

# Chosen-Ciphertext Secure Proxy Re-Encryption Schemes without Pairings

Jian Weng<sup>1,2,3</sup>, Robert H. Deng<sup>1</sup>, Shengli Liu<sup>3</sup>, Kefei Chen<sup>3</sup>, Junzuo Lai<sup>3</sup>, Xu An Wang<sup>4</sup>

<sup>1</sup>School of Information Systems, Singapore Management University, Singapore 178902

<sup>2</sup> Department of Computer Science, Jinan University, Guangzhou 510632, P.R. China

<sup>3</sup>Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, P.R. China

<sup>4</sup>Key Laboratory of Information and Network Security, Engineering College of Chinese Armed Police Force.  
cryptjweng@gmail.com

## Abstract

Proxy re-encryption (PRE), introduced by Blaze, Bleumer and Strauss, allows a semi-trusted proxy to convert a ciphertext originally intended for Alice into an encryption of the same message intended for Bob. PRE systems can be categorized into *bidirectional PRE* (in which the delegation from Alice to Bob also allows re-encryption from Bob to Alice) and *unidirectional PRE* (in which the delegation does not apply in the opposite direction). In their ACM CCS'07 paper, Canetti and Hohenberger presented a bidirectional PRE scheme secure against chosen-ciphertext attack (CCA), and left an important open problem to construct a CCA-secure PRE scheme without pairings. In this paper, we solve this open problem by presenting two direct constructions of CCA-secure PRE schemes *without pairings*: one is bidirectional and the other is unidirectional. Both schemes are fairly efficient due to two distinguished features: (i) they do not use the costly bilinear pairings; (ii) the computational cost and the ciphertext length *decrease* with re-encryption. In contrast, the existing CCA-secure PRE schemes do not share these desirable features.

**Keywords:** Proxy re-encryption, bilinear pairing, chosen-ciphertext security.

## 1 Introduction

### 1.1 Background

Imagine that one day you are on vacation and is inconvenient to access your email. You would wish to have the mail server forward your encrypted email messages to your colleague Bob, who can then read the them using his own secret key. A naive way is to have the mail server store your secret key and act as follows: when an encrypted email message for you arrives, the mail server decrypts it using the stored secret key and re-encrypts the plaintext using Bob's public key. However, such a solution is highly undesirable, especially in the case that the email server is untrustworthy, since the server learns both the plaintext and your secret key.

Proxy re-encryption (PRE), introduced by Blaze, Bleumer and Strauss [4], is a novel solution to the above scenario. In a PRE system, a proxy is given a re-encryption key  $rk_{i,j}$  so that it can convert a ciphertext under public key  $pk_i$  into a ciphertext of the same plaintext under a different public key  $pk_j$ . The proxy, however, learns nothing

about the plaintext under either key. Now, as to the aforementioned scenario, you can have the email server act as a proxy, and give it the re-encryption key instead of your secret key. Then the email server translates encrypted email messages intended to you into those encrypted under Bob’s public key, without learning the content of the email messages. Proxy re-encryptions find many other practical applications, such as distributed file systems, outsourced filtering of encrypted spam, and access control over network storage [1, 2, 23].

Blaze, Bleumer and Strauss [4] categorized two types of PRE schemes: *bidirectional* PRE (in which the re-encryption key  $rk_{i,j}$  allows the proxy to convert ciphertexts under  $pk_i$  into ciphertexts under  $pk_j$  and *vice versa*) and *unidirectional* PRE (in which the  $rk_{i,j}$  allows the proxy to convert only from  $pk_i$  to  $pk_j$ ). Blaze *et al.* [4] proposed the first bidirectional PRE scheme in 1998. In 2005, Ateniese *et al.* [1, 2] presented a unidirectional PRE scheme based on bilinear pairings. Both schemes are only secure against chosen-plaintext attack (CPA). However, applications often require security against chosen-ciphertext attacks (CCA).

To fill this gap, Canetti and Hohenberger [10] presented an elegant construction of CCA-secure bidirectional PRE scheme. Later, Libert and Vergnaud [20] presented a unidirectional PRE scheme with replayable chosen-ciphertext (RCCA) security. Both of these constructions rely on bilinear pairings. In spite of the recent advances in implementation technique, compared with standard operations such as modular exponentiation in finite fields [8], the pairing computation is still considered as a very expensive operation. It would be desirable for cryptosystems to be constructed without relying on pairings, especially in computation resource limited settings. In view of this, Canetti and Hohenberger left an important open problem in [10], i.e., how to construct a CCA-secure proxy re-encryption scheme without pairings.

## 1.2 Our Contributions

In this paper, we first circumvent several obstacles and construct a bidirectional PRE scheme without pairings. Based on the modified computational Diffie-Hellman (mCDH) problem, we prove its CCA security in the random oracle model. Compared with existing CCA-secure bidirectional proxy re-encryption schemes, our scheme is much more efficient due to the following facts: (i) our scheme does not use the costly bilinear pairing; (ii) the computational cost and the ciphertext length in our scheme decrease with re-encryption. In contrast, the existing CCA-secure bidirectional PRE schemes cannot share these desirable features.

Based on our bidirectional PRE scheme, we further propose a unidirectional PRE scheme without pairings. The chosen-ciphertext security of this scheme can be proved under the well-studied computational Diffie-Hellman (CDH) assumption. Again, the computational cost and the ciphertext length in our scheme *decrease* with re-encryption, whereas those in the existing unidirectional PRE schemes *increase* with re-encryption.

## 1.3 Related Works

Boneh, Goh and Matsuo [7] described a hybrid proxy re-encryption system based on the ElGamal-type public key encryption system [13] and Boneh-Boyen’s identity-based encryption system [3]. Recently, Libert and Vergnaud [22] proposed a traceable proxy re-encryption system, in which a proxy that leaks its re-encryption key can be identified by the delegator. Green and Ateniese [15] considered proxy re-encryption in identity-based scenarios: based on Boneh and Franklin’s identity-based encryption system [6],

they presented the first CPA and CCA-secure identity-based proxy re-encryption (IB-PRE) schemes in the random oracle model. Later, Chu and Tzeng [11] presented the constructions of CPA and CCA-secure IB-PRE schemes without random oracles.

Another kind of cryptosystems related to proxy re-encryption is the *proxy encryption* cryptosystem [12,17,24]. In NDSS'03, Dodis and Ivan [12] presented generic constructions of proxy encryption schemes as well as several efficient concrete schemes. It should be noted that, as argued in [10,20], proxy re-encryption schemes are a (strict) subset of proxy encryption schemes. In proxy encryption systems, a delegator allows a delegatee to decrypt ciphertexts intended for her with the help of a proxy by providing them with shares of her private key. This approach requires the delegatee to store an additional secret for each delegation. In contrast, the delegatee in proxy re-encryption schemes only needs to have its own decryption key.

Proxy re-encryption should not be confused with the universal re-encryption [16], in which the ciphertexts are re-randomized instead of changing the underlying public key.

## 1.4 Outline

The rest of the paper is organized as follows. In Section 2, we review the model of PRE systems and some complexity assumptions related to our proposed schemes. In Section 3, we propose a bidirectional PRE scheme without pairings, and give a comparison between our scheme and other existing bidirectional PRE schemes. We also prove the CCA-security of our bidirectional PRE scheme. In Section 4, based on our bidirectional PRE scheme, we further propose a unidirectional PRE scheme without pairings, and then prove its security. Finally, Section 5 lists some open problems and concludes the paper.

# 2 Preliminaries

## 2.1 Notations

We first present some notations used in the rest of this paper. For a prime  $q$ , let  $\mathbb{Z}_q$  denote the set  $\{0, 1, 2, \dots, q-1\}$ , and  $\mathbb{Z}_q^*$  denote  $\mathbb{Z}_q \setminus \{0\}$ . For a finite set  $S$ ,  $x \xleftarrow{\$} S$  means choosing an element  $x$  from  $S$  with a uniform distribution.

## 2.2 Model of Proxy Re-Encryption Systems

In this subsection, we review the definitions and security models for bidirectional and unidirectional proxy re-encryption systems.

Formally, a bidirectional PRE scheme consists of the following six algorithms [10]:

**GlobalSetup**( $\kappa$ ): The global setup algorithm takes as input a security parameter  $\kappa$ . It outputs the global parameters *param*.

For brevity, we assume that *param* is implicitly included in the input of the following algorithms.

**KeyGen**( $i$ ): The key generation algorithm generates the public/secret key pair  $(pk_i, sk_i)$  for user  $i$ .

**ReKeyGen**( $sk_i, sk_j$ ): The re-encryption key generation algorithm takes as input two secret keys  $sk_i$  and  $sk_j$ . It outputs a re-encryption key  $rk_{i,j}$ .

**Encrypt**( $pk, m$ ): The encryption algorithm takes as input a public key  $pk$  and a message  $m \in \mathcal{M}$ . It outputs a ciphertext CT under  $pk$ . Here  $\mathcal{M}$  denotes the message space.

**ReEncrypt**( $rk_{i,j}, CT_i$ ): The re-encryption algorithm takes as input a re-encryption key  $rk_{i,j}$  and a ciphertext  $CT_i$  under public key  $pk_i$ . It outputs a ciphertext  $CT_j$  under public key  $pk_j$ .

**Decrypt**( $sk, CT$ ): The decryption algorithm takes as input a secret key  $sk$  and a ciphertext  $CT$ . It outputs a message  $m \in \mathcal{M}$  or the error symbol  $\perp$ .

Roughly speaking, the correctness requires that, for any  $m \in \mathcal{M}$  and any couple of public/secret key pair  $(pk_i, sk_i), (pk_j, sk_j)$ , the following conditions hold:

$$\begin{aligned} \text{Decrypt}(sk_i, \text{Encrypt}(pk_i, m)) &= m, \\ \text{Decrypt}(sk_j, \text{ReEncrypt}(\text{ReKeyGen}(sk_i, sk_j), \text{Encrypt}(pk_i, m))) &= m. \end{aligned}$$

**Remark 1.** The definition of unidirectional PRE scheme is the same as that of bidirectional PRE scheme, with the exception that, the re-encryption key generation algorithm **ReKeyGen** takes as input the secret key  $sk_i$  and the public key  $pk_j$  instead of  $sk_j$ . We write it as **ReKeyGen**( $sk_i, pk_j$ ).

**Remark 2.** A proxy re-encryption scheme is said to be *multi-hop*, if a ciphertext can be consecutively re-encrypted, i.e., it can be re-encrypted from  $pk_1$  to  $pk_2$  and then to  $pk_3$  and so on. In contrast, a proxy re-encryption scheme is said to be *single-hop*, if a re-encrypted ciphertext can not be further re-encrypted. In this paper, we concentrate on single-hop proxy re-encryption schemes. Besides, for consistency and easy explanation, we adopt a term as used in [20]: the original ciphertext is called the *second-level ciphertext*, while the re-encrypted ciphertext is called the *first-level ciphertext*.

Next, we review the security notion for PRE systems. This security notion is derived from [10, 20], with slight modifications to allow the *adaptive* corruptions of users. The chosen-ciphertext security for a PRE scheme  $\Pi$  can be defined via the following game between an adversary  $\mathcal{A}$  and a challenger  $\mathcal{C}$ :

**Setup.**  $\mathcal{C}$  takes a security parameter  $\kappa$  and runs algorithm **GlobalSetup**. It gives  $\mathcal{A}$  the resulting global parameters *param*.

**Phase 1.**  $\mathcal{A}$  adaptively issues queries  $q_1, \dots, q_m$  where query  $q_i$  is one of the following:

- *Uncorrupted key generation query*  $\langle i \rangle$ :  $\mathcal{C}$  first runs algorithm **KeyGen** to obtain a public/secret key pair  $(pk_i, sk_i)$ , and then sends  $pk_i$  to  $\mathcal{A}$ .
- *Corrupted key generation query*  $\langle j \rangle$ :  $\mathcal{C}$  first runs algorithm **KeyGen** to obtain a public/secret key pair  $(pk_j, sk_j)$ , and then gives  $(pk_j, sk_j)$  to  $\mathcal{A}$ .
- *Re-encryption key generation query*  $\langle pk_i, pk_j \rangle$ :  $\mathcal{C}$  first runs **ReKeyGen**( $sk_i, sk_j$ ) if  $\Pi$  is a bidirectional PRE scheme, or algorithm **ReKeyGen**( $sk_i, pk_j$ ) if  $\Pi$  is unidirectional, to generate a re-encryption key  $rk_{i,j}$ . Finally,  $\mathcal{C}$  returns  $rk_{i,j}$  to  $\mathcal{A}$ . Here  $sk_i$  and  $sk_j$  are secret keys with respect to  $pk_i$  and  $pk_j$  respectively. It is required that  $pk_i$  and  $pk_j$  were generated beforehand by algorithm **KeyGen**. As argued in [10], for bidirectional PRE schemes, we require that either both  $pk_i$  and  $pk_j$  are corrupted, or alternately both are uncorrupted.
- *Re-encryption query*  $\langle pk_i, pk_j, CT_i \rangle$ :  $\mathcal{C}$  first runs algorithm **ReKeyGen** to generate the re-encryption key  $rk_{i,j}$ . Then it runs **ReEncrypt**( $rk_{i,j}, CT_i$ ) to obtain the resulting ciphertext  $CT_j$ , which is returned to  $\mathcal{A}$ . It is required that  $pk_i$  and  $pk_j$  were generated beforehand by **KeyGen**.
- *Decryption query*  $\langle pk, CT \rangle$ : Challenger  $\mathcal{C}$  returns the result of **Decrypt**( $sk, CT$ ) to  $\mathcal{A}$ , where  $sk$  is the secret key with respect to  $pk$ . It is required that  $pk$  was generated beforehand by **KeyGen**.

**Challenge.** Once  $\mathcal{A}$  decides that Phase 1 is over, it outputs two equal-length plaintexts  $m_0, m_1 \in \mathcal{M}$  and a target public key  $pk_{i^*}$  which is generated by the *uncorrupted* key generation query  $\langle i^* \rangle$ . For a unidirectional PRE scheme  $\Pi$ , it is also required that, for any public key  $pk_j$ ,  $\mathcal{A}$  has never simultaneously issued the re-encryption key generation query  $\langle pk_{i^*}, pk_j \rangle$  and the corrupted key generation query  $\langle j \rangle$ . Challenger  $\mathcal{C}$  flips a random coin  $\delta \in \{0, 1\}$ , and sets the challenge ciphertext to be  $CT^* = \text{Encrypt}(pk_{i^*}, m_\delta)$ , which is sent to  $\mathcal{A}$ .

**Phase 2.**  $\mathcal{A}$  issues additional queries  $q_{m+1}, \dots, q_{max}$  where each of the queries is one of the following:

- *Uncorrupted key generation query*  $\langle i \rangle$ :  $\mathcal{C}$  responds as in Phase 1.
- *Corrupted key generation query*  $\langle j \rangle$ :  $\mathcal{C}$  responds as in Phase 1. Here it is required that  $pk_j \neq pk_{i^*}$ . Besides, if  $\mathcal{A}$  has obtained a *derivative*<sup>1</sup>  $(pk', CT')$  of  $(pk_{i^*}, CT^*)$ , it is required that  $pk_j \neq pk'$ .
- *Re-encryption key generation query*  $\langle pk_i, pk_j \rangle$ : Challenger  $\mathcal{C}$  responds as in Phase 1. Here it is required that, if  $\mathcal{A}$  has obtained the secret key  $sk_j$  with respect to  $pk_j$ ,  $\mathcal{A}$  is disallowed to issue the re-encryption key generation query  $\langle pk_{i^*}, pk_j \rangle$  (and  $\langle pk_j, pk_{i^*} \rangle$  if  $\Pi$  is bidirectional).
- *Re-encryption query*  $\langle pk_i, pk_j, CT_i \rangle$ : Challenger  $\mathcal{C}$  responds as in Phase 1. Here it is required that, if  $\mathcal{A}$  has obtained the secret key  $sk_j$  with respect to  $pk_j$ , then  $(pk_i, CT_i)$  can not be a derivative of  $(pk_{i^*}, CT^*)$ .
- *Decryption query*  $\langle pk, CT \rangle$ : Challenger  $\mathcal{C}$  responds as in Phase 1. Here it is required that,  $(pk, CT)$  can not be a derivative of  $(pk_{i^*}, CT^*)$ .

**Guess.** Finally,  $\mathcal{A}$  outputs a guess  $\delta' \in \{0, 1\}$ .

We refer to adversary  $\mathcal{A}$  as an IND-PRE-CCA adversary, and we define his advantage in attacking scheme  $\Pi$  as

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-PRE-CCA}} = \left| \Pr[\delta' = \delta] - \frac{1}{2} \right|,$$

where the probability is taken over the random coins consumed by the challenger and the adversary. Note that the chosen plaintext security for a PRE scheme can be similarly defined as the above game except that the adversary is not allowed to issue any decryption queries.

**Definition 1** *A PRE scheme  $\Pi$  is said to be  $(t, q_u, q_c, q_{rk}, q_{re}, q_d, \epsilon)$ -IND-PRE-CCA secure, if for any  $t$ -time IND-PRE-CCA adversary  $\mathcal{A}$  who makes at most  $q_u$  uncorrupted key generation queries, at most  $q_c$  corrupted key generation queries, at most  $q_{rk}$  re-encryption key generation queries, at most  $q_{re}$  re-encryption queries and at most  $q_d$  decryption queries, we have  $\text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-PRE-CCA}} \leq \epsilon$ .*

---

<sup>1</sup>Derivative of  $(pk_{i^*}, CT^*)$  is inductively defined in [10] as below:

1.  $(pk_{i^*}, CT^*)$  is a derivative of itself;
2. If  $(pk, CT)$  is a derivative of  $(pk_{i^*}, CT^*)$  and  $(pk', CT')$  is a derivative of  $(pk, CT)$ , then  $(pk', CT')$  is a derivative of  $(pk_{i^*}, CT^*)$ .
3. If  $\mathcal{A}$  has issued a re-encryption query  $\langle pk, pk', CT \rangle$  and obtained the resulting re-encryption ciphertext  $CT'$ , then  $(pk', CT')$  is a derivative of  $(pk, CT)$ .
4. If  $\mathcal{A}$  has issued a re-encryption key generation query  $\langle pk, pk' \rangle$  (or  $\langle pk', pk \rangle$  if  $\Pi$  is bidirectional PRE scheme) to obtain the re-encryption key  $rk$ , and  $CT' = \text{ReEncrypt}(rk, CT)$ , then  $(pk', CT')$  is a derivative of  $(pk, CT)$ .

**Master Secret Security.** In [1], Ateniese *et al.* defined another security notion, named *master secret security*, for unidirectional PRE schemes. This security notion catches the intuition that, even if the dishonest proxy colludes with the delegatee, it is still impossible for them to derive the delegator’s secret key. Similarly, we can define this security notion for a C-PRE scheme  $\Pi$  via the following game:

**Setup.** Challenger  $\mathcal{C}$  runs algorithm  $\text{GlobalSetup}(1^\kappa)$  and gives the global parameters  $\text{param}$  to  $\mathcal{A}$ .

**Queries.**  $\mathcal{A}$  adaptively issues queries  $q_1, \dots, q_m$  where query  $q_i$  is one of the following:

- *Uncorrupted key generation query*  $\langle i \rangle$ :  $\mathcal{C}$  first runs algorithm  $\text{KeyGen}(i)$  to obtain a public/secret key pair  $(pk_i, sk_i)$ , and then sends  $pk_i$  to  $\mathcal{A}$ .
- *Corrupted key generation query*  $\langle j \rangle$ :  $\mathcal{C}$  first runs algorithm  $\text{KeyGen}(j)$  to obtain a public/secret key pair  $(pk_j, sk_j)$ , and then gives  $(pk_j, sk_j)$  to  $\mathcal{A}$ .
- *Re-encryption key query*  $\langle pk_i, pk_j \rangle$ :  $\mathcal{C}$  runs algorithm  $\text{ReKeyGen}(sk_i, pk_j)$  to generate a re-encryption key  $rk_{i,j}$  and returns it to  $\mathcal{A}$ . Here  $sk_i$  is the secret key with respect to  $pk_i$ . It is required that  $pk_i$  and  $pk_j$  were generated beforehand by algorithm  $\text{KeyGen}$ .

**Output.** Finally,  $\mathcal{A}$  outputs a secret key  $sk_{i^*}$  with respect to the public key  $pk_{i^*}$ .  $\mathcal{A}$  wins the game if  $sk_{i^*}$  is indeed a valid secret key and  $\mathcal{A}$  has never issue the corrupted key generation query on  $\langle i^* \rangle$  (i.e.,  $\mathcal{A}$  issue the uncorrupted key generation query on  $\langle i^* \rangle$ ).

We refer to the above adversary  $\mathcal{A}$  as a MSK-PRE adversary, and define his advantage in attacking  $\Pi$ ’s master secret security as  $\text{Adv}_{\Pi, \mathcal{A}}^{\text{MSK-PRE}} = \Pr[\mathcal{A} \text{ wins}]$ , where the probability is taken over the random coins consumed by the challenger and the adversary.

**Definition 2** *We say that a C-PRE scheme  $\Pi$  is  $(t, q_u, q_c, q_{rk}, \epsilon)$ -MSK-PRE secure, if for any  $t$ -time MSK-PRE adversary  $\mathcal{A}$  that makes at most  $q_u$  uncorrupted key generation queries, at most  $q_c$  corrupted key generation queries and at most  $q_{rk}$  re-encryption key queries, we have  $\text{Adv}_{\Pi, \mathcal{A}}^{\text{MSK-PRE}} \leq \epsilon$ .*

### 2.3 Complexity Assumptions

In this subsection, we review some related complexity assumptions which are used in the security proofs for our schemes.

The security of our bidirectional scheme is based on a variant of the CDH problem named modified computational Diffie-Hellman (mCDH) problem, which has been recently used to construct multi-use unidirectional proxy re-signatures [21].

**Definition 3** *Let  $\mathbb{G}$  be a cyclic multiplicative group with prime order  $q$ . The mCDH problem in group  $\mathbb{G}$  is, given a tuple  $(g, g^{\frac{1}{a}}, g^a, g^b) \in \mathbb{G}^4$  with unknown  $a, b \xleftarrow{\$} \mathbb{Z}_q^*$ , to compute  $g^{ab}$ .*

**Definition 4** *For a polynomial-time adversary  $\mathcal{B}$ , we define his advantage in solving the mCDH problem in group  $\mathbb{G}$  as*

$$\text{Adv}_{\mathcal{B}}^{\text{mCDH}} \triangleq \Pr \left[ \mathcal{B}(g, g^{\frac{1}{a}}, g^a, g^b) = g^{ab} \right],$$

where the probability is taken over the random choices of  $a, b$  and the random bits consumed by  $\mathcal{B}$ . We say that the  $(t, \epsilon)$ -mCDH assumption holds in group  $\mathbb{G}$  if no  $t$ -time adversary  $\mathcal{B}$  has advantage at least  $\epsilon$  in solving the mCDH problem in group  $\mathbb{G}$ .

The chosen-ciphertext security of our unidirectional PRE scheme is based on the CDH assumption.

**Definition 5** Let  $\mathbb{G}$  be a cyclic multiplicative group with prime order  $q$ . The CDH problem in group  $\mathbb{G}$  is, given a tuple  $(g, g^a, g^b) \in \mathbb{G}^3$  with unknown  $a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ , to compute  $g^{ab}$ .

**Definition 6** For a polynomial-time adversary  $\mathcal{B}$ , we define his advantage in solving the CDH problem in group  $\mathbb{G}$  as

$$\text{Adv}_{\mathcal{B}}^{\text{CDH}} \triangleq \Pr \left[ \mathcal{B}(g, g^a, g^b) = g^{ab} \right],$$

where the probability is taken over the random choices of  $a, b$  and the random bits consumed by  $\mathcal{B}$ . We say that the  $(t, \epsilon)$ -CDH assumption holds in group  $\mathbb{G}$  if no  $t$ -time adversary  $\mathcal{B}$  has advantage at least  $\epsilon$  in solving the CDH problem in group  $\mathbb{G}$ .

Bao *et al.* [5] introduced a variant of the CDH problem named divisible computation Diffie-Hellman (DCDH) problem. The DCDH problem in group  $\mathbb{G}$  is, given  $(g, g^{\frac{1}{a}}, g^b) \in \mathbb{G}^3$  with unknown  $a, b \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ , to compute  $g^{ab}$ . In [5], Bao *et al.* presented the relation between CDH problem and DCDH problem in the following lemma:

**Lemma 1** The DCDH problem in group  $\mathbb{G}$  is equivalent to the CDH problem in the same group.

The master secret security of our scheme is based on the discrete logarithm (DL) problem.

**Definition 7** The DL problem in group  $\mathbb{G}$  is, given a tuple  $(g, g^a) \in \mathbb{G}^2$  with unknown  $a$ , to compute  $a$ .

**Definition 8** For a polynomial-time algorithm  $\mathcal{B}$ , we define his advantage in solving the DL problem in group  $\mathbb{G}$  as  $\Pr[\mathcal{B}(g, g^a) = a]$ , where the probability is taken over the random choices of  $a$  in  $\mathbb{Z}_q$ , the random choice of  $g$  in  $\mathbb{G}$ , and the random bits consumed by  $\mathcal{B}$ . We say that the  $(t, \epsilon)$ -DL assumption holds in group  $\mathbb{G}$ , if no  $t$ -time adversary  $\mathcal{B}$  has advantage at least  $\epsilon$  in solving the DL problem in group  $\mathbb{G}$ .

### 3 Proposed Bidirectional Proxy Re-Encryption Scheme

In this section, we first describe the main idea of our bidirectional PRE scheme, and then propose the concrete construction. A comparison between our scheme and other bidirectional PRE schemes is also given in this section.

#### 3.1 Main Idea

The idea behind our construction begins with the CCA-secure “hashed” ElGamal encryption scheme [8, 13, 14] given in Figure 1. It is important to note that, in the ciphertext component  $F = H_2(pk^r) \oplus (m \parallel \omega)$ , the public key  $pk$  is embedded in the hash function  $H_2$  and masked by  $(m \parallel \omega)$ . This frustrates the proxy to re-encrypt the ciphertext, and hence this original scheme can not be directly used for our PRE scheme. To circumvent this obstacle, we slightly modify the scheme as shown in Figure 2 (see the bolded parts). Now,

Setup( $\kappa$ ):	Encrypt( $pk, m$ ):	Decrypt( $(E, F), sk$ ):
$x \xleftarrow{\$} \mathbb{Z}_q^*$ ; $pk = g^x$ ; $sk = x$ Return $(pk, sk)$	$\omega \xleftarrow{\$} \{0, 1\}^{l_1}$ ; $r = H_1(m, \omega)$ $E = g^r$ ; $F = H_2(pk^r) \oplus (m \parallel \omega)$ Return CT = $(E, F)$	$m \parallel \omega = F \oplus H_2(E^{sk})$ If $E = g^{H_1(m, \omega)}$ return $m$ Else return $\perp$

**Note:**  $H_1$  and  $H_2$  are hash functions such that  $H_1 : \{0, 1\}^{l_0} \times \{0, 1\}^{l_1} \rightarrow \mathbb{Z}_q^*$ ,  $H_2 : \mathbb{G} \rightarrow \{0, 1\}^{l_0+l_1}$ .

The message space is  $\mathcal{M} = \{0, 1\}^{l_0}$ .

Figure 1: CCA-secure “hashed” ElGamal encryption scheme

Setup( $\kappa$ ):	Encrypt( $pk, m$ ):	Decrypt( $(E, F), sk$ ):
$x \xleftarrow{\$} \mathbb{Z}_q^*$ ; $pk = g^x$ ; $sk = x$ Return $(pk, sk)$	$\omega \xleftarrow{\$} \{0, 1\}^{l_1}$ ; $r = H_1(m, \omega)$ $E = \mathbf{pk}^r$ ; $F = H_2(\mathbf{g}^r) \oplus (m \parallel \omega)$ Return CT = $(E, F)$	$m \parallel \omega = F \oplus H_2(\mathbf{E}^{\frac{1}{sk}})$ If $\mathbf{E} = \mathbf{pk}^{H_1(m, \omega)}$ return $m$ Else return $\perp$

Figure 2: Modified CCA-secure “hashed” ElGamal encryption scheme

the ciphertext component  $F$  does not involve the public key, and the ciphertext component  $E = pk^r = g^{xr}$  can be successfully re-encrypted into another ciphertext component  $E' = E^{\frac{y}{x}} = g^{yr}$  (under the public key  $pk' = g^y$ ) using the re-encryption key  $rk_{x,y} = \frac{y}{x}$ .

Indeed, the modified scheme can achieve the chosen-ciphertext security as a traditional public key encryption. However, it does not satisfy the chosen-ciphertext security for proxy re-encryptions. To explain more clearly, let’s take the following attack as an example:

Suppose  $\mathcal{A}$  is given a challenged ciphertext under a target public key  $pk_{i^*} = g^x$ , say  $\text{CT}^* = (E^*, F^*) = (g^{xr^*}, H_2(g^{r^*}) \oplus (m_\delta \parallel \omega^*))$ . Then adversary  $\mathcal{A}$  can win the IND-PRE-CCA game as follows: He first picks  $z \xleftarrow{\$} \{0, 1\}^{l_0+l_1}$ , and modifies the challenged ciphertext to get a new, although invalid, ciphertext  $\text{CT}' = (E', F') = (E^*, F^* \oplus z) = (g^{xr^*}, H_2(g^{r^*}) \oplus (m_\delta \parallel \omega^*) \oplus z)$ . Next, he issues a corrupted key generation query to obtain a public/secret key pair  $(pk', sk') = (g^y, y)$ , and then issues a re-encryption query to obtain a re-encrypt ciphertext, say  $\text{CT}'' = (E'', F'') = (g^{yr^*}, H_2(g^{r^*}) \oplus (m_\delta \parallel \omega^*) \oplus z)$ , under the public key  $pk' = g^y$ . Finally, using the secret key  $sk' = y$ ,  $\mathcal{A}$  can recover  $(m_\delta \parallel \omega^*)$  as  $(m_\delta \parallel \omega^*) = F'' \oplus H_2((E'')^{\frac{1}{y}}) \oplus z$ , and eventually obtain the bit  $\delta$ . Note that according to the constraints described in the IND-PRE-CCA game, it is legal for  $\mathcal{A}$  to issue the above queries. As a consequence, he wins the IND-PRE-CCA game.

The above attack succeeds due to the fact that, the validity of second-level ciphertexts can only be checked by the decryptor, not any other parties including the proxy. So, to achieve the IND-PRE-CCA security for a PRE scheme, the proxy must be able to check the validity of second-level ciphertexts. Furthermore, since a PRE scheme requires the proxy to re-encrypt ciphertexts *without* seeing the plaintexts, the validity of second-level ciphertexts must be publicly verifiable. It is worth noting that, it is not an easy job to construct a CCA-secure PRE scheme with public verifiability and yet without pairings, e.g., the existing CCA-secure PRE schemes achieve the public verifiability by resorting to bilinear pairings.

In this paper, we achieve this goal by resorting to the Schnorr signature scheme [25], which is given in Figure 3. Note that it is non-trivial to incorporate the Schnorr signature scheme into the modified ElGamal encryption scheme to obtain a secure PRE scheme. One may think that, it can be done by choosing a signing/verification key pair  $(vk_s, sk_s)$ , signing the ciphertext CT to obtain a signature  $\sigma$ , and publishing  $(vk_s, \text{CT}, \sigma)$  as the final ciphertext. Unfortunately, this does not work, since the adversary can still harmfully maul



the above ciphertext. Namely, he can choose another signing/verification key pair to sign the ciphertext component CT, and then obtain another valid ciphertext. The problem lies in the *loose* integration between the ciphertext component CT and the signature  $\sigma$ .

Setup( $\kappa$ ):	Sign( $sk, m$ ):	Verify( $pk, (e, s), m$ ):
$x \xleftarrow{\$} \mathbb{Z}_q^*$ ; $pk = g^x$ ; $sk = x$ Return $(pk, sk)$	$u \xleftarrow{\$} \mathbb{Z}_q^*$ ; $D = g^u$ $e = H(m, D)$ ; $s = (u + sk \cdot e) \bmod q$ Return $\sigma = (e, s)$	$D_v = g^s pk^{-e}$ ; $e_v = H(m, D_v)$ If $e = e_v$ return 1 Else return 0

**Note:**  $H$  is a hash function such that  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ .

Figure 3: Schnorr signature scheme

We here briefly explain how to *tightly* integrate the Schnorr signature scheme with the modified ElGamal encryption scheme to obtain our PRE scheme. To do so, we first slightly modify the Schnorr signature scheme as shown in Figure 4 (see the bolded parts). Next, given the ciphertext components  $(E, F) = (pk^r, H_2(g^r) \oplus (m \parallel \omega))$ , to tightly integrate  $(E, F)$  with the Schnorr signature, we generate the Schnorr signature as follows: Viewing  $F$  as the message to be signed, and  $(E, r) = (pk^r, r)$  as the verification/signing key pair (here the base  $pk$  in  $pk^r$  is similarly viewed as the base  $g$  in  $g^x$ ), we pick  $u \xleftarrow{\$} \mathbb{Z}_q^*$  and output the signature as  $(D, s) = (pk^u, u + rH_3(D, E, F))$ . The final ciphertext is  $(D, E, F, s)$ .

Setup( $\kappa$ ):	Sign( $sk, m$ ):	Verify( $pk, (D, s), m$ ):
$x \xleftarrow{\$} \mathbb{Z}_q^*$ ; $pk = g^x$ ; $sk = x$ Return $(pk, sk)$	$u \xleftarrow{\$} \mathbb{Z}_q^*$ ; $D = g^u$ $e = H(m, D)$ ; $s = (u + sk \cdot e) \bmod q$ Return $\sigma = (D, s)$	If $g^s = D \cdot pk^{H(m, D)}$ return 1 Else return 0

Figure 4: Modified Schnorr signature scheme

Next, we explain how to realize our re-encryption algorithm. We first present an unsuccessful solution: Suppose the proxy wants to re-encrypt a ciphertext  $CT = (D, E, F, s)$  under public key  $pk_i = g^{x_i}$  into another one under public key  $pk_j = g^{x_j}$ . The proxy first checks  $pk_i^s \stackrel{?}{=} D \cdot E^{H_3(D, E, F)}$  to ensure the validity of the ciphertext, and then outputs  $CT_j = (E', F) = (E^{x_j/x_i}, F)$  as the re-encrypted ciphertext (here  $x_j/x_i$  is the re-encryption key). At first glance, this solution appears to be successful. Unfortunately, this is not true, since there exists a simple attack: given a challenged ciphertext  $CT_i = (D, E, F, s)$  under a target public key  $pk_i$ , the adversary  $\mathcal{A}$  simply views  $(E, F)$  as a re-encrypted ciphertext under  $pk_i$ , and issues a decryption query on  $\langle pk_i, (E, F) \rangle$ . Note that according to the IND-PRE-CCA game, it is legal for  $\mathcal{A}$  to issue such a query. So, the adversary will be given the plaintext  $m_\delta$ , and hence breaks the challenge  $\delta$ . The problem behind the above solution is that, the re-encrypted ciphertext has the same form as some components of the original ciphertext, and the decryption policies for them are also identical. Therefore, given an original ciphertext under a public key, the adversary can simply take some components from the original ciphertext and obtain a *legal* re-encrypted ciphertext under the same public key, and then easily break the chosen-ciphertext security of the scheme.

To resist the above attack, a trivial solution is defining the re-encrypted ciphertext to be  $CT_j = (D, E, F, s, E')$  instead of  $CT_j = (E', F)$ . However, such a solution is not desirable, since it introduces *big* ciphertext overhead. Below, we give an efficient solution with short re-encrypted ciphertext.

To re-encrypt a ciphertext  $CT = (D, E, F, s)$  from public key  $pk_i = g^{x_i}$  to  $pk_j = g^{x_j}$ ,

the proxy first checks  $pk_i^s \stackrel{?}{=} D \cdot E^{H_3(D,E,F)}$  to ensure the validity of the ciphertext, and then outputs  $CT_j = (E', F') = (E^{\frac{x_j}{x_i}}, F \oplus H_4(E', g^{\frac{x_i}{x_j}}))$  as the re-encrypted ciphertext. The decryption algorithm for re-encrypted ciphertexts should be accordingly modified. To decrypt the re-encrypted ciphertext  $CT_j = (E', F')$ , the decryptor with secret key  $x_j$  works as follows: compute  $m||\omega = F' \oplus H_2(E'^{\frac{1}{x_j}}) \oplus H_4(E', (g^{x_i})^{\frac{1}{x_j}})$ , and returns  $m$  if  $E' = (g^{x_j})^{H_1(m,\omega)}$  holds and  $\perp$  otherwise. Now, the re-encrypted ciphertext (i.e.,  $(E', F')$ ) and the components (i.e.,  $(E, F)$ ) in the original ciphertext have different forms, and the decryption policies for them are also distinct. However, there still exists an attack mounted by the proxy: given a challenged ciphertext  $CT_i = (D, E, F, s)$  under public key  $pk_i = x_i$ , a proxy with a re-encrypted key  $rk_{i,j} = \frac{x_j}{x_i}$  first computes  $(E, F') = (E, F \oplus H_4(E, g^{\frac{x_j}{x_i}}))$ . It can be seen that the resulting  $(E, F')$  can be viewed as a *valid* re-encrypted ciphertext encrypted from  $pk_j$  to  $pk_i$ . Then, the proxy issues a decryption query on  $\langle pk_i, (E, F') \rangle$  and then breaks the challenge. To resist this attack, we modify the computation of  $r$  in the encryption algorithm as  $r = H_1(m, \omega, pk)$ , and the original public key will be implicitly embedded in the re-encrypted ciphertext. We will present the detailed construction of our PRE scheme in the next subsection, and give a formal security proof for the scheme in Section 3.4.

### 3.2 Proposed Scheme: $\Pi_{\mathbf{B}}$

We now present the detailed construction of our bidirectional PRE scheme. For notational simplicity, we denote this bidirectional PRE scheme by  $\Pi_{\mathbf{B}}$ . The proposed scheme consists of the following algorithms:

**GlobalSetup**( $\kappa$ ): Given a security parameter  $\kappa$ , choose two big primes  $p$  and  $q$  such that  $q|p-1$  and the bit-length of  $q$  is  $\kappa$ . Let  $g$  be a generator of group  $\mathbb{G}$ , which is a subgroup of  $\mathbb{Z}_q^*$  with order  $q$ . Besides, choose four hash functions  $H_1, H_2, H_3$  and  $H_4$  such that  $H_1 : \{0, 1\}^{l_0} \times \{0, 1\}^{l_1} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$ ,  $H_2 : \mathbb{G} \rightarrow \{0, 1\}^{l_0+l_1}$ ,  $H_3 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$  and  $H_4 : \mathbb{G} \times \mathbb{G} \rightarrow \{0, 1\}^{l_0+l_1}$ . Here  $l_0$  and  $l_1$  are security parameters, and the message space is  $\{0, 1\}^{l_0}$ . The global parameters are

$$param = (q, \mathbb{G}, g, H_1, H_2, H_3, H_4, l_0, l_1).$$

**KeyGen**( $i$ ): To generate the public/secret key pair for user  $i$ , this key generation algorithm picks a random  $x_i \xleftarrow{\$} \mathbb{Z}_q^*$ , and then sets  $pk_i = g^{x_i}$  and  $sk_i = x_i$ .

**ReKeyGen**( $sk_i, sk_j$ ): On input two secret keys  $sk_i = x_i$  and  $sk_j = x_j$ , this algorithm outputs the bidirectional re-encryption key  $rk_{i,j} = x_j/x_i \pmod q$ .

**Encrypt**( $pk, m$ ): On input a public key  $pk$  and a plaintext  $m \in \{0, 1\}^{l_0}$ , this algorithm works as below:

1. Pick  $u \xleftarrow{\$} \mathbb{Z}_q^*$ ,  $\omega \xleftarrow{\$} \{0, 1\}^{l_1}$ , and compute  $r = H_1(m, \omega, pk)$ .
2. Compute  $D = pk^u$ ,  $E = pk^r$ ,  $F = H_2(g^r) \oplus (m||\omega)$ ,  $s = u + r \cdot H_3(D, E, F) \pmod q$ .
3. Output the ciphertext  $CT = (D, E, F, s)$ .

**ReEncrypt**( $rk_{i,j}, CT_i, pk_j$ ): On input a re-encryption key  $rk_{i,j} = \frac{x_j}{x_i}$ , a second-level ciphertext  $CT_i$  under public key  $pk_i$ , this algorithm re-encrypts this ciphertext under public key  $pk_j$  as follows:

1. Parse  $CT_i$  as  $CT_i = (D, E, F, s)$ .

2. Check whether  $pk_i^s = D \cdot E^{H_3(D,E,F)}$  holds. If not, output  $\perp$ .
3. Otherwise, compute  $E' = E^{rk_{i,j}} = g^{(r \cdot x_i) \cdot x_j / x_i} = g^{r \cdot x_j}$ ,  $F' = F \oplus H_4(E', g^{x_i/x_j})$ , and output the first-level ciphertext  $CT_j = (pk_i, E', F')$ .

**Decrypt**(CT,  $sk$ ): On input a secret key  $sk = x$  and ciphertext CT, this algorithm works according to two cases:

- CT is a second-level ciphertext  $CT = (D, E, F, s)$ : If  $(g^x)^s = D \cdot E^{H_3(D,E,F)}$  does not hold, output  $\perp$ , else compute  $m \parallel \omega = F \oplus H_2(E^{\frac{1}{x}})$ , and return  $m$  if  $E = (g^x)^{H_1(m, \omega, g^x)}$  holds and  $\perp$  otherwise.
- CT is a first-level ciphertext  $CT = (pk_i, E', F')$ : Recall that we only concentrate on the single-hop scheme, hence  $pk_i$  should be different from the original public key  $g^x$ . To decrypt this ciphertext, first compute  $m \parallel \omega = F' \oplus H_2(E'^{\frac{1}{x}}) \oplus H_4(E', pk_i^{1/x})$ . If  $E' = (g^x)^{H_1(m, \omega, pk_i)}$  holds return  $m$ ; otherwise return  $\perp$ .

### 3.3 Comparison

In this subsection, we provide a comparison of our scheme  $\Pi_{\mathbb{B}_i}$  with other existing bidirectional PRE schemes. To conduct a fair comparison, we choose Canetti and Hohenberger's PRE schemes [10], which are also bidirectional and achieve chosen-ciphertext security. Two PRE schemes are presented in [10], including one secure in the random oracle model (referred to as CH Scheme I) and another one secure in the standard model (referred to as CH Scheme II). Table 1 gives a comparison between our scheme  $\Pi_{\mathbb{B}_i}$  and these two schemes. The comparison results indicate that our scheme is much more efficient than the other two schemes. For example, the encryption in CH Scheme I needs 4 exponentiations, 1 pairing and 1 one-time signature signing, while the encryption in our scheme  $\Pi_{\mathbb{B}_i}$  involves only 3 exponentiations. It's worth pointing out that, the computational cost and the ciphertext size in our scheme *decrease* with re-encryption, while those in CH Schemes I and II remain unchanged. Note that the computational cost and the ciphertext in some schemes such as [1, 2, 11, 20] *increase* with re-encryption. Although the ciphertext in our scheme involves less group elements than that in CH Schemes I and II, we do not claim that our ciphertext is shorter than theirs, since their schemes are implemented in the bilinear group which enables shorter representation of a group element. However, the pairings in bilinear group in turn add heavy computational overhead to their schemes. Both our scheme and CH Scheme I are provably secure in the random oracle model, while CH Scheme II is proved secure without random oracles.

### 3.4 Security Analysis

In this subsection, we prove the IND-PRE-CCA security for our scheme  $\Pi_{\mathbb{B}_i}$  in the random oracle model.

**Theorem 1** *Our PRE scheme  $\Pi_{\mathbb{B}_i}$  is IND-PRE-CCA secure in the random oracle model, assuming the mCDH assumption holds in group  $\mathbb{G}$  and the Schnorr signature is existential unforgeable against chosen message attack (EUF-CMA). Concretely, if there exists an adversary  $\mathcal{A}$ , who asks at most  $q_{H_i}$  random oracle queries to  $H_i$  with  $i \in \{1, \dots, 4\}$ , and*

---

<sup>3</sup>In Table 1, we neglect some operations such as hash function evaluation, modular multiplication and XOR, since the computational cost of these operations is far less than that of exponentiations or pairings. Note that, using the technique in [9, 18, 19], both the re-encryption and decryption in CH scheme I and II can further save two pairings, at the cost of several exponentiation operations.

Schemes		CH Scheme I	CH Scheme II	Our $\Pi_{\mathbb{B}_i}$
Comput.	Encrypt	$1t_p + 4t_e + 1t_s$	$1t_p + 3t_e + 1t_{me} + 1t_s$	$3t_e$
	Re-Encrypt	$4t_p + 1t_e + 1t_v$	$4t_p + 2t_e + 1t_v$	$4t_e$
Cost	Decrypt	2nd-level CiphTxt	$5t_p + 1t_e + 1t_v$	$4t_e$
		1st-level CiphTxt	$5t_p + 1t_e + 1t_v$	$3t_e$
CiphTxt Length	2nd-level CiphTxt	$1 pk_s +3 \mathbb{G}_e +1 \mathbb{G}_T +1 \sigma_s $	$1 pk_s +3 \mathbb{G}_e +1 \mathbb{G}_T +1 \sigma_s $	$3 \mathbb{G} +1 \mathbb{Z}_q $
	1st-level CiphTxt	$1 pk_s +3 \mathbb{G}_e +1 \mathbb{G}_T +1 \sigma_s $	$1 pk_s +3 \mathbb{G}_e +1 \mathbb{G}_T +1 \sigma_s $	$3 \mathbb{G} $
Without Random Oracles?		×	✓	×
Underlying Assumptions		DBDH	DBDH	mCDH

**Note:**  $t_p$ ,  $t_e$  and  $t_{me}$  represent the computational cost of a bilinear pairing, an exponentiation and a multi-exponentiation respectively, while  $t_s$  and  $t_v$  represent the computational cost of a one-time signature signing and verification respectively.  $|\mathbb{G}|$ ,  $|\mathbb{Z}_q|$ ,  $|\mathbb{G}_e|$  and  $|\mathbb{G}_T|$  denote the bit-length of an element in groups  $\mathbb{G}$ ,  $\mathbb{Z}_q$ ,  $\mathbb{G}_e$  and  $\mathbb{G}_T$  respectively. Here  $\mathbb{G}$  and  $\mathbb{Z}_q$  denote the groups used in our scheme, while  $\mathbb{G}_e$  and  $\mathbb{G}_T$  are the bilinear groups used in CH scheme I and II, i.e., the bilinear pairing is  $e : \mathbb{G}_e \times \mathbb{G}_e \rightarrow \mathbb{G}_T$ . Finally,  $|pk_s|$  and  $|\sigma_s|$  denote the bit length of the one-time signature's public key and a one-time signature respectively.

Table 1: Efficiency Comparison between Canetti-Hohenberger PRE Schemes and Our Scheme  $\Pi_{\mathbb{B}_i}$ <sup>3</sup>

breaks the  $(t, q_u, q_c, q_{rk}, q_{re}, q_d, \epsilon)$ -IND-PRE-CCA security of our scheme  $\Pi_{\mathbb{B}_i}$ , then, for any  $0 < \nu < \epsilon$ , there exists

- either an algorithm  $\mathcal{B}$  which can solve the  $(t', \epsilon')$ -mCDH problem in  $\mathbb{G}$  with

$$\begin{aligned}
t' &\leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\
&\quad + (q_u + q_c + 4q_{re} + 3q_d + (2q_d + q_{re})q_{H_1})t_e, \\
\epsilon' &\geq \frac{1}{q_{H_2}} \left( 2(\epsilon - \nu) - \frac{q_{H_1} + (q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} - \frac{q_{re} + 2q_d}{q} \right),
\end{aligned}$$

where  $t_e$  denotes the running time of an exponentiation in  $\mathbb{G}$ ;

- or an attacker who breaks the EUF-CMA security of the Schnorr signature with advantage  $\nu$  within time  $t'$ .

**Proof.** Without loss of generality, we assume that the Schnorr signature is  $(t', \nu)$ -EUF-CMA secure for some probability  $0 < \nu < \epsilon$ . Suppose there exists a  $t$ -time adversary  $\mathcal{A}$  who can break the IND-PRE-CCA security of scheme  $\Pi_{\mathbb{B}_i}$  with advantage  $\epsilon - \nu$ . Then we show how to construct an algorithm  $\mathcal{B}$  which can solve the  $(t', \epsilon')$ -mCDH problem in group  $\mathbb{G}$ .

Suppose  $\mathcal{B}$  is given as input an mCDH challenge tuple  $(g, g^{\frac{1}{a}}, g^a, g^b) \in \mathbb{G}^4$  with unknown  $a, b \xleftarrow{\$} \mathbb{Z}_q^*$ . Algorithm  $\mathcal{B}$ 's goal is to output  $g^{ab}$ . Algorithm  $\mathcal{B}$  acts as the challenger and plays the IND-PRE-CCA game with adversary  $\mathcal{A}$  in the following way.

**Setup.** Algorithm  $\mathcal{B}$  gives  $(q, \mathbb{G}, g, H_1, H_2, H_3, H_4, l_0, l_1)$  to  $\mathcal{A}$ . Here  $H_1, H_2, H_3$  and  $H_4$  are random oracles controlled by  $\mathcal{B}$ .

**Hash Oracle Queries.** At any time adversary  $\mathcal{A}$  can issue the random oracle queries  $H_1, H_2, H_3$  and  $H_4$ . Algorithm  $\mathcal{B}$  maintains four hash lists  $H_1^{\text{list}}, H_2^{\text{list}}, H_3^{\text{list}}$  and  $H_4^{\text{list}}$  which are initially empty, and responds as below:

- $H_1$  queries: On receipt of an  $H_1$  queries on  $(m, \omega, pk)$ , if this query has appeared on the  $H_1^{\text{list}}$  in a tuple  $(m, \omega, pk, r)$ , return the predefined value  $r$  as the result of the query. Otherwise, choose  $r \xleftarrow{\$} \mathbb{Z}_q^*$ , add the tuple  $(m, \omega, pk, r)$  to the list  $H_1^{\text{list}}$  and respond with  $H_1(m, \omega, pk) = r$ .

- $H_2$  queries: On receipt of an  $H_2$  query  $R \in \mathbb{G}$ , if this query has appeared on the  $H_2^{\text{list}}$  in a tuple  $(R, \beta)$ , return the predefined value  $\beta$  as the result of the query. Otherwise, choose  $\beta \xleftarrow{\$} \{0, 1\}^{l_0+l_1}$ , add the tuple  $(R, \beta)$  to the list  $H_2^{\text{list}}$  and respond with  $H_2(R) = \beta$ .
- $H_3$  queries: On receipt of an  $H_3$  query  $(D, E, F)$ , if this query has appeared on the  $H_3^{\text{list}}$  in a tuple  $(D, E, F, \gamma)$ , return the predefined value  $\gamma$  as the result of the query. Otherwise, choose  $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$ , add the tuple  $(D, E, F, \gamma)$  to the list  $H_3^{\text{list}}$  and respond with  $H_3(D, E, F) = \gamma$ .
- $H_4$  queries: On receipt of an  $H_4$  query  $(E', U)$ , if this query has appeared on the  $H_4^{\text{list}}$  in a tuple  $(E', U, \lambda)$ , return the predefined value  $\lambda$  as the result of the query. Otherwise, choose  $\lambda \xleftarrow{\$} \{0, 1\}^{l_0+l_1}$ , add the tuple  $(E', U, \lambda)$  to the list  $H_4^{\text{list}}$  and respond with  $H_4(E', U) = \lambda$ .

**Phase 1.** In this phase, adversary  $\mathcal{A}$  issues a series of queries as in the definition of the IND-PRE-CCA game.  $\mathcal{B}$  maintains a list  $K^{\text{list}}$  which is initially empty, and answers these queries for  $\mathcal{A}$  as follows:

- *Uncorrupted key generation query*  $\langle i \rangle$ . Algorithm  $\mathcal{B}$  first picks  $x_i \xleftarrow{\$} \mathbb{Z}_q^*$  and defines  $pk_i = (g^{1/a})^{x_i}$ ,  $c_i = 0$ . Next, it adds the tuple  $(pk_i, x_i, c_i)$  to  $K^{\text{list}}$  and returns  $pk_i$  to adversary  $\mathcal{A}$ . Here the bit  $c_i$  is used to denote whether the secret key with respect to  $pk_i$  is corrupted, i.e.,  $c_i = 0$  indicates uncorrupted and  $c_i = 1$  means corrupted.
- *Corrupted key generation query*  $\langle j \rangle$ . Algorithm  $\mathcal{B}$  first picks  $x_j \xleftarrow{\$} \mathbb{Z}_q^*$  and defines  $pk_j = g^{x_j}$ ,  $c_j = 1$ . Next, it adds the tuple  $(pk_j, x_j, c_j)$  to  $K^{\text{list}}$  and returns  $(pk_j, x_j)$  to adversary  $\mathcal{A}$ .
- *Re-encryption key generation query*  $\langle pk_i, pk_j \rangle$ : Recall that according to the definition of IND-PRE-CCA game, it is required that  $pk_i$  and  $pk_j$  were generated beforehand, and either both of them are corrupted or alternately both are uncorrupted. Algorithm  $\mathcal{B}$  first recovers tuples  $(pk_i, x_i, c_i)$  and  $(pk_j, x_j, c_j)$  from  $K^{\text{list}}$ , and then returns the re-encryption key  $x_j/x_i$  to  $\mathcal{A}$ .
- *Re-encryption query*  $\langle pk_i, pk_j, \text{CT}_i (= (D, E, F, s)) \rangle$ : If  $pk_i^s \neq D \cdot E^{H_3(D, E, F)}$ , then output  $\perp$ . Otherwise, algorithm  $\mathcal{B}$  responds to this query as follows:
  1. Recover tuples  $(pk_i, x_i, c_i)$  and  $(pk_j, x_j, c_j)$  from  $K^{\text{list}}$ .
  2. If  $c_i = c_j$ , compute  $E' = E^{x_j/x_i}$ ,  $F' = F \oplus H_4(E', g^{x_i/x_j})$  and return  $(E', F')$  as the first-level ciphertext to  $\mathcal{A}$ .
  3. Else, search whether there exists a tuple  $(m, \omega, pk_i, r) \in H_1^{\text{list}}$  such that  $pk_i^r = E$ . If there exists no such tuple, return  $\perp$ . Otherwise, first compute  $E' = pk_i^r$ . Next, if  $c_i = 1 \wedge c_j = 0$ , define  $F' = F \oplus H_4(E', g^{\frac{x_i a}{x_j}})$ ; else if  $c_i = 0 \wedge c_j = 1$ , define  $F' = F \oplus H_4(E', g^{\frac{x_i}{a x_j}})$ . Finally, return  $(E', F')$  as the first-level ciphertext to  $\mathcal{A}$ .
- *Decryption query*  $\langle pk, \text{CT} \rangle$ : Algorithm  $\mathcal{B}$  first recovers tuple  $(pk, x, c)$  from list  $K^{\text{list}}$ . If  $c = 1$ , algorithm  $\mathcal{B}$  runs  $\text{Decrypt}(\text{CT}, x)$  and returns the result to  $\mathcal{A}$ . Otherwise, algorithm  $\mathcal{B}$  works according to the following two cases:

- CT is a second-level ciphertext  $CT = (D, E, F, s)$ : If  $pk^s \neq D \cdot E^{H_3(D, E, F)}$ , return  $\perp$  to  $\mathcal{A}$ . Otherwise, search lists  $H_1^{\text{list}}$  and  $H_2^{\text{list}}$  to see whether there exist  $(m, \omega, pk, r) \in H_1^{\text{list}}$  and  $(R, \beta) \in H_2^{\text{list}}$  such that

$$pk^r = E, \beta \oplus (m \parallel \omega) = F \quad \text{and} \quad R = g^r.$$

If yes, return  $m$  to  $\mathcal{A}$ . Otherwise, return  $\perp$ .

- CT is a first-level ciphertext  $CT = (pk'', E', F')$ : Algorithm  $\mathcal{B}$  acts as follows:
  1. Recover tuples  $(pk, x, c)$  and  $(pk'', x'', c'')$  from  $K^{\text{list}}$ .
  2. Define  $U$  according to the following three cases:
    - \* If  $c = c''$ : Define  $U = g^{\frac{x''}{x}}$ ;
    - \* If  $c = 0 \wedge c'' = 1$ : Define  $U = g^{\frac{x'' \cdot a}{x}}$ ;
    - \* If  $c = 1 \wedge c'' = 0$ : Define  $U = g^{\frac{x}{ax}}$ .
  3. search lists  $H_1^{\text{list}}$  and  $H_2^{\text{list}}$  to see whether there exist  $(m, \omega, pk, r) \in H_1^{\text{list}}$  and  $(R, \beta) \in H_2^{\text{list}}$  such that

$$pk^r = E', \beta \oplus (m \parallel \omega) \oplus H_4(E', U) = F' \quad \text{and} \quad R = g^r.$$

If yes, return  $m$  to  $\mathcal{A}$ . Otherwise, return  $\perp$ .

**Challenge.** When  $\mathcal{A}$  decides that Phase 1 is over, it outputs a target public key  $pk_{i^*}$  and two equal-length messages  $m_0, m_1 \in \{0, 1\}^{l_0}$ . Algorithm  $\mathcal{B}$  responds as follows:

1. Recover tuple  $(pk_{i^*}, x^*, c^*)$  from  $K^{\text{list}}$ . Recall that according to the constraints described in IND-PRE-CCA game,  $K^{\text{list}}$  should contain this tuple, and  $c^*$  is equal to 0 (indicating that  $pk_{i^*} = g^{\frac{x^*}{a}}$ ).
2. Pick  $e^*, s^* \xleftarrow{\$} \mathbb{Z}_q^*$ , and compute  $D^* = (g^b)^{-e^* x^*} \left(g^{\frac{1}{a}}\right)^{x^* s^*}$  and  $E^* = (g^b)^{x^*}$ .
3. Pick  $F^* \xleftarrow{\$} \{0, 1\}^{l_0 + l_1}$  and define  $H_3(D^*, E^*, F^*) = e^*$ .
4. Pick  $\delta \xleftarrow{\$} \{0, 1\}, \omega^* \xleftarrow{\$} \{0, 1\}^{l_1}$ , and implicitly define  $H_2(g^{ab}) = (m_\delta \parallel \omega^*) \oplus F^*$  and  $H_1(m_\delta, \omega^*, pk_{i^*}) = ab$  (Note that algorithm  $\mathcal{B}$  knows neither  $ab$  nor  $g^{ab}$ ).
5. Return  $CT^* = (D^*, E^*, F^*, s^*)$  as the challenged ciphertext to adversary  $\mathcal{A}$ .

Note that by the construction given above, by letting  $u^* \triangleq s^* - abe^*$  and  $r^* \triangleq ab$ , we can see that the challenged ciphertext  $CT^*$  has the same distribution as the real one, since  $H_2$  acts as a random oracle, and

$$\begin{aligned} D^* &= (g^b)^{-e^* x^*} \left(g^{\frac{1}{a}}\right)^{x^* s^*} = \left(g^{\frac{x^*}{a}}\right)^{s^* - abe^*} = (pk_{i^*})^{s^* - abe^*} = (pk_{i^*})^{u^*}, \\ E^* &= (g^b)^{x^*} = \left(g^{\frac{x^*}{a}}\right)^{ab} = (pk_{i^*})^{ab} = (pk_{i^*})^{r^*}, \\ F^* &= H_2(g^{ab}) \oplus (m_\delta \parallel \omega^*) = H_2(g^{r^*}) \oplus (m_\delta \parallel \omega^*), \\ s^* &= (s^* - abe^*) + abe^* = u^* + ab \cdot H_3(D^*, E^*, F^*) = u^* + r^* \cdot H_3(D^*, E^*, F^*). \end{aligned}$$

**Phase 2.** Adversary  $\mathcal{A}$  continues to issue the rest of queries as in Phase 1, with the restrictions described in the IND-PRE-CCA game. Algorithm  $\mathcal{B}$  responds to these queries for  $\mathcal{A}$  as in Phase 1.

**Guess.** Eventually, adversary  $\mathcal{A}$  returns a guess  $\delta' \in \{0, 1\}$  to  $\mathcal{B}$ . Algorithm  $\mathcal{B}$  randomly picks a tuple  $(R, \beta)$  from the list  $H_2^{\text{list}}$  and outputs  $R$  as the solution to the given mCDH instance.

**Analysis:** Now let's analyze the simulation. The main idea of the analysis is borrowed from [8]. We first evaluate the simulations of the random oracles. From the constructions of  $H_3$  and  $H_4$ , it is clear that the simulations of  $H_3$  and  $H_4$  are perfect. As long as adversary  $\mathcal{A}$  does not query  $(m_\delta, \omega^*, pk_{i^*})$  to  $H_1$  nor  $g^{ab}$  to  $H_2$ , where  $\delta$  and  $\omega^*$  are chosen by  $\mathcal{B}$  in the Challenge phase, the simulations of  $H_1$  and  $H_2$  are perfect. By  $\text{AskH}_1^*$  we denote the event that  $(m_\delta, \omega^*)$  has been queried to  $H_1$ . Also, by  $\text{AskH}_2^*$  we denote the event that  $g^{ab}$  has been queried to  $H_2$ .

As argued before, the challenged ciphertext provided for  $\mathcal{A}$  is identically distributed as the real one from the construction. From the description of the simulation, it can be seen that the responses to  $\mathcal{A}$ 's re-encryption key queries are also perfect.

Next, we analyze the simulation of the re-encryption oracle. The responses to adversary  $\mathcal{A}$ 's re-encryption queries are perfect, unless  $\mathcal{A}$  can submit valid second-level ciphertexts without querying hash function  $H_1$  (denote this event by  $\text{ReEncErr}$ ). However, since  $H_1$  acts as a random oracle and adversary  $\mathcal{A}$  issues at most  $q_{re}$  re-encryption queries, we have

$$\Pr[\text{ReEncErr}] \leq \frac{q_{re}}{q}.$$

Now, we evaluate the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. Fortunately, these errors are not significant as shown below: Suppose that  $(pk, \text{CT})$ , where  $\text{CT} = (D, E, F, s)$  or  $\text{CT} = (E, F)$ , has been issued as a *valid* ciphertext. Even  $\text{CT}$  is valid, there is a possibility that  $\text{CT}$  can be produced without querying  $g^r$  to  $H_2$ , where  $r = H_1(m, \omega, pk)$ . Let  $\text{Valid}$  be an event that  $\text{CT}$  is valid, and let  $\text{AskH}_2$  and  $\text{AskH}_1$  respectively be events that  $g^r$  has been queried to  $H_2$  and  $(m, \omega, pk)$  has been queried to  $H_1$  with respect to  $(E, F) = (pk^r, H_2(g^r) \oplus (m \parallel \omega))$ , where  $r = H_1(m, \omega, pk)$ . We then have

$$\begin{aligned} \Pr[\text{Valid} | \neg \text{AskH}_2] &= \Pr[\text{Valid} \wedge \text{AskH}_1 | \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \neg \text{AskH}_1 | \neg \text{AskH}_2] \\ &\leq \Pr[\text{AskH}_1 | \neg \text{AskH}_2] + \Pr[\text{Valid} | \neg \text{AskH}_1 \wedge \neg \text{AskH}_2] \\ &\leq \frac{q_{H_1}}{2^{l_0+l_1}} + \frac{1}{q}, \end{aligned}$$

and similarly  $\Pr[\text{Valid} | \neg \text{AskH}_1] \leq \frac{q_{H_2}}{2^{l_0+l_1}} + \frac{1}{q}$ . Thus we have

$$\Pr[\text{Valid} | (\neg \text{AskH}_1 \vee \neg \text{AskH}_2)] \leq \Pr[\text{Valid} | \neg \text{AskH}_1] + \Pr[\text{Valid} | \neg \text{AskH}_2] \leq \frac{q_{H_1} + q_{H_2}}{2^{l_0+l_1}} + \frac{2}{q}.$$

Let  $\text{DecErr}$  be the event that  $\text{Valid} | (\neg \text{AskH}_1 \vee \neg \text{AskH}_2)$  happens during the entire simulation. Then, since  $q_d$  decryption oracles are issued, we have

$$\Pr[\text{DecErr}] \leq \frac{(q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} + \frac{2q_d}{q}.$$

Now let  $\text{Good}$  denote the event  $\text{AskH}_2^* \vee (\text{AskH}_1^* | \neg \text{AskH}_2^*) \vee \text{ReEncErr} \vee \text{DecErr}$ . If event  $\text{Good}$  does not happen, it is clear that adversary  $\mathcal{A}$  can not gain any advantage in guessing  $\delta$  due to the randomness of the output of the random oracle  $H_2$ . Namely, we

have  $\Pr[\delta = \delta' | \neg \text{Good}] = \frac{1}{2}$ . Hence, by splitting  $\Pr[\delta' = \delta]$ , we have

$$\begin{aligned} \Pr[\delta' = \delta] &= \Pr[\delta' = \delta | \neg \text{Good}] \Pr[\neg \text{Good}] + \Pr[\delta' = \delta | \text{Good}] \Pr[\text{Good}] \\ &\leq \frac{1}{2} \Pr[\neg \text{Good}] + \Pr[\text{Good}] \\ &= \frac{1}{2} (1 - \Pr[\text{Good}]) + \Pr[\text{Good}] \\ &= \frac{1}{2} + \frac{1}{2} \Pr[\text{Good}] \end{aligned}$$

and

$$\Pr[\delta' = \delta] \geq \Pr[\delta' = \delta | \neg \text{Good}] \Pr[\neg \text{Good}] = \frac{1}{2} (1 - \Pr[\text{Good}]) = \frac{1}{2} - \frac{1}{2} \Pr[\text{Good}].$$

Then we have

$$|\Pr[\delta' = \delta] - \frac{1}{2}| \leq \frac{1}{2} \Pr[\text{Good}].$$

By definition of the advantage  $(\epsilon - \nu)$  for the IND-PRE-CCA adversary, we then have

$$\begin{aligned} \epsilon - \nu &= |\Pr[\delta' = \delta] - \frac{1}{2}| \\ &\leq \frac{1}{2} \Pr[\text{Good}] = \frac{1}{2} (\Pr[\text{AskH}_2^* \vee (\text{AskH}_1^* | \neg \text{AskH}_2^*) \vee \text{ReEncErr} \vee \text{DecErr}]) \\ &\leq \frac{1}{2} (\Pr[\text{AskH}_2^*] + \Pr[\text{AskH}_1^* | \neg \text{AskH}_2^*] + \Pr[\text{ReEncErr}] + \Pr[\text{DecErr}]). \end{aligned}$$

Since  $\Pr[\text{ReEncErr}] \leq \frac{q_{re}}{q}$ ,  $\Pr[\text{DecErr}] \leq \frac{(q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} + \frac{2q_d}{q}$  and  $\Pr[\text{AskH}_1^* | \neg \text{AskH}_2^*] \leq \frac{q_{H_1}}{2^{l_0+l_1}}$ , we obtain

$$\begin{aligned} \Pr[\text{AskH}_2^*] &\geq 2(\epsilon - \nu) - \Pr[\text{AskH}_1^* | \neg \text{AskH}_2^*] - \Pr[\text{DecErr}] - \Pr[\text{ReEncErr}] \\ &\geq 2(\epsilon - \nu) - \frac{q_{H_1}}{2^{l_0+l_1}} - \frac{(q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} - \frac{2q_d}{q} - \frac{q_{re}}{q} \\ &= 2(\epsilon - \nu) - \frac{q_{H_1} + (q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} - \frac{q_{re} + 2q_d}{q}. \end{aligned}$$

Meanwhile, if event  $\text{AskH}_2^*$  happens, algorithm  $\mathcal{B}$  will be able to solve the mCDH instance, and consequently, we obtain

$$\epsilon' \geq \frac{1}{q_{H_2}} \left( 2(\epsilon - \nu) - \frac{q_{H_1} + (q_{H_1} + q_{H_2})q_d}{2^{l_0+l_1}} - \frac{q_{re} + 2q_d}{q} \right).$$

From the description of the simulation, the running time of algorithm  $\mathcal{B}$  can be bounded by

$$\begin{aligned} t' &\leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_u + q_c + q_{rk} + q_{re} + q_d) \mathcal{O}(1) \\ &\quad + (q_u + q_c + 4q_{re} + 3q_d + (2q_d + q_{re})q_{H_1})t_e. \end{aligned}$$

This completes the proof of Theorem 1.  $\square$

## 4 Proposed Unidirectional Proxy Re-Encryption Scheme

In this section, based on our bidirectional PRE scheme, we present a CCA-secure unidirectional PRE scheme without pairings. For notational convenience, we denote this scheme by  $\Pi_{\text{Uni}}$ . We stress that a previous version of our unidirectional PRE scheme can not satisfy the master secret security, and left it as an open question in the last version.



Fortunately, in this version, we can improve our scheme to satisfy this security.

#### 4.1 Proposed Scheme: $\Pi_{\text{Uni}}$

The proposed scheme  $\Pi_{\text{B}}$  consists of the following algorithms:

**GlobalSetup**( $\kappa$ ): The global setup algorithm is similar to that of scheme  $\Pi_{\text{B}}$ , with the exceptions that,  $H_1$  and  $H_4$  are defined as  $H_1 : \{0, 1\}^{l_0} \times \{0, 1\}^{l_1} \rightarrow \mathbb{Z}_q^*$  and  $H_4 : \mathbb{Z}_q^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$ , and another hash function  $H_5$  is chosen such that  $H_5 : \mathbb{G}^3 \rightarrow \mathbb{Z}_q^*$ . The global parameters are

$$param = (q, \mathbb{G}, g, H_1, H_2, H_3, H_4, H_5, l_0, l_1).$$

**KeyGen**( $i$ ): To generate the public/secret key pair for user  $i$ , this key generation algorithm picks a random  $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$ , and then sets  $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$  and  $sk_i = (x_{i,1}, x_{i,2})$ .

**ReKeyGen**( $sk_i, pk_j$ ): On input user  $i$ 's secret key  $sk_i = (x_{i,1}, x_{i,2})$  and user  $j$ 's public key  $pk_j = (pk_{j,1}, pk_{j,2})$ , this algorithm generates the re-encryption key  $rk_{i,j}$  from user  $i$  to  $j$  as below:

1. Pick  $v' \xleftarrow{\$} \mathbb{Z}_q^*$ . Compute  $v = H_4(v', pk_j)$ .
2. Compute  $V = g^v$ , and  $h_j = H_5(V, pk_j, pk_{j,2})$ .
3. Define  $rk_{i,j}^{(1)} = \frac{h_j}{x_{i,1} + x_{i,2}}$ . Return  $rk_{i,j} = (rk_{i,j}^{(1)}, V)$ .

**Encrypt**( $pk_i, m$ ): On input a public key  $pk_i = (pk_{i,1}, pk_{i,2})$  and a plaintext  $m \in \{0, 1\}^{l_0}$ , this algorithm works as below:

1. Pick  $u \xleftarrow{\$} \mathbb{Z}_q^*, \omega \xleftarrow{\$} \{0, 1\}^{l_1}$ , and compute  $r = H_1(m, \omega)$ .
2. Compute  $D = (pk_{i,1}pk_{i,2})^u, E = (pk_{i,1}pk_{i,2})^r, F = H_2(g^r) \oplus (m \parallel \omega), s = u + r \cdot H_3(D, E, F) \pmod q$ .
3. Output the ciphertext  $\text{CT} = (D, E, F, s)$ .

**ReEncrypt**( $rk_{i,j}, \text{CT}_i, pk_j$ ): On input a re-encryption key  $rk_{i,j}$ , a second-level ciphertext  $\text{CT}_i$  under public key  $pk_i = (pk_{i,1}, pk_{i,2})$ , this algorithm re-encrypts this ciphertext under public key  $pk_j = (pk_{j,1}, pk_{j,2})$  as follows:

1. Parse  $\text{CT}_i$  as  $\text{CT}_i = (D, E, F, s)$  and  $rk_{i,j}$  as  $rk_{i,j} = (rk_{i,j}^{(1)}, V)$ .
2. Check whether  $(pk_{i,1}pk_{i,2})^s = D \cdot E^{H_3(D, E, F)}$  holds. If not, output  $\perp$ .
3. Otherwise, compute  $E' = E^{rk_{i,j}^{(1)}} = g^{r \cdot (x_{i,1} + x_{i,2}) \cdot \frac{h_j}{x_{i,1} + x_{i,2}}} = g^{r \cdot h_j}$ , and output the first-level ciphertext  $\text{CT}_j = (E', F, V)$ .

**Decrypt**( $\text{CT}_i, sk_i$ ): On input a secret key  $sk_i = (x_{i,1}, x_{i,2})$  and ciphertext  $\text{CT}_i$ , this algorithm works according to two cases:

- $\text{CT}$  is a second-level ciphertext  $\text{CT} = (D, E, F, s)$ : Check whether  $(g^{x_{i,1} + x_{i,2}})^s = D \cdot E^{H_3(D, E, F)}$  holds. If not, output  $\perp$ . Otherwise, compute  $(m \parallel \omega) = F \oplus H_2(E^{\frac{1}{g^{x_{i,1} + x_{i,2}}}})$ , and return  $m$  if  $E = (g^{x_{i,1} + x_{i,2}})^{H_1(m, \omega)}$  holds; else return  $\perp$ .
- $\text{CT}$  is a first-level ciphertext  $\text{CT} = (E', F, V)$ : Compute  $h = H_5(V, pk, V^{x_{i,2}})$  and  $(m \parallel \omega) = F \oplus H_2(E'^{\frac{1}{h}})$ , and check whether  $E' = g^{H_1(m, \omega) \cdot h}$  holds. If yes, return  $m$ ; else return  $\perp$ .

It can be verified that, given the re-encryption key  $rk_{i,j} = (\frac{H_5(V, pk_j, pk_{j,2}^v)}{x_{i,1} + x_{i,2}}, g^v)$ , the proxy is unable to generate the re-encryption key  $rk_{j,i}$  for the opposite direction, and hence it is impossible for him to convert a ciphertext intended for user  $j$  into a ciphertext intended for user  $i$ . Therefore, our scheme is unidirectional.

## 4.2 Security Analysis

In this subsection, we prove the chosen-ciphertext security for scheme  $\Pi_{\text{Uni}}$  under the CDH assumption.

**Theorem 2** *Our scheme  $\Pi_{\text{Uni}}$  is IND-PRE-CCA secure in the random oracle model, assuming the CDH assumption holds in group  $\mathbb{G}$  and the Schnorr signature is EUF-CMA secure. Concretely, if there exists an adversary  $\mathcal{A}$ , who asks at most  $q_{H_i}$  random oracle queries to  $H_i$  with  $i \in \{1, \dots, 5\}$ , and breaks the  $(t, q_u, q_c, q_{rk}, q_{re}, q_d, \epsilon)$ -IND-PRE-CCA of our scheme  $\Pi_{\text{Uni}}$ , then, for any  $0 < \nu < \epsilon$ , there exists*

- either an algorithm  $\mathcal{B}$  which can solve the  $(t', \epsilon')$ -CDH problem in  $\mathbb{G}$  with

$$\begin{aligned} t' &\leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\ &\quad + (q_u + q_c + 2q_{rk} + 5q_{re} + q_d + q_{H_1}q_{re} + (q_{H_4} + 2q_{H_1})q_d)t_e, \\ \epsilon' &\geq \frac{1}{q_{H_2}} \left( \frac{2(\epsilon - \nu)}{e(1 + q_{rk})} - \frac{q_{H_1} + (q_{H_1} + q_{H_2} + q_{H_4})q_d}{2^{l_0 + l_1}} - \frac{q_{re} + 3q_d}{q} \right), \end{aligned}$$

where  $t_e$  denotes the running time of an exponentiation in group  $\mathbb{G}$ .

- or an attacker who breaks the EUF-CMA security of the Schnorr signature with advantage  $\nu$  within time  $t'$ .

**Proof.** Without loss of generality, we assume that the Schnorr signature is  $(t', \nu)$ -EUF-CMA secure for some probability  $0 < \nu < \epsilon$ . Since the CDH problem is equivalent to the DCDH problem, for convenience, we here prove this theorem under the DCDH problem. Suppose there exists a  $t$ -time adversary  $\mathcal{A}$  who can break the IND-PRE-CCA security of scheme  $\Pi_{\text{Uni}}$  with advantage  $\epsilon - \nu$ . Then we show how to construct an algorithm  $\mathcal{B}$  which can solve the  $(t', \epsilon')$ -DCDH problem in group  $\mathbb{G}$ .

Suppose  $\mathcal{B}$  is given as input a DCDH challenge tuple  $(g, g^{\frac{1}{a}}, g^b)$  with unknown  $a, b \xleftarrow{\$} \mathbb{Z}_q^*$ . Algorithm  $\mathcal{B}$ 's goal is to output  $g^{ab}$ . Algorithm  $\mathcal{B}$  acts as the challenger and plays the IND-PRE-CCA game with adversary  $\mathcal{A}$  in the following way.

**Setup.** Algorithm  $\mathcal{B}$  gives  $(q, \mathbb{G}, g, H_1, H_2, H_3, H_4, H_5, l_0, l_1)$  to  $\mathcal{A}$ . Here  $H_1, H_2, H_3, H_4$  and  $H_5$  are random oracles controlled by  $\mathcal{B}$ .

**Hash Oracle Queries.** At any time adversary  $\mathcal{A}$  can issue the random oracle queries  $H_i$  with  $i \in \{1, \dots, 5\}$ . Algorithm  $\mathcal{B}$  maintains five hash lists  $H_i^{\text{list}}$  with  $i \in \{1, \dots, 5\}$ , which are initially empty.  $\mathcal{B}$  responds  $H_2$  and  $H_3$  queries in the same way as in Theorem 1, and responds the other hash queries as below:

- $H_1$  queries: On receipt of an  $H_1$  queries on  $(m, \omega)$ , if this query has appeared on the  $H_1^{\text{list}}$  in a tuple  $(m, \omega, r)$ , return the predefined value  $r$  as the result of the query. Otherwise, choose  $r \xleftarrow{\$} \mathbb{Z}_q^*$ , add the tuple  $(m, \omega, r)$  to the list  $H_1^{\text{list}}$  and respond with  $H_1(m, \omega) = r$ .

- $H_4$  queries: On receipt of an  $H_4$  query  $(v', pk) \in \mathbb{Z}_q^* \times \mathbb{G}^2$ , if this query has appeared on the  $H_4^{\text{list}}$  in a tuple  $(v', pk, v)$ , return the predefined value  $v$  as the result of the query. Otherwise, choose  $v \xleftarrow{\$} \mathbb{Z}_q^*$ , add the tuple  $(v', pk, v)$  to the list  $H_4^{\text{list}}$  and respond with  $H_4(v', pk) = v$ .
- $H_5$  queries: On receipt of an  $H_5$  query  $(V, pk, S) \in \mathbb{G}^4$ , if this query has appeared on the  $H_5^{\text{list}}$  in a tuple  $(V, pk, S, \mu)$ , return the predefined value  $\mu$  as the result of the query. Otherwise, choose  $\mu \xleftarrow{\$} \mathbb{Z}_q^*$ , add the tuple  $(V, pk, S, \mu)$  to the list  $H_5^{\text{list}}$  and respond with  $H_5(V, pk, S) = \mu$ .

**Phase 1.** In this phase, adversary  $\mathcal{A}$  issues a series of queries as in the definition of the IND-PRE-CCA game.  $\mathcal{B}$  maintains two lists  $K^{\text{list}}$  and  $R^{\text{list}}$  which are initially empty, and answers these queries for  $\mathcal{A}$  as follows:

- *Uncorrupted key generation query*  $\langle i \rangle$ . Algorithm  $\mathcal{B}$  first picks  $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$  and flips a biased coin  $c_i \in \{0, 1\}$  that yields 0 with probability  $\theta$  and 1 with probability  $1 - \theta$ . If  $c_i = 0$ , it defines  $pk_i = (pk_{i,1}, pk_{i,2}) = \left( (g^{1/a})^{x_{i,1}}, (g^{1/a})^{x_{i,2}} \right)$ ; else defines  $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$ . Next, it adds the tuple  $(pk_i, x_{i,1}, x_{i,2}, c_i)$  to  $K^{\text{list}}$  and returns  $pk_i$  to adversary  $\mathcal{A}$ .
- *Corrupted key generation query*  $\langle j \rangle$ . Algorithm  $\mathcal{B}$  first picks  $x_{j,1}, x_{j,2} \xleftarrow{\$} \mathbb{Z}_q^*$  and defines  $pk_j = (pk_{j,1}, pk_{j,2}) = (g^{x_{j,1}}, g^{x_{j,2}})$ ,  $c_j = '-'$ . Next, it adds the tuple  $(pk_j, x_{j,1}, x_{j,2}, c_j)$  to  $K^{\text{list}}$  and returns  $(pk_j, (x_{j,1}, x_{j,2}))$  to adversary  $\mathcal{A}$ .
- *Re-encryption key generation query*  $\langle pk_i, pk_j \rangle$ : If  $R^{\text{list}}$  has contains a tuple for this entry  $(pk_i, pk_j)$ , return the predefined re-encryption key to  $\mathcal{A}$ . Otherwise, algorithm  $\mathcal{B}$  acts as follows:
  1. Recover tuples  $(pk_i, x_{i,1}, c_i)$  and  $(pk_j, x_{j,1}, c_j)$  from  $K^{\text{list}}$ .
  2. Pick  $v' \xleftarrow{\$} \mathbb{Z}_q^*$ . Compute  $v = H_4(v', pk_j)$ ,  $V = g^v$  and  $h_j = H_5(V, pk_j, pk_{j,2}^v)$ .
  3. Construct the first component  $rk_{i,j}^{(1)}$  according to the following cases:
    - $c_i = 1$  or  $c_i = '-'$ : define  $rk_{i,j}^{(1)} = \frac{h_j}{x_{i,1} + x_{i,2}}$ , and add  $(pk_i, pk_j, (rk_{i,j}^{(1)}, V), h_j, 1)$  into list  $R^{\text{list}}$ .
    - $(c_i = 0 \wedge c_j = 1)$  or  $(c_i = 0 \wedge c_j = 0)$ : pick  $rk_{i,j}^{(1)} \xleftarrow{\$} \mathbb{Z}_q^*$ , and add  $(pk_i, pk_j, (rk_{i,j}^{(1)}, V), h_j, 0)$  into list  $R^{\text{list}}$ .
    - $(c_i = 0 \wedge c_j = '-')$ : output “failure” and **abort**.
  4. Finally, return  $rk_{i,j} = (rk_{i,j}^{(1)}, V)$  to  $\mathcal{A}$ .
- *Re-encryption query*  $\langle pk_i, pk_j, \text{CT}_i (= (D, E, F, s)) \rangle$ : Parse  $pk_i$  as  $pk_i = (pk_{i,1}, pk_{i,2})$  and  $pk_j$  as  $pk_j = (pk_{j,1}, pk_{j,2})$ . If  $(pk_{i,1} pk_{i,2})^s \neq D \cdot E^{H_3(D, E, F)}$ , then output  $\perp$ . Otherwise, algorithm  $\mathcal{B}$  responds to this query as follows:
  1. Recover tuples  $(pk_i, x_{i,1}, x_{i,2}, c_i)$  and  $(pk_j, x_{j,1}, x_{j,2}, c_j)$  from  $K^{\text{list}}$ .
  2. If  $(c_i = 0 \wedge c_j = '-')$  does not hold, issue a re-encryption key generation query  $\langle pk_i, pk_j \rangle$  to obtain  $rk_{i,j}$ , and then return  $\text{ReEncrypt}(rk_{i,j}, \text{CT}_i, pk_j)$  to  $\mathcal{A}$ .
  3. Else, search whether there exists a tuple  $(m, \omega, pk_i, r) \in H_1^{\text{list}}$  such that  $pk_i^r = E$ . If there exists no such tuple, return  $\perp$ . Otherwise, first choose  $v' \xleftarrow{\$} \mathbb{Z}_q^*$ . Next, compute  $v = H_4(v', pk_j)$ ,  $V = g^v$  and  $h_j = H_5(V, pk_j, pk_{j,2}^v)$ . Finally, define  $E' = g^{h_j r}$ , and return  $(E', F, V)$  to  $\mathcal{A}$ .

- *Decryption query*  $\langle pk_i, CT_i \rangle$ :  $\mathcal{B}$  first parse  $pk_i = (pk_{i,1}, pk_{i,2})$  and recovers tuple  $(pk_i, x_{i,1}, x_{i,2}, c)$  from  $K^{\text{list}}$ . If  $c = 1$  or  $c = '-'$ , algorithm  $\mathcal{B}$  runs  $\text{Decrypt}(CT_i, (x_{i,1}, x_{i,2}))$  and returns the result to  $\mathcal{A}$ . Otherwise, algorithm  $\mathcal{B}$  works according to the following two cases:

- $CT_i$  is a second-level ciphertext  $CT_i = (D, E, F, s)$ : If  $(pk_{i,1}pk_{i,2})^s \neq D \cdot E^{H_3(D, E, F)}$ , return  $\perp$  to  $\mathcal{A}$  indicating that  $CT_i$  is an invalid ciphertext. Otherwise, search lists  $H_1^{\text{list}}$  and  $H_2^{\text{list}}$  to see whether there exist  $(m, \omega, pk_i, r) \in H_1^{\text{list}}$  and  $(R, \beta) \in H_2^{\text{list}}$  such that

$$(pk_{i,1}pk_{i,2})^r = E, \beta \oplus (m||\omega) = F \quad \text{and} \quad R = g^r.$$

If yes, return  $m$  to  $\mathcal{A}$ . Otherwise, return  $\perp$ .

- $CT_i$  is a first-level ciphertext  $CT_i = (E', F, V)$  re-encrypted from  $pk_j = (pk_{j,1}, pk_{j,2})$ : Algorithm  $\mathcal{B}$  first recovers tuples  $(pk_i, x_{i,1}, x_{i,2}, c)$  and  $(pk_j, x_{j,1}, x_{j,2}, c')$  from  $K^{\text{list}}$ , and then responds according to the following three cases:
  - \* If there exist a tuple  $(pk_j, pk_i, (rk^{(1)}, V), h, 1)$  in  $R^{\text{list}}$ : Compute  $(m||\omega) = F \oplus H_2(E'^{\frac{1}{h}})$ . If  $E' = g^{H_1(m, \omega) \cdot h}$  holds, return  $m$ , else return  $\perp$ .
  - \* If there exist a tuple  $(pk_j, pk_i, (rk^{(1)}, V), h, 0)$  in  $R^{\text{list}}$ : First, compute  $E = E'^{\frac{1}{rk^{(1)}}}$ . Next, search lists  $H_1^{\text{list}}$  and  $H_2^{\text{list}}$  to see whether there exist  $(m, \omega, r) \in H_1^{\text{list}}$  and  $(R, \beta) \in H_2^{\text{list}}$  such that

$$(pk_{j,1}pk_{j,2})^r = E, \beta \oplus (m||\omega) = F \quad \text{and} \quad R = g^r.$$

If yes, return  $m$  to  $\mathcal{A}$ , else return  $\perp$ .

- \* Otherwise: First search list  $H_4^{\text{list}}$  to see whether there exist a tuple  $(v', pk_i, v) \in H_4^{\text{list}}$  such that  $g^v = V$ . If no such tuple exists, return  $\perp$ . Otherwise, compute  $h = H_5(V, pk, pk_{i,2}^v)$ , and then search lists  $H_1^{\text{list}}$  and  $H_2^{\text{list}}$  to see whether there exist  $(m, \omega, r) \in H_1^{\text{list}}$  and  $(R, \beta) \in H_2^{\text{list}}$  such that

$$pk_{i,2}^h = E', \beta \oplus (m||\omega) = F \quad \text{and} \quad R = g^r.$$

If yes, return  $m$  to  $\mathcal{A}$ , else return  $\perp$ .

**Challenge.** When  $\mathcal{A}$  decides that Phase 1 is over, it outputs a target public key  $pk_{i^*} = (pk_{i^*,1}, pk_{i^*,2})$  and two equal-length messages  $m_0, m_1 \in \{0, 1\}^{l_0}$ . Algorithm  $\mathcal{B}$  responds as follows:

1. Recover tuple  $(pk_{i^*}, x_{i^*,1}, x_{i^*,2}, c^*)$  from  $K^{\text{list}}$ . Note that according to the constraints described in IND-PRE-CCA game,  $c^*$  must be equal to 1 or 0. If  $c^* = 1$ ,  $\mathcal{B}$  outputs “failure” and **abort**. Otherwise, it means that  $c^* = 0$ , and  $\mathcal{B}$  proceeds to execute the rest steps.
2. Pick  $e^*, s^* \xleftarrow{\$} \mathbb{Z}_q^*$ , and compute  $D^* = (g^b)^{-(x_{i^*,1}+x_{i^*,2})e^*} \left(g^{\frac{1}{a}}\right)^{(x_{i^*,1}+x_{i^*,2})s^*}$  and  $E^* = (g^b)^{(x_{i^*,1}+x_{i^*,2})}$ .
3. Pick  $F^* \xleftarrow{\$} \{0, 1\}^{l_0+l_1}$  and define  $H_3(D^*, E^*, F^*) = e^*$ .
4. Pick  $\delta \xleftarrow{\$} \{0, 1\}$ ,  $\omega^* \xleftarrow{\$} \{0, 1\}^{l_1}$ , and implicitly define  $H_2(g^{ab}) = (m_\delta || \omega^*) \oplus F^*$  and  $H_1(m_\delta, \omega^*) = ab$  (Note that algorithm  $\mathcal{B}$  knows neither  $ab$  nor  $g^{ab}$ ).
5. Return  $CT^* = (D^*, E^*, F^*, s^*)$  as the challenged ciphertext to adversary  $\mathcal{A}$ .

Again, let  $u^* \triangleq s^* - abe^*$  and  $r^* \triangleq ab$ , it can be easily verified that the challenged ciphertext  $CT^*$  has the same distribution as the real one.

**Phase 2.** Adversary  $\mathcal{A}$  continues to issue the rest of queries as in Phase 1, with the restrictions described in the IND-PRE-CCA game. Algorithm  $\mathcal{B}$  responds to these queries for  $\mathcal{A}$  as in Phase 1.

**Guess.** Eventually, adversary  $\mathcal{A}$  returns a guess  $\delta' \in \{0, 1\}$  to  $\mathcal{B}$ . Algorithm  $\mathcal{B}$  randomly picks a tuple  $(R, \beta)$  from the list  $H_2^{\text{list}}$  and outputs  $R$  as the solution to the given DCDH instance.

**Analysis:** Similar to the analysis in Theorem 1, we can have that algorithm  $\mathcal{B}$ 's advantage against the DCDH challenge is at least

$$\epsilon' \leq \frac{1}{q_{H_2}} \left( \frac{2(\epsilon - \nu)}{e(1 + q_{rk})} - \frac{q_{H_1} + (q_{H_1} + q_{H_2} + q_{H_4})q_d}{2^{l_0 + l_1}} - \frac{q_{re} + 3q_d}{q} \right),$$

and its time complexity is bounded by

$$t' \leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_u + q_c + q_{rk} + q_{re} + q_d)\mathcal{O}(1) \\ + (q_u + q_c + 2q_{rk} + 5q_{re} + q_d + q_{H_1}q_{re} + (q_{H_4} + 2q_{H_1})q_d)t_e.$$

This completes the proof of Theorem 2.  $\square$

Next, we prove the master secret security for our proposed C-PRE scheme.

**Theorem 3** *Our scheme  $\Pi_{\text{Uni}}$  has master secret security in the random oracle model, assuming the DL assumption holds in group  $\mathbb{G}$ . Concretely, if there exists an MSK-PRE adversary  $\mathcal{A}$ , who asks at most  $q_{H_i}$  random oracle queries to  $H_i$  with  $i \in \{1, \dots, 4\}$ , and breaks the  $(t, q_u, q_c, q_{rk}, \epsilon)$ -MSK-PRE security of our scheme, then there exists an algorithm  $\mathcal{B}$  which can solve the  $(t', \epsilon)$ -DL problem in group  $\mathbb{G}$  with  $t' \leq t + \mathcal{O}(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + 2q_u t_e + 2q_c t_e + q_{rk} t_e)$ .*

**Proof.** Suppose  $\mathcal{B}$  is given as input a DL challenge tuple  $(g, g^a) \in \mathbb{G}^2 \times \mathbb{G}_T$  with unknown  $a \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ . Algorithm  $\mathcal{B}$ 's goal is to output  $a$ . Algorithm  $\mathcal{B}$  acts as a challenger and plays the MSK-PRE game with adversary  $\mathcal{A}$  in the following way:

**Setup.** Algorithm  $\mathcal{B}$  gives  $(q, \mathbb{G}, g, H_1, H_2, H_3, H_4, H_5, l_0, l_1)$  to  $\mathcal{A}$ . Here  $H_1, H_2, H_3, H_4$  and  $H_5$  are random oracles controlled by  $\mathcal{B}$ , and they are conducted in the same way as in Theorem 2.

**Queries.** In this phase, adversary  $\mathcal{A}$  issues a series of queries as defined in the MSK-PRE game.  $\mathcal{B}$  maintains a list  $K^{\text{list}}$ , which is initially empty, and answers these queries for  $\mathcal{A}$  as follows:

- *Uncorrupted key generation query  $\langle i \rangle$ :* Algorithm  $\mathcal{B}$  first picks  $x_{i,1}, x_{i,2} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$ , and define  $pk_i = (pk_{i,1}, pk_{i,2}) = (g^a g^{x_{i,1}}, g^{-a} g^{x_{i,2}})$ . Next, set  $c_i = 0$  and add the tuple  $(pk_i, x_{i,1}, x_{i,2}, c_i)$  to the  $K^{\text{list}}$ . Finally, it returns  $pk_i$  to adversary  $\mathcal{A}$ . Note that the secret key with respect to  $pk_i$  is  $sk_i = (a + x_{i,1}, -a + x_{i,2})$ , which is unknown to both  $\mathcal{B}$  and  $\mathcal{A}$ .
- *Corrupted key generation query  $\langle j \rangle$ :*  $\mathcal{B}$  first picks  $x_{j,1}, x_{j,2} \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$  and defines  $pk_j = (pk_{j,1}, pk_{j,2}) = (g^{x_{j,1}}, g^{x_{j,2}})$  and  $c_j = 1$ . Next, it adds the tuple  $(pk_j, x_{j,1}, x_{j,2}, c_j)$  to the  $K^{\text{list}}$  and returns  $(pk_j, (x_{j,1}, x_{j,2}))$  to adversary  $\mathcal{A}$ .
- *Re-encryption key query  $\langle pk_i, pk_j \rangle$ :*  $\mathcal{B}$  first parses  $pk_i$  as  $pk_i = (pk_{i,1}, pk_{i,2})$  and  $pk_j = (pk_{j,1}, pk_{j,2})$ . Next, it recovers tuples  $(pk_i, x_{i,1}, x_{i,2}, c_i)$  and  $(pk_j, x_{j,1}, x_{j,2}, c_j)$  from the  $K^{\text{list}}$ . Then, it constructs the re-encryption key  $rk_{i,j}$  for adversary  $\mathcal{A}$  according to the following situations:

- If  $c_i = 1$ , it means that  $sk_i = (x_{i,1}, x_{i,2})$ , which is known to  $\mathcal{B}$ . So,  $\mathcal{B}$  certainly run algorithm  $\text{ReKeyGen}(sk_i, pk_j)$  to generate the re-encryption key, which is then returned to  $\mathcal{A}$ .
- If  $c_i = 0$ , it means that  $sk_i = (a + x_{i,1}, -a + x_{i,2})$ .  $\mathcal{B}$  first picks  $v' \xleftarrow{\$} \mathbb{Z}_q^*$ , and computes  $v = H_4(v', pk_j)$ ,  $V = g^v$  and  $h_j = H_5(V, pk_j, pk_{j,2}^v)$ . Next, it defines  $rk_{i,j}^{(1)} = \frac{h_j}{x_{i,1} + x_{i,2}}$ , and returns  $rk_{i,j} = (rk_{i,j}^{(1)}, V)$  to  $\mathcal{A}$ . Observe that this is indeed a valid re-encryption key, since  $rk_{i,j}^{(1)} = \frac{h_j}{x_{i,1} + x_{i,2}} = \frac{h_j}{(a + x_{i,1}) + (-a + x_{i,2})}$ .

**Output.** Eventually,  $\mathcal{A}$  outputs the secret key  $sk_{i^*} = (sk_{i^*,1}, sk_{i^*,2})$  with respect to the public key  $pk_{i^*}$ .  $\mathcal{B}$  first recovers the tuple  $(pk_{i^*}, x_{i^*,1}, x_{i^*,2}, c_{i^*})$  from the  $K^{\text{list}}$  (Note that according to the restriction specified in the MSK-PRE game, we have  $c_{i^*} = 0$ ), and then outputs  $sk_{i^*,1} - x_{i^*,1}$  as the solution to the DL challenge. Note that, if  $sk_{i^*} = (sk_{i^*,1}, sk_{i^*,2})$  is a valid secret key with respect to  $pk_{i^*}$ , then we have  $sk_{i^*,1} = a + x_{i^*,1}$  and  $sk_{i^*,2} = -a + x_{i^*,2}$ . Therefore,  $sk_{i^*,1} - x_{i^*,1}$  is indeed equal to  $a$ , and hence it is the solution to the DL instance.

This completes the description of the simulation. It is clear that the simulations of the random oracles, say  $H_1, H_2, H_3, H_4$ , are perfect. Also, it can be verified that the responses for the uncorrupted key generation queries, the corrupted key generation queries and the re-encryption key query are all perfect. Thus, when adversary  $\mathcal{A}$  outputs the valid secret key  $sk_{i^*}$  with advantage  $\epsilon$ ,  $\mathcal{B}$  can resolve the DL problem with the same advantage. It can be easily seen that  $\mathcal{B}$ 's running time is bounded by  $t' \leq t + \mathcal{O}(q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + 2q_{ut_e} + 2q_{ct_e} + q_{rk_{t_e}})$ . Thus the proof of Theorem 3 is concluded.  $\square$

### 4.3 Comparisons

In this subsection, we compare our scheme  $\Pi_{\text{Uni}}$  with Libert-Vergnaud's PRE scheme [20] (denoted by LV Scheme), which is RCCA secure in public key scenarios. The comparison results indicate that our scheme  $\Pi_{\text{Uni}}$  is much more efficient than LV Scheme. It is worth noting that the computational cost and the ciphertext length in our scheme *decrease* with re-encryption, while those in LV Scheme *increase* with re-encryption. The security of our scheme is related to the standard and well-studied CDH assumption, while LV Scheme is proved under a stronger and less-studied assumption, named 3-quotient decision bilinear Diffie-Hellman (3-QDBDH) assumption. A limitation of our scheme is that it is proved in the random oracle model, while LV Scheme is proved in the standard model.

Schemes		Libert-Vergnaud's Scheme	Our $\Pi_{\text{Bi}}$	
<b>Comput. Cost</b>	Encrypt	$2t_e + 1t_{me} + 1t_s$	$3t_e$	
	Re-Encrypt	$2t_p + 4t_e + 1t_v$	$3t_e$	
	Decrypt	2nd-level CiphTxt	$3t_p + 2t_e + 1t_v$	$4t_e$
		1st-level CiphTxt	$5t_p + 2t_e + 1t_v$	$3t_e$
<b>CiphTxt Length</b>	2nd-level CiphTxt	$1 pk_s  + 2 \mathbb{G}_e  + 1 \mathbb{G}_T  + 1 \sigma_s $	$3 \mathbb{G}  + 1 \mathbb{Z}_q $	
	1st-level CiphTxt	$1 pk_s  + 4 \mathbb{G}_e  + 1 \mathbb{G}_T  + 1 \sigma_s $	$3 \mathbb{G} $	
<b>Without Random Oracles?</b>		✓	×	
<b>Master secret security</b>		✓	✓	
<b>Underlying Assumptions</b>		3-QDBDH	CDH	

Table 2: Efficiency Comparison between Scheme  $\Pi_{\text{Uni}}$  and Libert-Vergnaud's Scheme

## 5 Conclusions

We presented two direct constructions of CCA-secure proxy re-encryption schemes, including one bidirectional scheme and one unidirectional scheme. Both schemes share the following distinguished features: (i) no bilinear pairing computation is involved; (ii) the ciphertext size and the computational cost decrease with re-encryption. Thus, compared with existing CCA-secure proxy re-encryption schemes, our scheme is highly efficient. We remark that our schemes are single-hop and can only be proved CCA-secure in the random oracle model. It would be interesting to construct a PRE scheme which is multi-hop, CCA-secure in the standard model, and yet without pairings. .

## 6 Acknowledgements

We would like to thank the anonymous referees for their helpful comments. This work is supported by the Office of Research, Singapore Management University. It is also partially supported by the National Science Foundation of China under Grant Nos. 90704004, 60873229, 60673077, and the National High Technology Research and Development Program of China (863 Program) under Grants No 2008AA01Z403 and 2007AA01Z456.

## References

- [1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In Proc. of NDSS 2005, pp. 29-43, 2005.
- [2] 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, February 2006.
- [3] D. Boneh, and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In advances in Cryptology-Eurocrypt'04, LNCS 3027, pp. 223-238, Springer-Verlag, 2004.
- [4] M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In advances in Cryptology-Eurocrypt'98, LNCS 1403, pp. 127-144, Springer-Verlag, 1998.
- [5] F. Bao, R. H. Deng, H. Zhu. Variations of Diffie-Hellman Problem. In Proc. of ICICS'03, LNCS 2836, pp. 301-312, Springer-Verlag, 2003.
- [6] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In Advances in Cryptology-Crypto'01, LNCS 2139, pp. 213-229. Springer-Verlag, 2001.
- [7] D. Boneh, E.-J. Goh, and T. Matsuo. Proposal for P1363.3 Proxy Re-encryption. <http://grouper.ieee.org/groups/1363/IBC/submissions/NTTDataProposal-for-P1363.3-2006-09-01.pdf>.
- [8] J. Baek, R. Safavi-Naini, and W. Susilo. Certificateless Public Key Encryption without Pairing. In Proc. of ISC'05. LNCS 3650, pp. 134-148, Springer-Verlag, 2005.

- [9] R. Canetti, S. Goldwasser. An Efficient Threshold Public Key Cryptosystem Secure against Adaptive Chosen Ciphertext Attack. In *advances in Cryptology-Eurocrypt'99*, LNCS 1592, pp.90-106. Springer-Verlag, 1999.
- [10] R. Caneti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In *Proceeding of ACM CCS 2007*.
- [11] C. Chu and W. Tzeng. Identity-Based Proxy Re-Encryption without Random Oracles. In *Proc. of ISC'07*, LNCS 4779, pp. 189-202, Springer-Verlag, 2007.
- [12] Y. Dodis, and A.-A. Ivan. Proxy Cryptography Revisited. In *Proc. of NDSS'03*, 2003.
- [13] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology-Crypto'84*, LNCS 196, pp.10-18, Springer-Verlag, 1984.
- [14] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes, In *Advances in Cryptology-Crypto'99*, LNCS 1666, pp. 537-554, Springer-Verlag, 1999.
- [15] M. Green and G. Ateniese. Identity-Based Proxy Re-Encryption. In *Proc. of ACNS'07*, LNCS 4521, pp. 288-306, Springer-Verlag, 2007.
- [16] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal Re-Encryption for Mixnets. In *Proc. of CT-RSA'04*, LNCS 2964, pp. 163-178, Springer-Verlag, 2004.
- [17] M. Jakobsson. On Quorum Controlled Asymmetric Proxy Re-Encryption. In *Proc. of PKC'99*, LNCS 1560, pp. 112-121, Springer-Verlag, 1999.
- [18] E. Kiltz and D. Galindo. Direct Chosen-Ciphertext Secure Identity-Based Key Encapsulation without Random Oracles. *Cryptology ePrint Archive*, Report 2006/034, 2006. <http://eprint.iacr.org/>.
- [19] Eike Kiltz. Chosen-Ciphertext Secure Identity-Based Encryption in the Standard Model with Short Ciphertexts. *Cryptology ePrint Archive*, Report 2006/122, 2006. <http://eprint.iacr.org/>.
- [20] B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In *Proc. of PKC'08*, LNCS 4929, pp. 360-379, Springer-Verlag, 2008.
- [21] B. Libert and D. Vergnaud. Multi-Use Unidirectional Proxy Re-Signatures. In P. Syverson and S. Jha, editor(s), *15th ACM Conference on Computer and Communications Security (ACM CCS 2008)*, ACM Press, October 2008, To appear.
- [22] B. Libert and D. Vergnaud. Tracing Malicious Proxies in Proxy Re-Encryption. In *Proc. of Pairing'2008*, LNCS 5209, pp. 332-353. Springer-Verlag, 2008.
- [23] T. Matsuo. Proxy Re-Encryption Systems for Identity-Based Encryption. In *Proc. of Paring'07*, LNCS 4575, pp. 247-267, Springer-Verlag, 2007.
- [24] Masahiro Mambo and Eiji Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. *IEICE Trans. Fund. Electronics Communications and Computer Science*, E80-A/1:54-63, 1997.
- [25] C. P. Schnorr. Efficient Identifications and Signatures for Smart Cards. In *advances in Cryptology-Crypto'89*, LNCS 435, pp. 239-251, Springer-Verlag, 1990.