Chosen-Ciphertext Secure Proxy Re-Encryption without Pairings*

Jian Weng[1,2], Robert H. Deng[1], Shengli Liu[3], Kefei Chen[3], Junzuo Lai[3], Xu An Wang[4]

[1]School of Information Systems, Singapore Management University, Singapore 178902
[2] Department of Computer Science, Jinan University, Guangzhou 510632, P.R. China
[3]Dept. of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, P.R. China
[4]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. Proxy re-encryption has found many practical applications, such as encrypted email forwarding, secure distributed file systems, and outsourced filtering of encrypted spam. In ACM CCS'07, Canetti and Hohenberger presented a bidirectional PRE scheme with chosen-ciphertext security, and left an important open problem to construct a chosen-ciphertext secure proxy re-encryption scheme without pairings. In this paper, we propose a bidirectional PRE scheme with chosen-ciphertext security. The proposed scheme is fairly efficient due to two distinguished features: (i) it does not use the costly bilinear pairings; (ii) the computational cost and the ciphertext length *decrease* with re-encryption.

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 private key. A naive way is to have the mail server store your private key and act as follows: when an encrypted email message for you arrives, the mail server decrypts it using the stored private 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 private 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

*This is a full and improved version which appears in CANS 2008, M.K. Franklin, L.C.K. Hui, and D.S. Wong (Eds.), volume 5339 of LNCS, Springer-Verlag, 2008.

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 private 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, 21].

Blaze, Bleumer and Strauss [4] categorized two types of PRE schemes. If the re-encryption key $rk_{i,j}$ allows the proxy to convert ciphertexts under pk_i into ciphertexts under pk_j and *vice versa*, then the scheme is called *bidirectional*. If $rk_{i,j}$ allows the proxy to convert only from pk_i to pk_j, then the scheme is called *unidirectional*. 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 of these 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 [8] presented an elegant construction of CCA-secure bidirectional PRE scheme. Later, Libert and Vergnaud [18] 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 [6], 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 [8], 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 [1]. 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.

1.3 Related Works

Boneh, Goh and Matsuo [5] described a hybrid proxy re-encryption system based on the ElGamal-type public key encryption system [11] and Boneh-Boyen's identity-based encryption system [3]. Recently, Libert and Vergnaud [20] proposed a traceable proxy re-encryption system, in which a proxy that leaks its re-encryption key can be identified by the delegator. Proxy re-encryption has also been studied in identity-based settings [9, 13].

Another kind of cryptosystems related to proxy re-encryption is the *proxy encryption* cryptosystem [10, 15, 22]. In NDSS'03, Dodis and Ivan [10] presented generic constructions of proxy encryption schemes as well as several efficient concrete schemes. It should be

[1]Shao [24] claimed that he independently proposed a CCA-secure bidirectional PRE scheme without pairings. However, his scheme is not CCA-secure as they claimed, since there exist an attack against his scheme (please refer to Section 3.1).

noted that, as argued in [8, 18], 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 [14], 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 **??**, based on our bidirectional PRE scheme, we further propose a unidirectional PRE scheme without parings, and then prove its security. Finally, Section 4 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 $\{0, 1, 2, \cdots, q-1\}$, and $\mathbb{Z}_q^*$ denote $\mathbb{Z}_q \backslash \{0\}$. For a finite set S, $x \stackrel{\$}{\leftarrow} S$ 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 [8]:

GlobalSetup(κ)**:** The global setup algorithm takes as input a security parameter κ. 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/private key pair (pk_i, sk_i) for user i.

ReKeyGen(sk_i, sk_j)**:** The re-encryption key generation algorithm takes as input two private 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$(rk_{i,j}, \mathbf{CT}_i)$**:** 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$(sk, \mathbf{CT})$**:** The decryption algorithm takes as input a private key sk and a cipertext 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/private key pairs $(pk_i, sk_i), (pk_j, sk_j)$, the following conditions hold:

$$\mathsf{Decrypt}(sk_i, \mathsf{Encrypt}(pk_i, m)) = m,$$
$$\mathsf{Decrypt}\,(sk_j, \mathsf{ReEncrypt}(\mathsf{ReKeyGen}(sk_i, sk_j), \mathsf{Encrypt}(pk_i, m))) = m.$$

Remark 1. 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 [18]: 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 notions is derived from [8, 18], with slight modifications to allow the *adaptive* corruptions of users. The chosen-ciphertext security for a bidirectional PRE scheme Π can be defined via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

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

Phase 1. $\mathcal{A}$ adaptively issues queries $q_1, \cdots, q_m$ 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/private 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/private 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 $\mathsf{ReKeyGen}(sk_i, sk_j)$ to generate a re-encryption key $rk_{i,j}$. Then $\mathcal{C}$ returns $rk_{i,j}$ to $\mathcal{A}$. Here sk_i and sk_j are private 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 [8], it is required that either both pk_i and pk_j are corrupted, or alternately both are uncorrupted.
- *Re-encryption query* $\langle pk_i, pk_j, \mathrm{CT}_i \rangle$: $\mathcal{C}$ first runs algorithm $\mathsf{ReKeyGen}$ to generate the re-encryption key $rk_{i,j}$. Then it runs $\mathsf{ReEncrypt}(rk_{i,j}, \mathrm{CT}_i)$ 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* $\langle pk, \mathrm{CT} \rangle$: Challenger $\mathcal{C}$ returns the result of $\mathsf{Decrypt}(sk, \mathrm{CT})$ to $\mathcal{A}$, where sk is the private 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 $\langle i^* \rangle$. Challenger $\mathcal{C}$ flips a random coin $\delta \in \{0, 1\}$, and sets the challenge ciphertext to be $\mathrm{CT}^* = \mathsf{Encrypt}(pk_{i^*}, m_\delta)$, which is sent to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ issues additional queries $q_{m+1}, \cdots, q_{max}$ 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*[2] (pk', CT') of $(pk_{i^*}, \mathrm{CT}^*)$, 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 private key sk_j with respect to pk_j, $\mathcal{A}$ can not issue the re-encryption key generation query on $\langle pk_{i^*}, pk_j \rangle$ nor $\langle pk_j, pk_{i^*} \rangle$.
- *Re-encryption query* $\langle pk_i, pk_j, \mathrm{CT}_i \rangle$: Challenger $\mathcal{C}$ responds as in Phase 1. Here it is required that, if $\mathcal{A}$ has obtained the private key sk_j with respect to pk_j, then (pk_i, CT_i) can not be a derivative of $(pk_{i^*}, \mathrm{CT}^*)$.
- *Decryption query* $\langle pk, \mathrm{CT} \rangle$: Challenger $\mathcal{C}$ responds as in Phase 1. Here it is required that, (pk, CT) can not be a derivative of $(pk_{i^*}, \mathrm{CT}^*)$.

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 Π as

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

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 bidirectional PRE scheme* Π *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* $\mathrm{Adv}_{\Pi,\mathcal{A}}^{\text{IND-PRE-CCA}} \leq \epsilon$.

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 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 [19].

Definition 2 *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}.

[2]Derivative of $(pk_{i^*}, \mathrm{CT}^*)$ is inductively defined in [8] as below:

1. $(pk_{i^*}, \mathrm{CT}^*)$ is a derivative of itself;
2. If (pk, CT) is a derivative of $(pk_{i^*}, \mathrm{CT}^*)$ and (pk', CT') is a derivative of (pk, CT), then (pk', CT') is a derivative of $(pk_{i^*}, \mathrm{CT}^*)$.
3. If $\mathcal{A}$ has issued a re-encryption query $\langle pk, pk', \mathrm{CT} \rangle$ 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 $\langle pk', pk \rangle$ to obtain the re-encryption key rk, and $\mathrm{CT}' = \mathsf{ReEncrypt}(rk, \mathrm{CT})$, then (pk', CT') is a derivative of (pk, CT).

Definition 3 *For a polynomial-time adversary $\mathcal{B}$, we define his advantage in solving the* mCDH *problem in group $\mathbb{G}$ as*

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

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

3 Proposed 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 [6, 11, 12] given in Figure 1. It is important to note that, in the ciphertext component $F = H_2(pk^r) \oplus (m\|\omega)$, the public key pk is embedded in the hash function H_2 and masked by $(m\|\omega)$. 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, 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 $rk_{x,y} = \frac{y}{x}$.

Setup(κ):	Encrypt(pk, m):	Decrypt($(E, F), sk$):
$x \xleftarrow{\$} \mathbb{Z}_q^*; pk = g^x; sk = x$	$\omega \xleftarrow{\$} \{0,1\}^{l_1}; r = H_1(m, \omega)$	$m\|\omega = F \oplus H_2(E^{sk})$
Return (pk, sk)	$E = g^r; F = H_2(pk^r) \oplus (m\|\omega)$	If $E = g^{H_1(m,\omega)}$ return m
	Return CT $= (E, F)$	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 massage space is $\mathcal{M} = \{0,1\}^{l_0}$.

Figure 1: CCA-secure "hashed" ElGamal encryption scheme

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

Figure 2: Modified CCA-secure "hashed" ElGamal encryption scheme

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 $\mathrm{CT}^* = (E^*, F^*) = \left(g^{xr^*}, H_2(g^{r^*}) \oplus (m_\delta\|\omega^*)\right)$. 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 $\mathrm{CT}' = (E', F') = (E^*, F^* \oplus z) = (g^{xr^*}, H_2(g^{r^*}) \oplus (m_\delta \| \omega^*) \oplus z)$. Next, he issues a corrupted key generation query to obtain a public/private key pair $(pk', sk') = (g^y, y)$, and then issues a re-encryption query to obtain a re-encrypt ciphertext, say $\mathrm{CT}'' = (E'', F'') = (g^{yr^*}, H_2(g^{r^*}) \oplus (m_\delta \| \omega^*) \oplus z)$, under the public key $pk' = g^y$. Finally, using the private key $sk' = y$, $\mathcal{A}$ can recover $(m_\delta \| \omega^*)$ as $(m_\delta \| \omega^*) = F'' \oplus H_2((E'')^{\frac{1}{y}}) \oplus z$, and eventually obtain the bit δ. 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 non-trivial 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 [23], 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 σ, and publishing $(vk_s, \mathrm{CT}, \sigma)$ 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 σ.

Setup(κ):	Sign(sk, m):	Verify($pk, (e, s), m$):
$x \xleftarrow{\$} \mathbb{Z}_q^*; pk = g^x; sk = x$ Return (pk, sk)	$u \xleftarrow{\$} \mathbb{Z}_q^*; D = g^u$ $e = H(m, D); s = (u + sk \cdot e) \mod q$ 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 $H : \{0,1\}^* \to \mathbb{Z}_q^*$.

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 $(E, F) = (pk^r, H_2(g^r) \oplus (m \| \omega))$, 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(κ):	Sign(sk, m):	Verify($pk, \boldsymbol{(D, s)}, m$):
$x \xleftarrow{\$} \mathbb{Z}_q^*; pk = g^x; sk = x$ Return (pk, sk)	$u \xleftarrow{\$} \mathbb{Z}_q^*; D = g^u$ $e = H(m, D); s = (u + sk \cdot e) \mod q$ Return $\boldsymbol{\sigma = (D, s)}$	If $\boldsymbol{g^s = D \cdot pk^{H(m,D)}}$ 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 $\mathrm{CT}_i = (D, E, F, s)$ 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 $\mathrm{CT}_j = (E', F) = (E^{x_j/x_i}, F)$ 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[3]: given a challenged ciphertext $\mathrm{CT}_i = (D, E, F, s)$ 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 this decryption query, since $(pk_i, (E, F))$ is *not* a derivative of (pk_i, CT_i). So, the adversary will be given the plaintext m_δ, and hence breaks the challenge δ. 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 $\mathrm{CT}_j = (D, E, F, s, E')$ instead of $\mathrm{CT}_j = (E', F)$. 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 $\mathrm{CT}_i = (D, E, F, s)$ 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 $\mathrm{CT}_j = (E', F') = (E^{\frac{x_j}{x_i}}, F \oplus H_4(E', g^{\frac{x_i}{x_j}}))$ as the re-encrypted ciphertext. The decryption algorithm for re-encrypted ciphertexts should be accordingly modified. To decrypt the re-encrypted ciphertext $\mathrm{CT}_j = (E', F')$, the decryptor with private 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 $\mathrm{CT}_i = (D, E, F, s)$ 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 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]This attack also applies to Shao's bidirectional PRE scheme [24]. In Shao's scheme, the original ciphertext is $\mathrm{CT}_i = (A, B, C, D, c, s)$, where $A = pk_i^r \bmod p, B = h^r \bmod p, C = g^r \cdot \sigma \bmod p, D = H_2(\sigma) \oplus m$ and $(c, s) \leftarrow \texttt{SoK.Gen}(A, B, pk_i, h, (C, D))$. While the re-encrypted ciphertext is $\mathrm{CT}_j = (A', C, D)$, where $A' = pk_j^r \bmod p$. Note that the components (A, C, D) of CT_i has the same form as the re-encrypted ciphertext CT_j, and $(pk_i, (A, C, D))$ is not a derivative of (pk_i, CT_i). Therefore, the adversary can break the CCA-security by issuing a decryption query on $\langle pk_i, (A, C, D)\rangle$.

3.2 Construction

We now present the detailed construction of our bidirectional PRE scheme. The proposed scheme consists of the following algorithms:

GlobalSetup(κ)**:** Given a security parameter κ, choose two big primes p and q such that $q|p-1$ and the bit-length of q is κ. 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 $H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \times \mathbb{G} \to \mathbb{Z}_q^*, H_2 : \mathbb{G} \to \{0,1\}^{l_0+l_1}, H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$ and $H_4 : \mathbb{G} \times \mathbb{G} \to \{0,1\}^{l_0+l_1}$. 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/private 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 private keys $sk_i = x_i$ and $sk_j = x_j$, this algorithm outputs the bidirectional re-encryption key $rk_{i,j} = x_j/x_i \mod q$.

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 $D = pk^u, E = pk^r, F = H_2(g^r) \oplus (m\|\omega), s = u + r \cdot H_3(D, E, F) \mod q$.
3. Output the ciphertext $\mathrm{CT} = (D, E, F, s)$.

ReEncrypt($rk_{i,j}, \mathbf{CT}_i, pk_j$)**:** 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 $\mathrm{CT}_i = (D, E, F, s)$.
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 $\mathrm{CT}_j = (pk_i, E', F')$.

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

- CT is a second-level ciphertext $\mathrm{CT} = (D, E, F, s)$: If $(g^x)^s = D \cdot E^{H_3(D,E,F)}$ does not hold, output $\perp$, else compute $m\|\omega = F \oplus H_2(E^{\frac{1}{x}})$, and return m if $E = (g^x)^{H_1(m,\omega,g^x)}$ holds and $\perp$ otherwise.
- CT is a first-level ciphertext $\mathrm{CT} = (pk_i, E', F')$: 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 $m\|\omega = F' \oplus H_2(E'^{\frac{1}{x}}) \oplus H_4(E', pk_i^{1/x})$. 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 with other existing bidirectional PRE schemes. To conduct a fair comparison, we choose Canetti and Hohenberger's PRE schemes [8], which are also bidirectional and achieve chosen-ciphertext security [4].

[4]Since Shao's scheme [24] is not CCA-secure, we do not compare his scheme with ours. In fact, our scheme beats Shao's scheme in all aspects.

Two PRE schemes are presented in [8], including one secure in the random oracle model (refereed to as CH Scheme I) and another one secure in the standard model (refereed to as CH Scheme II). Table 1 gives a comparison between our scheme 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 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 $[1, 2, 9, 18]$ *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.

Schemes			CH Scheme I	CH Scheme II	Our Scheme
Comput. Cost	Encrypt		$1t_p + 4t_e + 1t_{\mathsf{s}}$	$1t_p + 3t_e + 1t_{me} + 1t_{\mathsf{s}}$	$3t_e$
	Re-Encrypt		$4t_p + 1t_e + 1t_{\mathsf{v}}$	$4t_p + 2t_e + 1t_{\mathsf{v}}$	$4t_e$
	Decrypt	2nd-level CiphTxt	$5t_p + 1t_e + 1t_{\mathsf{v}}$	$5t_p + 2t_e + 1t_{\mathsf{v}}$	$4t_e$
		1st-level CiphTxt	$5t_p + 1t_e + 1t_{\mathsf{v}}$	$5t_p + 2t_e + 1t_{\mathsf{v}}$	$3t_e$
CiphTxt Length	2nd-level CiphTxt		$1\lvert pk_{\mathsf{s}}\rvert+3\lvert\mathbb{G}_e\rvert+1\lvert\mathbb{G}_T\rvert+1\lvert\sigma_{\mathsf{s}}\rvert$	$1\lvert pk_{\mathsf{s}}\rvert+3\lvert\mathbb{G}_e\rvert+1\lvert\mathbb{G}_T\rvert+1\lvert\sigma_{\mathsf{s}}\rvert$	$3\lvert\mathbb{G}\rvert+1\lvert\mathbb{Z}_q\rvert$
	1st-level CiphTxt		$1\lvert pk_{\mathsf{s}}\rvert+3\lvert\mathbb{G}_e\rvert+1\lvert\mathbb{G}_T\rvert+1\lvert\sigma_{\mathsf{s}}\rvert$	$1\lvert pk_{\mathsf{s}}\rvert+3\lvert\mathbb{G}_e\rvert+1\lvert\mathbb{G}_T\rvert+1\lvert\sigma_{\mathsf{s}}\rvert$	$3\lvert\mathbb{G}\rvert$
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, $|pk_{\mathsf{s}}|$ and $|\sigma_{\mathsf{s}}|$ denote the bit length of the one-time signature's public key and a one-time signature respectively.

Table 1: Comparison between Canetti-Hohenberger Schemes and Our Scheme [5]

3.4 Security Analysis

In this subsection, we prove the IND-PRE-CCA security for our scheme in the random oracle model.

Theorem 1 *Our proposed scheme 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 quires to H_i with $i \in \{1, \cdots, 4\}$, and breaks the $(t, q_u, q_c, q_{rk}, q_{re}, q_d, \epsilon)$*-IND-PRE-CCA *security of our scheme, then, for any $0 < \nu < \epsilon$, there exists*

[5] 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 [7,16,17], both the re-encryption and decryption in CH scheme I and II can further save two pairings, at the cost of several exponentiation operations.

- *either an algorithm $\mathcal{B}$ which can solve the (t', ϵ')-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 attacher who breaks the EUF-CMA security of the Schnorr signature with advantage ν within time t'.*

Proof. Without loss of generality, we assume that the Schnorr signature is (t', ν)-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 our scheme with advantage $\epsilon - \nu$. Then we show how to construct an algorithm $\mathcal{B}$ which can solve the (t', ϵ')-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^{list} and H_4^{list} which are initially empty, and responds as below:

- H_1 *queries*: On receipt of an H_1 queries on (m, ω, pk), if this query has appeared on the H_1^{list} in a tuple (m, ω, pk, r), return the predefined value r as the result of the query. Otherwise, choose $r \xleftarrow{\$} \mathbb{Z}_q^*$, add the tuple (m, ω, pk, r) to the list H_1^{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^{list} in a tuple (R, β), return the predefined value β as the result of the query. Otherwise, choose $\beta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add the tuple (R, β) to the list H_2^{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^{list} in a tuple (D, E, F, γ), return the predefined value γ as the result of the query. Otherwise, choose $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$, add the tuple (D, E, F, γ) to the list H_3^{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^{list} in a tuple (E', U, λ), return the predefined value λ as the result of the query. Otherwise, choose $\lambda \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add the tuple (E', U, λ) to the list H_4^{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^{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 $pk_i = \left(g^{1/a}\right)^{x_i}, c_i = 0$. Next, it adds the tuple (pk_i, x_i, c_i) to K^{list} and returns pk_i to adversary $\mathcal{A}$. Here the bit c_i is used to denote whether the private 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^{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^{list}, and then returns the re-encryption key x_j/x_i to $\mathcal{A}$.

- *Re-encryption query* $\langle pk_i, pk_j, \mathrm{CT}_i(= (D, E, F, s)) \rangle$: 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^{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 $E' = pk_j^r$. 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* $\langle pk, \mathrm{CT} \rangle$: Algorithm $\mathcal{B}$ first recovers tuple (pk, x, c) from list K^{list}. If $c = 1$, algorithm $\mathcal{B}$ runs $\mathsf{Decrypt}(\mathrm{CT}, x)$ and returns the result to $\mathcal{A}$. Otherwise, algorithm $\mathcal{B}$ works according to the following two cases:
  - CT is a second-level ciphertext $\mathrm{CT} = (D, E, F, s)$: If $pk^s \neq D \cdot E^{H_3(D,E,F)}$, return $\perp$ to $\mathcal{A}$. Otherwise, search lists H_1^{list} and H_2^{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
  $$pk^r = E, \beta \oplus (m \| \omega) = F \quad \text{and} \quad R = g^r.$$
  If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.
  - CT is a first-level ciphertext $\mathrm{CT} = (pk'', E', F')$: Algorithm $\mathcal{B}$ acts as follows:
    1. Recover tuples (pk, x, c) and (pk'', x'', c'') from K^{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 $U = g^{\frac{x'' a}{x}}$;
       * If $c = 1 \wedge c'' = 0$: Define $U = g^{\frac{x''}{a x}}$.
    3. search lists H_1^{list} and H_2^{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
    $$pk^r = E', \beta \oplus (m \| \omega) \oplus H_4(E', U) = F' \quad \text{and} \quad R = g^r.$$
    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^{list}. Recall that according to the constraints described in IND-PRE-CCA game, K^{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 $D^* = \left(g^b\right)^{-e^*x^*} \left(g^{\frac{1}{a}}\right)^{x^*s^*}$ and $E^* = \left(g^b\right)^{x^*}$.
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 $H_2(g^{ab}) = (m_\delta \| \omega^*) \oplus F^*$ and $H_1(m_\delta, \omega^*, pk_{i^*}) = ab$ (Note that algorithm $\mathcal{B}$ knows neither ab nor g^{ab}).
5. Return $\text{CT}^* = (D^*, E^*, F^*, s^*)$ 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

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

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, β) from the list H_2^{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 [6]. 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 δ and ω^* are chosen by $\mathcal{B}$ in the Challenge phase, the simulations of H_1 and H_2 are perfect. By AskH_1^* we denote the event that (m_δ, ω^*) has been queried to H_1. Also, by 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 $\mathsf{ReEncErr}$). However, since H_1 acts as a random oracle and adversary $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have

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

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, CT), where $\mathrm{CT} = (D, E, F, s)$ or $\mathrm{CT} = (E, F)$, has been issued as a *valid* ciphertext. Even CT is valid, there is a possibility that CT can be produced without querying g^r to H_2, where $r = H_1(m, \omega, pk)$. Let Valid be an event that CT is valid, and let AskH_2 and AskH_1 respectively be events that g^r has been queried to H_2 and (m, ω, pk) has been queried to H_1 with respect to $(E, F) = (pk^r, H_2(g^r) \oplus (m\|\omega))$, where $r = H_1(m, \omega, pk)$. We then have

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

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

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

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

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

Let Good denote the event $\mathsf{AskH}_2^* \vee (\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*) \vee \mathsf{ReEncErr} \vee \mathsf{DecErr}$. If event Good does not happen, it is clear that adversary $\mathcal{A}$ can not gain any advantage in guessing δ due to the randomness of the output of the random oracle H_2. Namely, we have $\Pr[\delta = \delta'|\neg\mathsf{Good}] = \frac{1}{2}$. Hence, by splitting $\Pr[\delta' = \delta]$, we have

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

and

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

Then we have

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

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

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

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

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

Meanwhile, if event AskH_2^* happens, algorithm $\mathcal{B}$ will be able to solve the mCDH instance, and consequently, we obtain

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

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

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

This completes the proof of Theorem 1. □

4 Conclusions

We presented a new bidirectional proxy re-encryption scheme, and proved its security in the random oracle model. Our proposed scheme is fairly efficient, since it does not rely on the costly bilinear pairings, and its computational cost and ciphertext length *decrease* with re-encryption. A limitation of our scheme is that its security is proved in the random oracle model. It would be interesting to construct CCA-secure proxy re-encryption without parings in the standard model. Another open question is to propose CCA-secure proxy re-encryption scheme with *multi-hop* and without pairings.

5 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] 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.

[6] 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.

[7] 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.

[8] R. Caneti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In Proceeding of ACM CCS 2007.

[9] 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.

[10] Y. Dodis, and A.-A. Ivan. Proxy Cryptography Revisited. In Proc. of NDSS'03, 2003.

[11] 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.

[12] 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.

[13] M. Green and G. Ateniese. Identity-Based Proxy Re-Encryption. In Proc. of ACNS'07, LNCS 4521, pp. 288-306, Springer-Verlag, 2007.

[14] 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.

[15] M. Jakobsson. On Quorum Controlled Asummetric Proxy Re-Encryption. In Proc. of PKC'99, LNCS 1560, pp. 112-121, Springer-Verlag, 1999.

[16] 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/.

[17] 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/.

[18] 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.

[19] 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.

[20] 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.

[21] T. Matsuo. Proxy Re-Encryption Systems for Identity-Based Encryption. In Proc. of Paring'07, LNCS 4575, pp. 247-267, Springer-Verlag, 2007.

[22] 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.

[23] C. P. Schnorr. Efficient Identifications and Signatures for Smart Cards. In advances in Cryptology-Crypto'89, LNCS 435, pp. 239-251, Springer-Verlag, 1990.

[24] J. Shao. Proxy re-cryptography, revisited. PhD Thesis. 2008.