk \leftarrow \text{SKE.Gen}(1^\ell) \) and \( \text{nce}.aux \leftarrow \text{FakeSetup}(1^\ell, X^*) \), and computes \( \text{nce}.CT_X \leftarrow \text{NCE.Fake}(\text{nce}.pk, \text{nce}.aux, X^*) \) and \( \text{ske}.CT^* \leftarrow \text{SKE.Enc}(\text{ske}.sk, m_b) \). The target ciphertext is \( \text{CT}_X := (\text{nce}.CT^*, \text{ske}.CT^*) \). (2) the challenger generates secret keys as follows. It generates \( \text{nce}.sk_P \leftarrow \text{NCE.FakeSK}(\text{nce}.pk, \text{nce}.aux, P) \) and returns it. (3) The challenger reveals \( \text{msk} \leftarrow \text{Reveal}(\text{nce}.pk, \text{nce}.aux, \text{nce}.CT_X, \text{ske}.sk) \) instead of \( \text{nce}.msk \).

**Proposition 4.12.** \( |\text{Pr}[\text{Exp}_{\Sigma_{\text{cd}}}^{\text{ind-sel-cpa-cd}}(\lambda, b) = 1] - \text{Pr}[\text{Hyb}(b) = 1]| \leq \text{Adv}_{\Sigma_{\text{nce}}, \delta_1}^{\text{ncr-sel2-cpa}}(\lambda) \).
Proof. We construct an adversary $B_1$ that breaks the selective-message RNC security of $\Sigma_{nce}$ by using the distinguisher $D$ for these two games. First, the distinguisher declares the target attribute $X^*$. $B_1$ generates $\text{ske.sk} \leftarrow \text{SKE.Gen}(\lambda)$ and passes $(X^*, \text{ske.sk})$ as the target attribute and message to the challenger, receives $\text{nce.pk}$, and passes $\text{nce.pk}$ to $D$.

When $D$ sends a key query $P$, $B_1$ returns $\bot$ if $P(X^*) = \bot$, otherwise it passes $P$ to the challenger, receives $\text{sk}_P$, and passes it to $D$. When the challenger sends $(\text{nce.}\text{CT}_X^*, \text{nce.}\text{msk}^*)$ to $B_1$, $B_1$ generates $\text{ske.CT} \leftarrow \text{SKE.Enc(}\text{ske.sk}, m_0)$ and sends $(\text{nce.}\text{CT}_X^*, \text{ske.CT})$ to $D$ as the challenge ciphertext. At some point, $D$ outputs $\text{cert}$. If $\text{SKE.Vrfy(}\text{cert}) = \bot$, $B_1$ sends $\text{msk}^*$ to $D$.

- If $(\text{nce.}\text{CT}_X^*, \text{msk}^*) = (\text{NCE.}\text{Enc(}\text{nce.pk}, X^*, \text{ske.sk}), \text{nce.}\text{msk})$ and $\text{sk}_P = \text{NCE.KeyGen(}\text{nce.}\text{msk}, P)$, $B_1$ perfectly simulates $\text{Exp}_{\Sigma_{nce}, A}^{\text{ind-select-CPA-final}}(\lambda, b)$.

- If $(\text{nce.}\text{CT}_X^*, \text{msk}^*) = (\text{NCE.}\text{FakeCT(nce.pk, nce.aux, X^*)}, \text{NCE.Reveal(nce.pk, nce.aux, nce.}\text{CT}_X^*, \text{ske.sk}))$ and $\text{sk}_P = \text{NCE.}\text{FakeSK(}\text{nce.pk}, \text{nce.aux}, P)$, $B_1$ perfectly simulates $\text{Hyb}(b)$.

Thus, if $D$ distinguishes the two games, $B_1$ breaks the selective-message RNC security. This completes the proof.

Proposition 4.13. $|\Pr[\text{Hyb}(0) = 1] - \Pr[\text{Hyb}(1) = 1]| \leq \text{Adv}_{\Sigma_{nce}, \text{Hyb}}(\lambda)$.

Proof. We construct an adversary $B_2$ that breaks the OT-CD security of $\Sigma_{nce}$ by using the distinguisher $D$ for these two games. First, $D$ declares the target attribute $X^*$. $B_2$ generates $(\text{nce.pk}, \text{nce.aux}) \leftarrow \text{NCE.Setup}(\lambda, X^*)$ and sends $\text{nce.pk}$ to $D$. When $D$ sends a key query $P$, $B_2$ generates $\text{sk}_P \leftarrow \text{NCE.}\text{FakeSK(}\text{nce.pk}, \text{nce.aux}, P)$ and returns it to $D$. When $D$ sends $(m_0, m_1)$, $B_2$ sends $(m_0, m_1)$ to the challenger of OT-CD SKE, receives $\text{sk.e.}\text{CT}^*$ and generates $\text{nce.}\text{CT}_X^* \leftarrow \text{NCE.}\text{FakeCT(nce.pk, nce.aux, X^*)}$. $B_2$ sends $(\text{nce.}\text{CT}_X^*, \text{sk.e.}\text{CT}^*)$ to $D$ as the challenge ciphertext. At some point, $D$ outputs $\text{cert}$. $B_2$ passes $\text{cert}$ to the challenger of OT-CD SKE. If the challenger returns $\text{ske.sk}$, $B_2$ generates $\text{msk} \leftarrow \text{NCE.}\text{Reveal(nce.pk, nce.msk, nce.aux, nce.}\text{CT}_X^*, \text{ske.sk})$ and sends $\text{msk}$ to $D$.

- If $\text{sk.e.}\text{CT}^* = \text{SKE.Enc(}\text{ske.sk}, m_0)$, $B_2$ perfectly simulates $\text{Hyb}(0)$.

- If $\text{sk.e.}\text{CT}^* = \text{SKE.Enc(}\text{ske.sk}, m_1)$, $B_2$ perfectly simulates $\text{Hyb}(1)$.

Thus, if $D$ distinguishes the two games, we can break the OT-CD security. This completes the proof.

By Propositions 4.12 and 4.13, we immediately obtain Theorem 3.4.

Summary of this section. Since IO and OWFs imply computational NIZK proof for $\text{NP}$ [BP15] and IND-select-CPA secure ABE with $\mathcal{P} = \text{P/poly}$, we immediately obtain the following corollary by using Theorems 2.11, 2.18, 4.6 and 4.11.

Corollary 4.14. If there exist secure IO for $\text{P/poly}$ against QPT adversaries, there exists ABE with certified deletion with policy space $\text{P/poly}$.

Note that we can instantiate all building blocks except NCABE without IO (that is, can construct from the LWE assumption against QPT adversaries).

References


