CCA-Secure PRE Scheme without Random Oracles

Jun Shao[1], Zhenfu Cao[2], and Peng Liu[1]

[1] College of Information Sciences and Technology
Pennsylvania State University
[2] Department of Computer Science and Engineering
Shanghai Jiao Tong University
`chn.junshao@gmail.com, zfcao@cs.sjtu.edu.cn, pliu@ist.psu.edu`

Abstract. In a proxy re-encryption scheme, a semi-trusted proxy can transform a ciphertext under Alice's public key into another ciphertext that Bob can decrypt. However, the proxy cannot access the plaintext. Due to its transformation property, proxy re-encryption can be used in many applications, such as encrypted email forwarding. In this paper, by using the techniques of Canetti-Hohenberger and Kurosawa-Desmedt, we propose a new single-use unidirectional proxy re-encryption scheme. Our proposal is secure against chosen ciphertext attack (CCA) and collusion attack in the *standard model*.

Keywords: CCA Security, Collusion Resistance, Unidirectional Proxy Re-encryption, Standard Model

1 Introduction

Proxy re-encryption (PRE), introduced by Blaze, Bleumer and Strauss at EUROCRYPT 1998 [4], allows a semi-trusted proxy, with some additional information (a.k.a., re-encryption key), to transform a ciphertext under Alice's public key into a new ciphertext under Bob's public key on the same message. However, the proxy cannot learn any information about the messages encrypted under the public key of either Alice or Bob.

Generally speaking, there are two main methods to classify PRE schemes. One method is according to the direction of transformation. If the re-encryption key allows the proxy to transform from Alice to Bob, and vice versa, the PRE scheme is *bidirectional*; otherwise, it is *unidirectional*. The other method is according to the times of transformation. If the ciphertext can be transformed from Alice to Bob, and then from Bob to Charlie, and so on, the PRE scheme is *multi-use*; otherwise, it is *single-use*.

Due to its specific transformation property, PRE can be used in many applications, including simplification of key distribution [4], key escrow [16], distributed file systems [2,3], security in publish/subscribe systems [19], multicast [10], secure certified email mailing lists [20,18], interoperable architecture of DRM [25], access control [26], and privacy for public transportation [15]. We refer the reader to [1] for the full list.

Since the concept of PRE was proposed, many PRE schemes have been presented. The first (bidirectional) PRE scheme is proposed by Blaze *et al.* [4] based on ElGamal public key encryption [12]. However, their scheme suffers from collusion attacks, i.e., Alice (Bob) can collude with the proxy to reveal Bob's (Alice's) secret key. Furthermore, no unidirectional PRE scheme was proposed in [4]. Later, Jakobsson [17], and Zhou *et al.* [30] gave a partial solution to collusion resistance by proposing a quorum-based protocol where the proxy is divided into sub-components.

Based on the key sharing technique, Ivan and Dodis [16] proposed a generic construction for unidirectional PRE, where the delegator's (Alice's) secret key is divided into two parts, one is sent to the proxy, and the other is sent to the delegatee (Bob). Although this generic construction gives a method to construct unidirectional PRE, it has two disadvantages. (1) Besides his own secret key, the delegatee (Bob) has to store an additional secret to decrypt re-encrypted ciphertext for every delegator. (2) Though the delegator (Alice) cannot collude with the proxy to reveal the delegatee's (Bob's) secret key, the delegatee (Bob) can collude with the proxy to reveal the delegator's (Alice's) secret key. To remove the above two disadvantages, Ateniese *et al.* [2,3] proposed several new PRE schemes based on pairings.

Nevertheless, the above PRE schemes achieve at most chosen plaintext attack (CPA) security [9,23]. The first CCA-secure (bidirectional) PRE scheme in the standard model was proposed by Canetti and Hohenberger [9]. Their result is very nice. To achieve CCA security, Canetti and Hohenberger applied the Canetti, Halevi and Katz paradigm [8] (which is for transforming any selective-identity, CPA-secure ID-based encryption scheme into a CCA-secure encryption scheme) with a little modification. However, like previous bidirectional PRE schemes, their scheme suffers from collusion attacks. Furthermore, they didn't propose any CCA-secure unidirectional PRE scheme in the standard model, but left it as an open problem[1]. Note that unidirectional PRE is more powerful than bidirectional one, since a bidirectional scheme can always be implemented by a unidirectional one. Based on Canetti-Hohenberger technique [9], Libert and Vergnaud [22] proposed a new unidirectional PRE scheme which is replayable chosen ciphertext attack (RCCA) secure and collusion resistant in the standard model. Note that RCCA security is weaker than CCA security, since it disallows the adversary to query the decryption oracle with the ciphertext whose corresponding message is one of the challenge messages in the RCCA security model [22], while only the derivatives[2] of the challenge ciphertext are disallowed in the CCA security model [9]. See the details in Remark 2 (Section 2).

Recently, Shao and Cao [23] proposed the first CCA-secure and collusion resistant unidirectional PRE scheme by using public key encryption with double trapdoors [7] and Fijisaki-Okamoto technique [13]. Weng *et al.* [28] proposed a more efficient CCA-secure and collusion resistant unidirectional PRE scheme without pairings. However, these two schemes are only proven secure in the random oracle model. Shao *et al.* [24] proposed a generic construction of CCA-secure and collusion resistant unidirectional PRE based on CCA-secure ID-based threshold encryption. However, they worked in a weaker security model than our security model in Section 2.3. In their CCA security model, the adversary cannot get the re-encryption keys from uncorrupted users (not the challenge one) to corrupted users.

1.1 Intuition behind Our Construction

In this paper, by using the techniques of Canetti-Hohenberger [9] and Kurosawa-Desmedt [21], we propose a new single-use unidirectional PRE scheme, which is CCA-secure and collusion resistant in the standard model.

There are several methods of the re-encryption key's generation to obtain the collusion resistance, including the AFGH method [2,3] ($g^{y/x}$, where x and y are the private keys of the delegator and the delegatee, respectively), double trapdoor encryption [23], identity-based encryption [27,29,24]. In this paper, we adopt the AFGH method to obtain the collusion resistance.

Compared to the collusion resistance, we need more techniques and more sophisticated design to obtain the CCA security. The key of designing a CCA-secure, single-use, unidirectional PRE scheme is to provide validity check for original ciphertexts and re-encrypted ciphertexts [9,22,23].

The validity check for original ciphertexts guarantees that the original ciphertexts are indeed computed by the encryptor. In this paper, we use the technique of Canetti-Hohenberger [9] to provide public verifiability of the original ciphertext. Actually, the encryption algorithm of our proposal is almost the same as that in [22], except that we use the technique of Kurosawa-Desmedt [21]. However, this modification helps us to improve the security from the RCCA security up to the CCA security, in particular, it provides us the validity check for re-encrypted ciphertexts.

The validity check for re-encrypted ciphertexts guarantees that re-encrypted ciphertexts are not modified after the proxy sends them out. At the first glance, this verifiability should be done by the proxy, since the re-encrypted ciphertext is different from the original ciphertext. However, it is very difficult to use the re-encryption key $g^{y/x}$ to obtain the verifiability (at least with the current techniques). We observed that if the values computed by the proxy in the re-encryption algorithm have some "fixed" relationships with the original ciphertext, then we can check the validity of the re-encrypted ciphertexts via checking the validity of the corresponding message. "Fixed" means that for a re-encryption key and an original ciphertext pair, some parts of the re-encrypted ciphertext are constant, no matter what the random value used during the re-encryption is. In this paper, we use the technique of Kurosawa-Desmedt [21] to provide the "fixed" relationships. In particular, the obtained message in the decryption algorithm can be verified by the Kurosawa-Desmedt technique.

However, by only using the techniques of Canetti-Hohenberger and Kurosawa-Desmedt, we still cannot obtain the CCA security. Note that the Canetti-Hohenberger technique can only be used to answer decryption queries for the original ciphertext, and the Kurosawa-Desmedt cannot be used to answer decryption queries. To simulate the decryption oracle correctly, the re-encrypted ciphertext should contain the plaintext ciphertext. Nevertheless, the re-encrypted ciphertext cannot explicitly contain the original ciphertext according to the CCA security. To solve

[1] Canetti and Hohenberger [9] proposed four open problems on proxy re-encryption, such that building (1) unidirectional CCA-secure schemes in the standard model, (2) multi-hop, unidirectional schemes, (3) unidirectional or CCA-secure scheme without bilinear groups, (4) secure obfuscations of CCA-secure re-encryption or other key translation schemes.

[2] See the definition in Section 2.3.

this conflict, we let every user have *two* public keys but *only one* secret key, i.e., $pk^{(1)} = (g^x, g)$ and $pk^{(2)} = (g^x, g')$. For example, Alice has the public key (g^x, g, g'). (g^x, g) is used by the encryptor to compute the original ciphertexts of Alice, and (g^x, g') is used by the proxy to *encrypt* the original ciphertexts of Alice's delegator.

Combining the above ideas, we obtain a new single-use unidirectional PRE scheme, which is CCA-secure and collusion resistant in the standard model.

In the rest of this paper, we first review the definitions related to our proposal, including the definitions for one-time signature, one-time symmetric key encryption, single-use unidirectional PRE, bilinear maps, and the underlying assumptions. After that, we present our proposal and give the security proof in the standard model. Finally, the conclusion is given.

2 Preliminaries

In this section, we review some basic knowledge we use in this paper.

2.1 One-Time Symmetric-Key Encryption [11,21]

Definition 1. *A one-time symmetric-key encryption scheme* `SKE` *contains three algorithms* `SKE.KeyGen`*,* `SKE.Enc` *and* `SKE.Dec`*.*

– *`SKE.KeyGen`$(1^\lambda) \rightarrow k$. On input the security parameter λ, the key generation algorithm `SKE.KeyGen` outputs a key k.*
– *`SKE.Enc`$(k, m) \rightarrow C$: On input a key k and a message m, the encryption algorithm `SKE.Enc` outputs a ciphertext C.*
– *`SKE.Dec`$(k, C) \rightarrow m$: On input a key k and a ciphertext C, the decryption algorithm `SKE.Dec` outputs a message m or the special symbol $\perp$.*

Correctness. The correctness property is that for any message m in the message space and any key $k \leftarrow$ `SKE.KeyGen`(1^λ), the following condition must hold: `SKE.Dec`$(k,$`SKE.Enc`$(k, m)) = m$.

Chosen Ciphertext Security for One-Time Symmetric-Key Encryption. The CCA security for SKE is defined by the following chosen-ciphertext attack game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup: $\mathcal{C}$ setups the system parameters.
$\mathcal{O}_{enc}$: $\mathcal{A}$ outputs two equal length plaintexts m_0, m_1 from the message space. $\mathcal{C}$ generates a random key k along with a random bit $\mathbf{b}$, and encrypts the message $m_{\mathbf{b}}$ using the key k to get a ciphertext C^*. At last, $\mathcal{C}$ sends C^* to $\mathcal{A}$ as the challenge ciphertext. This oracle can be queried only once.
$\mathcal{O}_{dec}$: $\mathcal{A}$ outputs a ciphertext $C \neq C^*$, and $\mathcal{C}$ returns the corresponding message under k to $\mathcal{A}$. This oracle can be queried for any number of times.
Guess: Finally, $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0, 1\}$ and wins the game if $\mathbf{b} = \mathbf{b}'$.

The advantage $\mathbf{Adv}_{\mathtt{SKE}}^{\mathtt{CCA}\text{-}\mathcal{A}}(\lambda)$ is defined as $|\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. The scheme SKE is said to be secure against chosen-ciphertext attacks if for all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}_{\mathtt{SKE}}^{\mathtt{CCA}\text{-}\mathcal{A}}(\lambda)$ is negligible.

2.2 One-Time Signature Scheme [8]

Definition 2. *A one-time signature scheme* `SIG` *contains three algorithms* `SIG`$.\mathcal{G}$*,* `SIG`$.\mathcal{S}$ *and* `SIG`$.\mathcal{V}$*.*

– *`SIG`$.\mathcal{G}(1^k) \rightarrow (svk, ssk)$. On input the security parameter k, the key generation algorithm `SIG`$.\mathcal{G}$ outputs a key pair (svk, ssk).*
– *`SIG`$.\mathcal{S}(ssk, m) \rightarrow S$: On input a signing key ssk and a message m, the signing algorithm `SIG`$.\mathcal{S}$ outputs a signature S on m.*
– *`SIG`$.\mathcal{V}(svk, m, S) \rightarrow 1$ or 0: On input a verifying key svk, a message m and a signature S, the verification algorithm `SIG`$.\mathcal{V}$ outputs 1 if S is a signature of m under svk; otherwise, it outputs 0.*

Correctness. The correctness property is that for any message $m \neq m'$ in the message space and any key pair $(svk, ssk) \leftarrow \mathtt{SIG}.\mathcal{G}(1^k)$, the following conditions must hold:

$$\mathtt{SIG}.\mathcal{V}(svk, m, \mathtt{SIG}.\mathcal{S}(ssk, m)) = 1, \quad \text{and} \quad \mathtt{SIG}.\mathcal{V}(svk, m, \mathtt{SIG}.\mathcal{S}(ssk, m')) = 0,$$

Strong Unforgeability for One-Time Signature Scheme. The strong unforgeability security for SIG is defined by the following chosen-message attack game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

Setup: $\mathcal{C}$ setups the system parameters.

$\mathcal{O}_{sig}$: $\mathcal{A}$ outputs a m from the message space. $\mathcal{C}$ generates a random key pair (svk, ssk), and signs the message m using the singing key ssk to get a signature S^*. At last, $\mathcal{C}$ returns (svk, S^*) to $\mathcal{A}$. This oracle can be queried only once.

Forge: Finally, $\mathcal{A}$ outputs a signature S' of a message m' under svk. If any of following conditions holds, $\mathcal{A}$ wins the game.

- $m \neq m'$ and $\mathtt{SIG}.\mathcal{V}(svk, m', S') = 1$.
- $m = m'$, $S \neq S'$, and $\mathtt{SIG}.\mathcal{V}(svk, m', S') = 1$.

The advantage $\mathbf{Adv}^{\mathtt{SU}\text{-}\mathcal{A}}_{\mathtt{SIG}}(k)$ is defined as $\Pr[\mathcal{A} \text{ wins}]$. The scheme SIG is said to be strongly unforgeable under chosen-message attacks if for all efficient adversaries $\mathcal{A}$, the advantage $\mathbf{Adv}^{\mathtt{SU}\text{-}\mathcal{A}}_{\mathtt{SIG}}(k)$ is negligible.

2.3 Definitions for Single-Use Unidirectional PRE

The similar definitions can be found in [2,3,14,9,23].

Definition 3 (Single-Use Unidirectional `PRE`). *A single-use unidirectional proxy re-encryption scheme* `PRE` *is a tuple of PPT algorithms (*`PRE.KeyGen`*,* `PRE.ReKeyGen`*,* `PRE.Enc`*,* `PRE.ReEnc`*,* `PRE.Dec`*):*

- `PRE.KeyGen`$(1^k) \rightarrow (pk, sk)$. *On input the security parameter* 1^k*, the key generation algorithm* `PRE.KeyGen` *outputs a public key* pk *and a secret key* sk*.*
- `PRE.ReKeyGen`$(sk_1, pk_2) \rightarrow rk_{1,2}$. *On input a secret key* sk_1 *and a public key* pk_2*, the re-encryption key generation algorithm* `PRE.ReKeyGen` *outputs a unidirectional re-encryption key* $rk_{1,2}$*.*
- `PRE.Enc`$(pk, m) \rightarrow C$. *On input a public key* pk *and a message* m *in the message space, the encryption algorithm* `PRE.Enc` *outputs a ciphertext* C*.*
- `PRE.ReEnc`$(rk_{1,2}, C_1) \rightarrow C_2$. *On input a re-encryption key* $rk_{1,2}$ *and a ciphertext* C_1*, the re-encryption algorithm* `PRE.ReEnc` *outputs a re-encrypted ciphertext* C_2 *or a special symbol* $\perp$*.*
- `PRE.Dec`$(sk, C) \rightarrow m$. *On input a secret key* sk *and a ciphertext* C*, the decryption algorithm* `PRE.Dec` *outputs a message* m *in the message space or a special symbol* $\perp$*.*

Correctness. The correctness property has two requirements. For any message m in the message space and any key pairs $(pk, sk), (pk', sk') \leftarrow \mathtt{PRE.KeyGen}(1^k)$. Then the following two conditions must hold:

$$\mathtt{PRE.Dec}(sk, \mathtt{PRE.Enc}(pk, m)) = m,$$
$$\mathtt{PRE.Dec}(sk', \mathtt{PRE.ReEnc}(\mathtt{PRE.ReKeyGen}(sk, pk'), C)) = m,$$

where C is the ciphertext for message m under pk from algorithm `PRE.Enc`.

Remark 1 (Two types of ciphertexts). In all existing single-use unidirectional proxy re-encryption schemes, there are two types of ciphertexts. One is the *first-level* ciphertext, which could be generated from `PRE.ReEnc` (or `PRE.Enc`); the other is the *second-level* ciphertext, which is generated only from `PRE.Enc`.

Chosen Ciphertext Security for Single-Use Unidirectional Proxy Re-Encryption. We say that a single-use unidirectional proxy re-encryption scheme PRE is semantically secure against an adaptive chosen ciphertext attack if no polynomial bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following Uni-PRE-CCA game. Note that we work in the static corruption model, where the adversary should decide the corrupted users before the game starts. Since there are two types of ciphertexts, we have two situations in some of the following oralces.

Setup: The challenger sets up the system parameters.

Phase 1: The adversary $\mathcal{A}$ issues queries $q_1, \cdots, q_{n_1}$ where query q_i is one of:

– *Public key generation oracle* $\mathcal{O}_{pk}$: On input an index i,[3] the challenger takes a security parameter k, and responds by running algorithm $\texttt{PRE.KeyGen}(1^k)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in the table T_K.
– *Secret key generation oracle* $\mathcal{O}_{sk}$: On input pk_i by $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, if pk_i is corrupted, the challenger searches pk_i in the table T_K and returns sk_i; otherwise, the challenger returns $\perp$.
– *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input (pk_i, pk_j) by $\mathcal{A}$, where pk_i, pk_j are from $\mathcal{O}_{pk}$, the challenger returns the re-encryption key $rk_{i,j} = \texttt{PRE.ReKeyGen}(sk_i, pk_j)$, where sk_i is the secret key corresponding to pk_i.
– *Re-encryption oracle* $\mathcal{O}_{re}$: On input (pk_i, pk_j, C) by $\mathcal{A}$, where pk_i, pk_j are from $\mathcal{O}_{pk}$, the challenger returns the re-encrypted ciphertext $C' = \texttt{PRE.ReEnc}(\texttt{PRE.ReKeyGen}(sk_i, pk_j), C)$, where sk is the secret key corresponding to pk.
– *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk_i, C_i), where pk_i is from $\mathcal{O}_{pk}$, the challenger returns $\texttt{PRE.Dec}(sk_i, C_i)$, where sk_i is the secret key corresponding to pk_i.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: There are two cases, and the adversary can perform only one case for only once in this phase.

– *the first-level challenge ciphertext*: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and two public key pk, pk^* on which it wishes to challenge. There are two constraints on the public key pk^*: (i) it is from $\mathcal{O}_{pk}$; (ii) it is uncorrupted. The challenger picks a random bit $\mathbf{b} \in \{0,1\}$ and sets $C^* = \texttt{PRE.ReEnc}(rk, \texttt{PRE.Enc}(pk, m_{\mathbf{b}}))$, where rk is a re-encryption key from pk to pk^*. It sends C^* as the challenge to $\mathcal{A}$.
– *the second-level challenge ciphertext*: Once the adversary $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0, m_1 from the message space, and a public key pk^* on which it wishes to challenge. There are three constraints on the public key pk^*, (i) it is from $\mathcal{O}_{pk}$; (ii) it is uncorrupted; (iii) if $(pk^*, \bigstar)$ did appear in any query to $\mathcal{O}_{rk}$, then $\bigstar$ is uncorrupted. The challenger picks a random bit $\mathbf{b} \in \{0,1\}$ and sets $C^* = \texttt{PRE.Enc}(pk^*, m_{\mathbf{b}})$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ issues more queries $q_{n_1+1}, \cdots, q_n$ where query q_i is one of:

– $\mathcal{O}_{pk}$, $\mathcal{O}_{sk}$: The challenger responds as in Phase 1.
– $\mathcal{O}_{rk}$:
  - *the first-level ciphertext security*: it is the same as that in Phase 1.
  - *the second-level ciphertext security*: if the following requirements are all satisfied, the challenger responds as in Phase 1; otherwise, the challenger outputs $\perp$.
    * pk_i and pk_j are from $\mathcal{O}_{pk}$;
    * if $pk_i = pk^*$, then pk_j is uncorrupted.
– $\mathcal{O}_{re}$: On input (pk_i, pk_j, C_i) by $\mathcal{A}$, if the following requirements are all satisfied, the challenger responds as in Phase 1; otherwise, the challenger outputs $\perp$.
  - pk_i and pk_j are from $\mathcal{O}_{pk}$;
  - if (pk_i, C_i) is a derivative of (pk^*, C^*), then pk' is uncorrupted.
– $\mathcal{O}_{dec}$: On input (pk_i, C_i), if the following requirements are all satisfied, the challenger responds the same as in Phase 1; otherwise, the challenger outputs $\perp$.
  - pk_i is from $\mathcal{O}_{pk}$;

[3] This index is just used to distinguish the different public keys.

- (pk_i, C_i) is not a derivative of (pk^*, C^*).

These queries may be also asked adaptively.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $\mathbf{b}' \in \{0,1\}$ and wins the game if $\mathbf{b} = \mathbf{b}'$.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRE-CCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking `PRE` as the following function of the security parameter k:

$$\mathbf{Adv}_{\mathtt{PRE}}^{\mathtt{CCA}\text{-}\mathcal{A}}(k) = |\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|.$$

Using the Uni-PRE-CCA game, we can define chosen ciphertext security for unidirectional proxy re-encryption schemes.

Remark 2. Derivatives of (pk^*, C^*) for the CCA security are defined as follows [9]:

1. (pk^*, C^*) is a derivative of itself.
2. If (pk, C) is a derivative of (pk^*, C^*) and (pk', C') is a derivative of (pk, C), then (pk', C') is a derivative of (pk^*, C^*).
3. If $\mathcal{A}$ has queried $\mathcal{O}_{re}$ on input (pk, pk', C) and obtained (pk', C'), then (pk', C') is a derivative of (pk, C).
4. If $\mathcal{A}$ has queried $\mathcal{O}_{rk}$ on input (pk, pk'), and $C' = \mathtt{PRE.ReEnc}(\mathcal{O}_{re}(pk, pk'), C)$, then (pk', C') is a derivative of (pk, C).

From the above, we know that if a re-encrypted ciphertext is not obtained *directly* by `PRE.ReEnc` or $\mathcal{O}_{re}$, then this ciphertext cannot be a derivative of any ciphertext. Hence, if $C'' = F(C')$, where F is a transformation function which makes $C'' \neq C'$ and $\mathtt{Dec}(C'', pk) = \mathtt{Dec}(C', pk)$, and $C' = \mathcal{O}_{re}(pk^*, pk', C^*)$, then C'' is not a derivative of (pk^*, C^*).

However, according to the definition of the derivative of (pk^*, C^*) for the RCCA security [22], the above C'' is a derivative of (pk^*, C^*). Because the derivative of (pk^*, C^*) in [22] is defined as: if the message of a re-encrypted ciphertext is one of the challenge messages, then this re-encrypted ciphertext is a derivative of (pk^*, C^*).

Definition 4 (Uni-PRE-CCA security). *We say that the unidirectional proxy re-encryption scheme* `PRE` *is semantically secure against an adaptive chosen ciphertext attack if for any polynomial time Uni-PRE-CCA adversary $\mathcal{A}$ the function $\mathbf{Adv}_{\mathtt{PRE}}^{\mathtt{CCA}\text{-}\mathcal{A}}(k)$ is negligible. As shorthand, we say that* `PRE` *is Uni-PRE-CCA-secure.*

Collusion Resistance for Single-Use Unidirectional Proxy Re-Encryption. We say that a unidirectional proxy re-encryption scheme `PRE` is collusion resistant under an adaptive chosen ciphertext attack if no polynomial bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following Uni-PRE-CR game. Here, we also work in the static corruption model.

Setup: The challenger sets up the system parameters.
Find: Almost the same as Phase 1 in Uni-PRE-CCA game, except that there is no re-encryption oracle in this game, since the adversary can get every re-encryption key.
Output: Finally, the adversary $\mathcal{A}$ outputs private key sk, and wins the game if the corresponding public key pk is uncorrupted.

We refer to such an adversary $\mathcal{A}$ as a Uni-PRE-CR adversary. We define adversary $\mathcal{A}$'s advantage in attacking `PRE` as the following function of the security parameter k:

$$\mathbf{Adv}_{\mathtt{PRE}}^{\mathtt{CR}\text{-}\mathcal{A}}(k) = \Pr[\mathcal{A} \text{ wins}]$$

Using the Uni-PRE-CR game, we can define collusion resistance under chosen ciphertext attack for single-use unidirectional proxy re-encryption schemes.

Definition 5 (Uni-PRE-CR security). *We say that the single-use unidirectional proxy re-encryption scheme* `PRE` *is collusion resistant under an adaptive chosen ciphertext attack if for any polynomial time Uni-PRE-CR adversary $\mathcal{A}$ the function $Adv_{\mathtt{PRE}}^{\mathtt{CR}\text{-}\mathcal{A}}(k)$ is negligible. As shorthand, we say that* `PRE` *is Uni-PRE-CR secure.*

Remark 3. As mentioned in [22], the security of collusion resistance is implied by the *first-level* ciphertext security.

2.4 Bilinear Groups

In this subsection, we briefly review the definitions about bilinear maps and bilinear map groups, which follow those in [5,6].

1. $\mathbb{G}$ and $\mathbb{G}_T$ are two (multiplicative) cyclic groups of prime order q;
2. g is a generator of $\mathbb{G}$;
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups as above. An *admissible bilinear map* is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

1. *Bilinearity*: For all $P, Q, R \in \mathbb{G}$, $e(P \cdot Q, R) = e(P, R) \cdot e(Q, R)$ and $e(P, Q \cdot R) = e(P, Q) \cdot e(P, R)$.
2. *Non-degeneracy*: If $e(P, Q) = 1$ for all $Q \in \mathbb{G}$, then $P = \mathcal{O}$, where $\mathcal{O}$ is a point at infinity.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists a group $\mathbb{G}_T$ and an efficiently computable bilinear map as above. We denote `BSetup` as an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $q \in \Theta(2^k)$.

2.5 Complexity Assumptions

The security of the schemes proposed in this paper are based on the 3-Quotient Decision Bilinear Diffie-Hellman assumption (3-QDBDH), the extended 3-Quotient Decision Bilinear Diffie-Hellman assumption (e3-QDBDH), and the Extended Discrete Logarithm assumption (EDL).

3-QDBDH Problem. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow$`BSetup`$(1^k)$. The 3-QDBDH problem is as follows: Given $(g, g^a, g^{a^2}, g^{a^3}, g^b, Q)$ for some $a, b \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_T$, decide whether $Q = e(g, g)^{b/a}$. An algorithm $\mathcal{A}$ has advantage ε in solving 3-QDBDH problem if

$$|\Pr[\mathcal{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, e(g, g)^{b/a}) = 0] - \Pr[\mathcal{A}(g, g^a, g^{a^2}, g^{a^3}, g^b, Q) = 0]| \leq \varepsilon$$

where the probability is over the random choice of a, b in $\mathbb{Z}_q$, the random choice of Q in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

Definition 6 (3-QDBDH Assumption). *We say that the ε-3-QDBDH assumption holds if no PPT algorithm has advantage at least ε in solving the 3-QDBDH problem.*

The 3-QDBDH assumption is used in [22] to build a RCCA-secure and collusion-resistant PRE scheme.

e3-QDBDH Problem. Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow$`BSetup`$(1^k)$. The e3-QDBDH problem is as follows: Given $(g, g^{1/a}, g^a, g^{a^2}, g^{a^3}, g^b, g^c, Q)$ for some $a, b, c \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_T$, decide whether $Q = e(g, g)^{bc/a}$. An algorithm $\mathcal{A}$ has advantage ε in solving 3-QDBDH problem if

$$|\Pr[\mathcal{A}(g, g^{1/a}, g^a, g^{a^2}, g^{a^3}, g^b, g^c, e(g, g)^{bc/a}) = 0] - \Pr[\mathcal{A}(g, g^{1/a}, g^a, g^{a^2}, g^{a^3}, g^b, g^c, Q) = 0]| \leq \varepsilon$$

where the probability is over the random choice of a, b in $\mathbb{Z}_q$, the random choice of Q in $\mathbb{G}_T$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

Definition 7 (e3-QDBDH Assumption). *We say that the ε-e3-QDBDH assumption holds if no PPT algorithm has advantage at least ε in solving the e3-QDBDH problem.*

EDL Problem. Let $\langle g \rangle = \mathbb{G}$ is a finite cyclic group with prime order q, and g is one of its generator. The EDL problem is as follows: Given $(g, g^a, g^{1/a})$ for some $a \in \mathbb{Z}_q$, compute a. An algorithm $\mathcal{A}$ has advantage ε in solving EDL problem if

$$\Pr[\mathcal{A}(g, g^a, g^{1/a}) = a] \leq \varepsilon$$

where the probability is over the random choice of a in $\mathbb{Z}_q$, the random choice of $g \in \mathbb{G}^*$, and the random bits of $\mathcal{A}$.

Definition 8 (EDL Assumption). *We say that the ε-EDL assumption holds if no PPT algorithm has advantage at least ε in solving the EDL problem.*

It is easy to see that the EDL problem is easier than the Discrete Logarithm problem (DL), and is equal to 2-DL problem[4], which is used in [2,3] to build a collusion resistant PRE scheme.

3 CCA-Secure, Collusion-Resistant, Single-Use, Unidirectional PRE

Notations and Configuration. Let 1^k be the security parameter and $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \texttt{BSetup}(1^k)$, g' and h be random numbers in $\mathbb{G}$, $\texttt{SIG}$ be a strongly unforgeable one-time signature scheme, and $\texttt{SKE}$ be CCA-secure one-time symmetric key encryption. Let $F(y) \overset{def}{=} g_2^y \cdot g_3$,[5] and $H : \mathbb{G}_T \rightarrow \{0,1\}^{k_1}$, where g_2 and g_3 are random elements in $\mathbb{G}$, k_1 is the bit-length of the underlying one-time symmetric encryption's key, and we require that H is uniformly distributed over $\{0,1\}^{k_1}$ if the input is uniformly distributed over $\mathbb{G}_T$. We can use any concrete hash function, such as SHA-1 [21]. Define the algorithm *Check* on input a ciphertext tuple (A, B, C, D, E, S) and a key pk as follows:

1. Run $\texttt{SIG}.\mathcal{V}(A, (B, C, D, E), S)$ to verify signature S on message (C, D, E) with respect to key A.
2. Check that $e(F(A), B) = e(C, pk)$.
3. Check that $e(F(A), D) = e(C, h)$.
4. If any of these checks fails, output 0; else output 1.

The system parameters of our proposal are $(q, g, g', h, \mathbb{G}, \mathbb{G}_T, e, \texttt{SIG}, \texttt{SKE}, F, H)$. The algorithms are as follows.

$\texttt{PRE.KeyGen}$: On input 1^k, select random $x \in \mathbb{Z}_q$. Set $pk = g^x$ and $sk = x$.

$\texttt{PRE.ReKeyGen}$: On input a public key pk_Y and a secret key $sk_X = x$, output the unidirectional re-encryption key $rk_{X,Y} = (rk^{(1)}_{X,Y}, rk^{(2)}_{X,Y}, rk^{(3)}_{X,Y}) = ((pk_Y)^{1/x}, pk_X, pk_Y) = (g^{y/x}, g^x, g^y)$.

$\texttt{PRE.Enc}$: On input pk and a message $m \in \{0,1\}^n$, do:

1. Select a one-time signature key pair as $\texttt{SIG}.\mathcal{G}(1^k) \rightarrow (svk, ssk)$. Set $A = svk$.
2. Select a random number $r \in \mathbb{Z}_q$ and compute

$$B = pk^r, \ C = F(A)^r, \ D = h^r, \ v = e(g,g)^r, \ k = H(v).$$

3. Run the encrypting algorithm $\texttt{SKE.Enc}(k, m)$, where the plaintext is m, and denote the ciphertext E.
4. Run the signing algorithm $\texttt{SIG}.\mathcal{S}(ssk, (B, C, D, E))$, where the message to sign is the tuple (B, C, D, E), and denote the signature S.
5. Output the ciphertext (A, B, C, D, E, S).

$\texttt{PRE.ReEnc}$: On input a re-encryption key $rk_{X,Y}$ and a ciphertext $K = (A, B, C, D, E, S)$ under key pk_X, if $Check(K, pk_X) = 0$, output $\perp$ and terminate; otherwise, re-encrypt the ciphertext to be under key pk_Y as:

1. Compute $B' = e(B, rk^{(1)}_{X,Y}) = e(g,g)^{yr}$.
2. Select a one-time signature key pair as $\texttt{SIG}.\mathcal{G}(1^k) \rightarrow (s\bar{v}k, s\bar{s}k)$. Set $\bar{A} = s\bar{v}k$.
3. Select a random number $\bar{r} \in \mathbb{Z}_q$ and compute

$$\bar{B} = (rk^{(3)}_{X,Y})^{\bar{r}}, \ \bar{C} = F(\bar{A})^{\bar{r}}, \ \bar{D} = h^{\bar{r}}, \ \bar{v} = e(g', g)^{\bar{r}}, \ \bar{k} = H(\bar{v}).$$

4. Run the encrypting algorithm $\texttt{SKE.Enc}(\bar{k}, B'||rk^{(2)}_{X,Y}||A||B||C||D||E||S)$, where the plaintext is $B'||rk^{(2)}_{X,Y}||A||B||C||D||E||S$, and denote the ciphertext $\bar{E}$.

[4] The 2-DL problem is: given (g, g^a, g^{a^2}) as input, to compute a.

[5] In fact, as mentioned in [9], we use y instead of $\tilde{y}$ for simplicity, where $\tilde{y}$ is a fixed one-to-one representation of y in $\mathbb{Z}_q$. This one-to-one mapping from y to $\tilde{y}$ can be implemented by an additional hash function.

5. Run the signing algorithm $\texttt{SIG}.\mathcal{S}(\bar{ssk}, (\bar{B}||\bar{C}||\bar{D}||\bar{E}))$, where the message to sign is the tuple $(\bar{B}, \bar{C}, \bar{D}, \bar{E})$, and denote the signature $\bar{S}$.
6. Output the new ciphertext $(\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$.

$\texttt{PRE.Dec}$**:** On input a secret key sk and any ciphertext K, parse K,

Case $K = (A, B, C, D, E, S)$**:** If $Check(K, g^{sk}) = 0$, output $\perp$ and terminate; otherwise, compute $v = e(B, g)^{1/sk}$, $k = H(v)$.

Case $K = (\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$**:** If $Check(K, g^{sk}) = 0$, output $\perp$ and terminate; otherwise, compute $\bar{v} = e(\bar{B}, g')^{1/sk}$, $\bar{k} = H(v)$, decrypt $\bar{E}$ under $\bar{k}$ using $\texttt{SKE.Dec}$ to get $B'||rk^{(2)}_{X,Y}||A||B||C||D||E||S$. If $Check((A, B, C, D, E, S), rk^{(2)}_{X,Y}) = 0$, output $\perp$ and terminate; otherwise, compute $v = B'^{1/sk}$, $k = H(v)$.

Then decrypt E under k using $\texttt{SKE.Dec}$ to get m (m may be $\perp$).

Correctness. The correctness property can be obtained from the following equations.

Case $K = (A, B, C, D, E, S)$**:** We have

$$e(B, g)^{1/sk} = e(pk^r, g)^{1/sk} = e(g, g)^r = v.$$

Case $K = (\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$**:** We have

$$e(\bar{B}, g')^{1/sk} = e(pk^{\bar{r}}, g')^{1/sk} = e(g', g)^{\bar{r}} = \bar{v},$$

and

$$B'^{1/sk} = e(B, rk^{(1)}_{X,Y})^{1/sk} = (e(g, g)^{sk\cdot r})^{1/sk} = e(g, g)^r = v.$$

Remark 4. We have no explicit test for checking the validity of B'. However, if it is malformed, the decrypted session key $H(B'^{1/sk})$ will be uniformly random and independent of the true key. Hence, the final decrypted output will be either correct or indistinguishable from random.

It seems that $(rk^{(2)}_{X,Y}, A, B, C, D, S)$ is useless, since we can compute m from (B', E) and the validity of m can be guaranteed by $\texttt{SKE}$. However, we cannot removed them from the re-encrypted ciphertext. Because we cannot answer the decryption oracle when we don't have $(rk^{(2)}_{X,Y}, A, B, C, D, S)$. See the details in the security proof.

3.1 Security Analysis

For convenience, we will prove security under equivalent formulations of 3-QDBDH assumption and e3-QDBDH assumption as that in [22].

Lemma 1. *The 3-QDBDH problem is equivalent to decide whether Q equals $e(g, g)^b$ or a random value, given $(g, g^{1/a}, g^a, g^{a^2}, g^{a^2\cdot b})$ as input.*

Proof. Given $(g, g^{1/a}, g^a, g^{a^2}, g^{a^2\cdot b})$, we can build a 3-QDBDH instance by setting $(y = g^{1/a}, y^A = g, y^{A^2} = g^a, y^{A^3} = g^{a^2}, y^B = g^{a^2\cdot b})$, which implicitly define $A = a$ and $B = a^3\cdot b$. Then, we have $e(y, y)^{B/A} = e(g^{1/a}, g^{1/a})^{(a^3\cdot b)/a} = e(g, g)^b$. The converse implication is easily established and demonstrates the equivalence between both problems.■

Similarly, we get the following.

Lemma 2. *The 3-QDBDH problem is equivalent to decide whether Q equals $e(g, g)^{bc}$ or a random value, given $(g, g^{1/a^2}, g^{1/a}, g^a, g^{a^2}, g^{a^2\cdot b}, g^c)$ as input.*

Theorem 1 (Uni-PRE-CCA security). *Our proposal is Uni-PRE-CCA-secure in the standard model under assumptions that the 3-QDBDH problem and the e3-QDBDH problem are hard, that $\texttt{SKE}$ is CCA-secure, and that $\texttt{SIG}$ is strongly unforgeable. In particular, we have*

$$\mathbf{Adv}^{CCA\text{-}\mathcal{A}}_{\texttt{PRE}}(k) \leq \begin{cases} q_{pk} \cdot (\epsilon_{\texttt{e3-QDBDH}} + \epsilon_{\texttt{SIG}} + q_{r+d} \cdot \delta + \epsilon^{CCA}_{\texttt{SKE}}), & \textit{first-level;} \\ q_{pk} \cdot (\epsilon_{\texttt{3-QDBDH}} + \epsilon_{\texttt{SIG}} + q_{r+d} \cdot \delta + \epsilon^{CCA}_{\texttt{SKE}}), & \textit{second-level.} \end{cases}$$

where q_{pk} is the amount of queries to the public key generation oracle for uncorrupted public keys, q_{r+d} is the amount of queries to the re-encryption oracle and the decryption oracle, $\epsilon_{\texttt{e3-QDBDH}}$ is the probability that the adversary solves the e3-QDBDH problem, $\epsilon_{\texttt{3-QDBDH}}$ is the probability that the adversary solves the 3-QDBDH problem, $\epsilon_{\texttt{SIG}}$ is the probability that the adversary breaks the strong unforgeability security of $\texttt{SIG}$, δ is the probability that any given verification key is output by $\texttt{SIG}.\mathcal{G}$, and $\epsilon^{CCA}_{\texttt{SKE}}$ is the probability that the adversary breaks the CCA security of $\texttt{SKE}$.

Proof. We incrementally define a sequence of games starting at the real attack (Game G_0), and ending up at Game G_5, which clearly shows that the adversary cannot break the system. We define E_i to be the event that $\mathbf{b} = \mathbf{b'}$ in Game G_i, where $\mathbf{b}$ is the bit involved in the challenge phase, and $\mathbf{b'}$ is the output of $\mathcal{A}$ in the Guess phase.

Game G_0. This game corresponds to the real attack. By definition,

$$|\Pr[E_0] - 1/2| = \mathbf{Adv}^{\texttt{CCA-}\mathcal{A}}_{\texttt{PRE}}(k) \tag{1}$$

Game G_1. In this game, we modify the public key oracle and challenge phase as follows.

- $\mathcal{O}_{pk}$: On input an index i, if it is an uncorrupted public key, the challenger decides whether it is the challenge public key pk^*. If yes, the challenger marks the public key as $\mathbf{pk}$. The other performances in this oracle are the same as those in Game G_0.
- ***Challenge:*** if the challenge public key $pk^* \neq \mathbf{pk}$, the challenger reports "failure" and aborts; otherwise, the challenger does the same performances as those in Game G_0.

It is easy to see that only if the challenger guessed the correct challenge public key, Game G_1 and Game G_0 are indistinguishable. The probability that the challenger guesses the correct challenge public key is $1/q_{pk}$ at least. Hence, we have

$$|\Pr[E_1] - 1/2| \geq \frac{1}{q_{pk}} \cdot |\Pr[E_0] - 1/2| \tag{2}$$

Game G_2. In this game, we use the modified e3-QDBDH input and modified 3-QDBDH input to modify the Setup and Challenge phase.

- ***Setup:***
  - *the first-level ciphertext security:* given a modified e3-QDBDH input $(g, g^{1/a^2}, g^{1/a}, g^a, g^{a^2}, g^{a^2 \cdot b}, g^c, Q)$, the challenger sets up the global parameters for $\mathcal{A}$ as follows: the description of the groups $\langle g \rangle = \mathbb{G}, \mathbb{G}_T$, their prime order q, and the mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $(svk^*, ssk^*) \leftarrow \texttt{SIG}.\mathcal{G}(1^k)$, $g_2 = g^{\alpha_1}$, $g_3 = g^{a^2 \cdot \alpha_2 - \alpha_1 \cdot svk^*}$, $h = (g^{a^2})^w$, $g' = g^c$, where α_1, α_2, and w are random numbers from $\mathbb{Z}_q$. The system parameters are $(q, g, g', g_2, g_3, \mathbb{G}, \mathbb{G}_T, e, F, H, \texttt{SIG}, \texttt{SKE})$.
  - *the second-level ciphertext security*: given a modified 3-QDBDH input $(g, g^{1/a}, g^a, g^{a^2}, g^{a^2 \cdot b}, Q)$, the challenger sets up the global parameters for $\mathcal{A}$ as follows: the description of the groups $\langle g \rangle = \mathbb{G}, \mathbb{G}_T$, their prime order q, and the mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, $(svk^*, ssk^*) \leftarrow \texttt{SIG}.\mathcal{G}(1^k)$, $g_2 = g^{\alpha_1}$, $g_3 = g^{a^2 \cdot \alpha_2 - \alpha_1 \cdot svk^*}$, $h = (g^{a^2})^w$, $g' = g^u$ where α_1, α_2, w and u are four random numbers from $\mathbb{Z}_q$. The system parameters are $(q, g, g', g_2, g_3, \mathbb{G}, \mathbb{G}_T, e, F, H, \texttt{SIG}, \texttt{SKE})$.
- ***Challenge:*** The challenger sets $A^* = svk^*$, and computes S by using ssk^*. Other performances are the same as those in Game G_1.

Since $(g, g^{1/a^2}, g^{1/a}, g^a, g^{a^2}, g^{a^2 \cdot b}, g^c, Q, \alpha_1, \alpha_2, w, u)$ are random, the above setup phase and challenge phase are indistinguishable from those in Game G_1. Hence, we have

$$\Pr[E_2] = \Pr[E_1] \tag{3}$$

Game G_3. In this game, we modify the public key generation oracle, re-encryption oracle, and decryption oracle.

- ***Phase 1.***
  - $\mathcal{O}_{pk}$: the challenger picks a random $x_i \in \mathbb{Z}_q$. If it is a corrupted public key, compute $pk_i = g^{x_i}$. Otherwise, decide whether it is the challenge public key. If it is, compute $pk_i = (g^{a^2})^{x_i}$; otherwise, compute $pk_i = (g^a)^{x_i}$. At last, the challenger records the tuple (pk_i, x_i) in T_K, and responds $\mathcal{A}$ with pk_i.
  - $\mathcal{O}_{sk}$: On input pk_i, the challenger checks whether pk_i exists in T_K, if not, the challenger aborts. Otherwise, if pk_i is a corrupted public key, the challenger outputs x_i; otherwise, the challenger outputs $\perp$.

- $\mathcal{O}_{rk}$: On input (pk_i, pk_j), the challenger checks whether pk_i and pk_j both exist in T_K, if not, the challenger aborts. Otherwise, the challenger does the following performances.
  - ∗ If pk_i and pk_j are both corrupted, the challenger responds $\mathcal{A}$ with g^{x_j/x_i}.
  - ∗ If pk_i and pk_j are both uncorrupted and neither of them is the guessed challenge public key, the challenger responds $\mathcal{A}$ with g^{x_j/x_i}.
  - ∗ If pk_i is corrupted, and pk_j is uncorrupted but not the guessed challenge public key, the challenger responds $\mathcal{A}$ with $(g^{1/a})^{x_j/x_i}$.
  - ∗ If pk_i is corrupted, and pk_j is the guessed challenge public key, the challenger responds $\mathcal{A}$ with $(g^a)^{x_j/x_i}$.
  - ∗ If pk_i is uncorrupted but not the guessed challenge public key, and pk_j is the guessed challenge public key, the challenger responds $\mathcal{A}$ with $(g^a)^{x_j/x_i}$.
  - ∗ If pk_i is uncorrupted but not the guessed challenge public key, and pk_j is corrupted, the challenger responds $\mathcal{A}$ with $(g^a)^{x_j/x_i}$.
  - ∗ If pk_i is the guessed challenge public key, and pk_j is uncorrupted but not the guessed challenge public key, the challenger responds $\mathcal{A}$ with $(g^{1/a})^{x_j/x_i}$.
  - ∗ (*only for the first-level ciphertext security*) If pk_i is the guessed challenge public key, and pk_j is corrupted, the challenger responds $\mathcal{A}$ with $(g^{1/a^2})^{x_j/x_i}$.
- $\mathcal{O}_{re}$: On input (pk_i, pk_j, K), the challenger checks whether pk_i and pk_j both exist in the table T_K, if not, the challenger aborts. Otherwise, if $Check(K, pk_i^{(1)}) = 0$, then the ciphertext is not well-formed, the challenger outputs $\perp$ and aborts; otherwise, the challenger parses $K = (A, B, C, D, E, S)$, and does the following performances.
  - ∗ *the first-level ciphertext security:* Return $\texttt{PRE.ReEnc}(\mathcal{O}_{rk}(pk_i, pk_j), K)$.
  - ∗ *the second-level ciphertext security:*
    - · If pk_i is the guessed challenge public key, and pk_j is a corrupted public key, the challenger computes:

$$t = \frac{C}{D^{\alpha_2/w}},\ \lambda = \frac{1}{\alpha_1(A - A^*)}.$$

      Then the challenger gets $B' = e((t^\lambda)^{x_j}, g)$, and compute $(\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$ as the real execution. At last, the challenger responds $(\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$.
      Note that when $A \neq A^*$, then the challenger can solve for $t^\lambda = g^r$ since:

$$\begin{aligned} t &= \tfrac{F(A)^r}{(h^r)^{\alpha_2/w}} = \tfrac{g_2^{r\cdot A} g_3^r}{g^{a^2\cdot r\cdot\alpha_2}} = \tfrac{(g^{\alpha_1})^{r\cdot A}(g^{a^2\cdot\alpha_2 - \alpha_1\cdot A^*})^r}{g^{r\cdot a^2\cdot\alpha_2}} \\ &= \tfrac{g^{r\alpha_1(A-A^*)+r\cdot a^2\cdot\alpha_2}}{g^{r\cdot a^2\cdot\alpha_2}} = g^{r\cdot\alpha_1(A-A^*)}. \end{aligned}$$

    - · Otherwise, Return $\texttt{PRE.ReEnc}(\mathcal{O}_{rk}(pk_i, pk_j), K)$.
- $\mathcal{O}_{dec}$: On input (pk_i, K), the challenger checks whether pk_i exists in table T_K, if not, the challenger outputs $\perp$. Otherwise, the challenger does the following performances.
  - ∗ If pk_i is corrupted, the challenger responds $\mathcal{A}$ with $\texttt{PRE.Dec}(x_i, K)$.
  - ∗ If pk_i is uncorrupted, the challenger parses K,

    Case $K = (A, B, C, D, E, S)$**:** If $Check(K, pk_i) = 0$, the challenger outputs $\perp$ and terminates; otherwise, the challenger solves for g^r as it does in $\mathcal{O}_{re}$, and computes $v = e(g^r, g)$, $k = H(v)$. Then decrypt E under k using $\texttt{SKE.Dec}$ to get m (m may be $\perp$).

    Case $K = (\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$**:** If $Check(K, pk_i) = 0$, the challenger outputs $\perp$ and terminates; otherwise, the challenger solves for $g^{\bar{r}}$ as it does in $\mathcal{O}_{re}$, and computes $\bar{v} = e(g^{\bar{r}}, g')$, $\bar{k} = H(\bar{v})$. Then decrypt $\bar{E}$ under $\bar{k}$ using $\texttt{SKE.Dec}$ to get $B'||rk_{X,i}^{(2)}||A||B||C||D||E||S$ (m may be $\perp$). the challenger searches $pk_X = rk_{X,i}^{(2)}$ in T_K, and calls $\mathcal{O}_{rk}$ with (pk_X, pk_i) to get the corresponding re-encryption key $rk_{X,i}$, and checks $e(B, rk_{X,i}^{(1)}) \overset{?}{=} B'$. If it holds, the challenger calls $\mathcal{O}_{dec}$ with $(pk_X, (A, B, C, D, E, S))$ to get the message m and outputs it; otherwise, the challenger outputs $\perp$.
    Note that in the real execution, according to the correctness of SKE, *iff $B' = e(B, rk_{X,i}^{(1)})$,* PRE.Dec *outputs a message m not $\perp$ when the input is a re-encrypted ciphertext. Hence, the above check process ($e(B, rk_{X,i}^{(1)}) \overset{?}{=} B'$) does not affect but guarantees the distinguishability between the simulation and the real execution.*

- ***Phase 2:*** Almost the same as Phase 1, but with the restrictions in Uni-PRE-CCA game.

The keys responses in Game G_3 are indistinguishable from those in Game G_2. The decryption oracle and re-encryption oracle are also indistinguishable from those in Game G_2, except that the challenger cannot always answer them when $A = svk^*$. On the one hand, before the challenge ciphertext is given, the adversary could query with a ciphertext where $A = svk^*$ with probability $q_{r+d} \cdot \delta$. On the other hand, after the challenge is given, the adversary could query with a well-formed ciphertext where $A = svk^*$ and yet the ciphertext is not the challenge ciphertext is $\epsilon_{\mathtt{SIG}}$. Note that $B = g^r$, $E = \mathtt{SKE.Enc}(H(e(g,g)^r), m)$ uniquely fix m. If $A = svk^*$ then the ciphertext must not be identical to the challenge ciphertext. If the ciphertext is well-formed, then S is a valid forgery against $\mathtt{SIG}$.

As a result, we have

$$|\Pr[E_3] - \Pr[E_2]| \leq \epsilon_{\mathtt{SIG}} + q_{r+d} \cdot \delta \tag{4}$$

Game G_4**.** In this game, we modify Challenge Phase.

- ***Challenge:***
  - *the first-level challenge ciphertext*: the challenger first runs $\mathtt{PRE.Enc}(pk_i, m_{\mathbf{b}})$ $(pk_i \neq pk^*)$ to get (A, B, C, D, E, S), and queries $\mathcal{O}_{rk}$ with (pk_i, pk^*) to get re-encryption key. Then the challenger uses the re-encryption key to get B', and computes

$$A^* = svk^*,\ B^* = (g^{a^2 \cdot b})^{x^*} = (pk^*)^b,$$
$$C^* = (g^{a^2 \cdot b})^{\alpha_2} = ((g^{\alpha_1})^{A^*} \cdot g^{a^2 \cdot \alpha_2 - \alpha_1 \cdot A^*})^b = (g_2^{A^*} \cdot g_3)^b = F(A^*)^b,$$
$$D^* = (g^{a^2 \cdot b})^w = h^b,\ v^* = Q,\ k^* = H(v^*),$$
$$E^* = \mathtt{SKE.Enc}(k^*, B' || pk_i^{(1)} || A || B || C || D || E || S),$$
$$S^* = \mathtt{SIG}.\mathcal{S}(ssk^*, (B^*, C^*, D^*, E^*)),$$

  - *the second-level challenge ciphertext*:

$$A^* = svk^*,\ B^* = (g^{a^2 \cdot b})^{x^*} = (pk^*)^b,$$
$$C^* = (g^{a^2 \cdot b})^{\alpha_2} = ((g^{\alpha_1})^{A^*} \cdot g^{a^2 \cdot \alpha_2 - \alpha_1 \cdot A^*})^b = (g_2^{A^*} \cdot g_3)^b = F(A^*)^b,$$
$$D^* = (g^{a^2 \cdot b})^w = h^b,\ v^* = Q,\ k^* = H(v^*),$$
$$E^* = \mathtt{SKE.Enc}(k^*, m_{\mathbf{b}}),\ S^* = \mathtt{SIG}.\mathcal{S}(ssk^*, (B^*, C^*, D^*, E^*)),$$

  where x^* is in the same item with pk^* in the table T_K. the challenger returns $K^* = (A^*, B^*, C^*, D^*, E^*, S^*)$ to $\mathcal{A}$.

It is easy to see that when the challenger receives an e3-QDBDH instance (for the first-level challenge ciphertext) and a 3-QDBDH instance (for the second-level challenge ciphertext) as input, its challenge ciphertext is a perfectly distributed, proper encryption of message $m_{\mathbf{b}}$. Hence, we have

$$|\Pr[E_4] - \Pr[E_3]| \leq \begin{cases} \epsilon_{\mathtt{e3\text{-}QDBDH}}, & \text{first-level;} \\ \epsilon_{\mathtt{3\text{-}QDBDH}}, & \text{second-level.} \end{cases} \tag{5}$$

Game G_5**.** In this game, we continue to modify the challenge phase, such that k^* is chosen randomly from $\{0,1\}^{k_1}$. Since Q is uniformly distributed, $H(v^*)$ is uniformly distributed over $\{0,1\}^{k_1}$. Hence, we have

$$\Pr[E_5] = \Pr[E_4] \tag{6}$$

In this game,

- *the second-level ciphertext security*: only $E^* = \mathtt{SKE.Enc}(k^*, m_{\mathbf{b}})$ uniquely fixes $m_{\mathbf{b}}$ no matter in the original challenge ciphertext or the re-encrypted ciphertexts of the challenge ciphertext.

– *the first-level ciphertext security*: only B', E uniquely fixes $m_{\mathbf{b}}$ no matter in the challenge ciphertext, and E^* uniquely fixes B', E no matter in the challenge ciphertext.

Hence, we have that

$$\Pr[E_5] = \epsilon_{\mathtt{SKE}}^{\mathtt{CCA}} + 1/2 \tag{7}$$

Combining the equations (1)—(7), we get the result as required. ■

As mentioned before, the IE-CCA security (the first-level ciphertext security) implies the CR-CCA security, however, we still give the following theorem of CR-CCA security for the completeness.

Theorem 2. *Our proposal is Uni-PRE-CR secure in the standard model under the EDL assumption in $\mathbb{G}$. In particular, we have*

$$\mathbf{Adv}_{\mathit{PRE}}^{\mathit{CR}\text{-}\mathcal{A}}(k) \leq \epsilon_{\mathit{EDL}},$$

where ϵ_{EDL} is the probability that the adversary solves the EDL problem in $\mathbb{G}$.

Proof. Assume that there is an adversary $\mathcal{A}$ that can break the collusion resistance of our proposal, then we can build an algorithm $\mathcal{B}$ that solves the EDL problem by using $\mathcal{A}$. On input $(g, g^{1/a}, g^a)$, $\mathcal{B}$ aims to output a. At first, $\mathcal{B}$ set $g' = g^u$, where u is a random number from $\mathbb{Z}_q$.

– ***Find:***
  - $\mathcal{O}_{pk}$: $\mathcal{B}$ chooses random numbers x_i from $\mathbb{Z}_q$,
    * if it is a corrupted public key, set $pk_i = g^{x_i}$.
    * if it is an uncorrupted public key, set $pk_i = (g^a)^{x_i}$.

    At last, $\mathcal{B}$ records (pk_i, x_i) into the table T_k.
  - $\mathcal{O}_{sk}$: On input pk_i by the adversary, if it is a corrupted public key, $\mathcal{B}$ searches tuple (pk_i, x_i) in the table T_k, and returns the corresponding x_i to the adversary; otherwise, $\mathcal{B}$ outputs $\perp$.
  - $\mathcal{O}_{rk}$: On input (pk_i, pk_j), where pk_i and pk_j are both from $\mathcal{O}_{pk}$, $\mathcal{B}$ first searches the tuples corresponding to pk_i and pk_j in the table T_k, and gets the values of x_i and x_j.
    * If pk_i and pk_j are both corrupted or both uncorrupted, $\mathcal{B}$ computes $rk_{i,j} = g^{x_j/x_i}$.
    * If pk_i is corrupted, and pk_j is uncorrupted, $\mathcal{B}$ computes $rk_{i,j} = (g^a)^{x_j/x_i}$.
    * If pk_i is uncorrupted, and pk_j is corrupted, $\mathcal{B}$ computes $rk_{i,j} = (g^{1/a})^{x_j/x_i}$.
  - $\mathcal{O}_{dec}$: On input (pk_i, K), the challenger checks whether pk_i exists in table T_K, if not, the challenger outputs $\perp$. Otherwise, the challenger does the following performances.
    * If pk_i is corrupted, the challenger responds $\mathcal{A}$ with $\mathtt{PRE.Dec}(x_i, K)$.
    * If pk_i is uncorrupted, the challenger parses K,

      Case $K = (A, B, C, D, E, S)$**:** If $Check(K, pk_i) = 0$, the challenger outputs $\perp$ and terminates; otherwise, the challenger computes $e(B, g^{1/a})^{1/x_i} = e(g, g)^r$, and then uses $e(g, g)^r$ to get m as the real execution.

      Case $K = (\bar{A}, \bar{B}, \bar{C}, \bar{D}, \bar{E}, \bar{S})$**:** If $Check(K, pk_i) = 0$, the challenger outputs $\perp$ and terminates; otherwise, the challenger outputs $\perp$ and terminates; otherwise, the challenger computes $e(B, g^{1/a})^{u/x_i} = e(g, g')^{\bar{r}}$, and then uses $e(g, g')^{\bar{r}}$ to get $B'||rk_{X,i}^{(2)}||A||B||C||D||E||S$ (may be $\perp$). the challenger searches $pk_X = rk_{X,i}^{(2)}$ in T_K, and calls $\mathcal{O}_{rk}$ with (pk_X, pk_i) to get the corresponding re-encryption key $rk_{X,i}$, and checks $e(B, rk_{X,i}) \overset{?}{=} B'$. If it holds, the challenger calls $\mathcal{O}_{dec}$ with $(pk_X, (A, B, C, D, E, S))$ to get the message m and outputs it; otherwise, the challenger outputs $\perp$.

      Note that in the real execution, according to the correctness of $\mathtt{SKE}$*, iff $B' = e(B, rk_{X,i}^{(1)})$,* $\mathtt{PRE.Dec}$ *outputs a message m not $\perp$ when the input is a re-encrypted ciphertext. Hence, the above check process ($e(B, rk_{X,i}^{(1)}) \overset{?}{=} B'$) does not affect but guarantees the distinguishability between the simulation and the real execution.*

– ***Output:*** The adversary outputs sk_i corresponding to an uncorrupted public key pk_i, and $\mathcal{B}$ outputs sk_i/x_i as the solution of the 3-EDL problem, where x_i is the corresponding value to pk_i in the table T_K.

The above simulation always succeeds, hence we have

$$\mathbf{Adv}_{\mathtt{PRE}}^{\mathtt{CR}\text{-}\mathtt{CCA}\text{-}\mathcal{A}}(k) \leq \epsilon_{\mathtt{EDL}}.$$

□

4 Conclusions

By using the techniques of Kurosawa-Desmedt and Canetti-Hohenberger, we propose a new single-use unidirectional proxy re-encryption scheme, which is CCA-secure and collusion resistant in the standard model.

References

1. http://tdt.sjtu.edu.cn/~jshao/prcbib.htm.
2. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In *Internet Society (ISOC): NDSS 2005*, pages 29–43, 2005.
3. G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.
4. M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127–144, 1998.
5. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 231–229, 2001.
6. D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
7. E. Bresson, D. Catalano, and D. Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In *ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 37–54, 2003.
8. R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 207–222, 2004.
9. R. Canetti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In *ACM CCS 2007*, 2007. Full version: Cryptology ePrint Archieve: Report 2007/171.
10. Y-P. Chiu, C-L. Lei, and C-Y. Huang. Secure multicast using proxy encryption. In *ICICS 2005*, volume 3783 of *LNCS*, pages 280–290, 2005.
11. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
12. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
13. E. Fujisaki and T. Okamoto. How to enhance the security of public-key encryption at minimum cost. In *PKC 1999*, volume 1560 of *LNCS*, pages 53–68, 1999.
14. M. Green and G. Ateniese. Identity-Based Proxy Re-encryption. In *ACNS 2007*, volume 4521 of *LNCS*, pages 288–306, 2007. Full version: Cryptology ePrint Archieve: Report 2006/473.
15. T.S. Heydt-Benjamin, H. Chae, B. Defend, and K. Fu. Privacy for public transportation. In *PET 2006*, volume 4258 of *LNCS*, pages 1–19, 2005.
16. A. Ivan and Y. Dodis. Proxy Cryptography Revisited. In *Internet Society (ISOC): NDSS 2003*, 2003.
17. M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-encryption. In *PKC 1999*, volume 1560 of *LNCS*, pages 112–121, 1999.
18. H. Khurana and H-S. Hahm. Certified mailing lists. In *ASIACCS 2006*, pages 46–58, 2006.
19. H. Khurana and R. Koleva. Scalable security and accounting services for content-based publish subscribe systems. *International Journal of E-Business Research*, 2(3), 2006.
20. H. Khurana, A. Slagell, and R. Bonilla. Sels: A secure e-mail list service. In *ACM SAC 2005*, pages 306–313, 2005.
21. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 426–442, 2004.
22. B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. In *PKC 2008*, volume 4939 of *LNCS*, pages 360–379, 2008.
23. J. Shao and Z. Cao. CCA-Secure Proxy Re-Encryption without Pairings. In *PKC 2009*, volume 5443 of *LNCS*, pages 357–376, 2009.
24. J. Shao, Z. Cao, and P. Liu. Sccr: a generic approach to simultaneously achieve cca security and collusion-resistance in proxy re-encryption. *SECURITY AND COMMUNICATION NETWORKS*, 2009.
25. G. Taban, A.A. Cárdenas, and V.D. Gligor. Towards a secure and interoperable drm architecture. In *ACM DRM 2006*, pages 69–78, 2006.
26. A. Talmy and O. Dobzinski. Abuse freedom in access control schemes. In *AINA 2006*, pages 77–86, 2006.
27. Q. Tang. Type-based proxy re-encryption and its construction. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 130–144, 2008.
28. J. Weng, S.S.M. Chow, Y. Yang, and R.H. Deng. Efficient Unidirectional Proxy Re-Encryption. http://eprint.iacr.org/2009/189.

29. J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ACM ASIACCS 2009*, pages 322–332, 2009.
30. L. Zhou, M.A. Marsh, F.B. Schneider, and A. Redz. Distributed Blinding for Distributed ElGamal Re-encryption. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 824–834, 2005.