International Journal of Network Security & Its Applications (IJNSA), Vol.4, No.2, March 2012

EFFICIENT CONDITIONAL PROXY RE-ENCRYPTION WITH CHOSEN CIPHER TEXT SECURITY

S. Sree Vivek[1], S. Sharmila Deva Selvi[1], V. Radhakishan[2], C. Pandu Rangan[1]

[1]Department of Computer Science and Engineering, Indian Institute of Technology Madras
svivek@cse.iitm.ac.in, sharmila@cse.iitm.ac.in, prangan@iitm.ac.in
[2]National Institute of Technology Trichy, India
vrkishan@gmail.com

ABSTRACT

In a proxy re-encryption (PRE) scheme, Alice gives a special information to a proxy that allows it to transform messages encrypted under Alice's public key into a encryption under Bob's public key such that the message is not revealed to the proxy. In [14], Jian Weng and others introduced the notion of conditional proxy re-encryption (C-PRE) with bilinear pairings. Later, a break for the same was published in [17] and a new C-PRE scheme with bilinear pairings was introduced. In C-PRE, the proxy also needs to have the right condition key to transform the ciphertext (associated with a condition set by Alice) under Alice's public key into ciphertext under Bob's public key, so that Bob can decrypt it. In this paper, we propose an efficient C-PRE scheme which uses substantially less number of bilinear pairings when compared to the existing one [17]. We then prove its chosen-ciphertext security under modified Computational Diffie-Hellman (mCDH) and modified Computational Bilinear Diffie-Hellman (mCBDH) assumptions in the random oracle model.

KEYWORDS

Random Oracle Model, Proxy Re-Cryptography, Conditional Proxy Re-encryption, Chosen Ciphertext Security.

1. INTRODUCTION

Encryption is used as a building block of any application requiring confidentiality. Let pk_i and pk_j be two independent public keys. As pointed out by Mambo and Okamato in [15], it is a common situation in practice where a data encrypted under pk_i is required to be encrypted under pk_j ($j \neq i$). When the holder of sk_i is online, $E_i(m)$ is decrypted using sk_i and then message m is encrypted under pk_j giving $E_j(m)$. But in many applications like encrypted mail forwarding, secure distributed file systems, and outsourced filtering of encrypted spam, when the holder of sk_i is not online, this has to be done by an untrusted party.

In 1998 Blaze, Bleumar, and Strauss [9] introduced the concept of proxy re-encryption (PRE). A re-encryption key ($rk_{i,j}$) is given to a potentially untrusted proxy so that the proxy can transform a message m encrypted under public key pk_i into an encryption of the same message m under a different public key pk_j without knowing the message. A PRE scheme can be of two types - unidirectional and bidirectional. The former is a scheme in which a re-encryption key ($rk_{i \rightarrow j}$) can be used to transform from pk_i to pk_j but not vice versa and the latter is a scheme in which the same re-encryption key ($rk_{i \leftrightarrow j}$) can be used to transform from pk_i to pk_j and vice versa. The re-encryption algorithm can be of two types - single hop, in which the re-encrypted ciphertext cannot be further re-encrypted and multi hop, in which the re-encrypted ciphertext can be further re-encrypted.

DOI : 10.5121/ijnsa.2012.4214

PRE can be used in many applications, including simplification of key distribution [9], key escrow [13], multicast [19], distributed file systems [3, 5], security in publish/subscribe systems [4], secure certified email mailing lists [20, 23], the DRM of Apple's iTunes [22], interoperable architecture of DRM [21], access control [11], and privacy for public transportation [7]. Hohenberger and others published a result of securely obfuscating re-encryption [16], which is the first positive result for obfuscating an encryption functionality. Shao and Cao have proposed a unidirectional PRE scheme without pairing [2]. Matthew Green and Giuseppe Ateniese have proposed a PRE scheme for ID-based cryptosystems [18].

Ran Canetti and Susan Hohenberger proposed a definition of security against chosen-ciphertext attacks for PRE schemes and presented a scheme that satisfied the definition [1]. In 2009, Jian Weng and others [14] introduced the concept of C-PRE, whereby Alice has a fine-grained control over the delegation. As a result, Alice can flexibly assign Bob the decryption capability based on the conditions attached to the messages using a proxy. For example, suppose Alice is on a vacation. She can make Bob to read only those messages which have the keyword "urgent" in their subject. This flexible delegation is obviously not possible with PRE schemes. In this paper, two separate keys are used - a partial re-encryption key and a condition key. The message can be delegated by the proxy only if both the keys are known.

Later in 2009, Jian Weng and others published a break of the scheme in [14] and gave a new scheme for C-PRE [17], which combines the re-encryption key and the condition key into a single key, which is then used for re-encryption. Also Cheng-Kang Chu and others in [8] introduced a generalized version of C-PRE named conditional proxy broadcast re-encryption (CPBRE), in which the proxy can re-encrypt the ciphertexts for a set of users at a time.

In this paper, we propose an efficient C-PRE scheme (single-hop and unidirectional) which uses significantly less number of bilinear pairings when compared to the existing schemes in [14] and [17]. Our scheme, as in [14], uses two separate keys for re-encryption.

1.1. Our Results

Let us briefly describe a C-PRE scheme. A C-PRE scheme involves a delegator (say user U_i), a delegatee (say user U_j) and a proxy. A message sent to U_i with condition w is encrypted by the sender using both U_i's public key and w. To re-encrypt the message to U_j, the proxy is given the re-encryption key ($rk_{i \to j}$) and the condition key ($ck_{i,w}$) corresponding to w. Both the keys can be generated only by U_i. These two keys form the secret trapdoor to be used by the proxy to perform translation. Proxy will not be able to re-encrypt cipher texts for which the right condition key is not available. Thus U_i can flexibly assign U_j the decryption rights by setting condition keys properly. The scheme works in practice as follows: the message encrypted for U_i is first handled by proxy and under appropriate conditions the proxy transforms the ciphertext into a ciphertext for U_j. However, proxy will obtain no information about the original message. While it is some what easier to design a PRE without pairing, designing C-PRE requires pairing based operations crucially. We have used a few constructions from [12] which drastically reduces the number of bilinear pairings. Table 1 compares the number of bilinear pairings and exponentiations between the scheme in [17] and our scheme.

Table 1. Computational Complexity Comparison

Algorithm	Scheme in [17]		Our Scheme	
	BP	EXP	BP	EXP
Encryption case 1	1	4	0	0
Encryption case 2	1	3	1	6
Re-Encryption	3	4	1	3
Decryption case 1	3	3	1	4
Decryption case 2	1	1	0	6
Total	9	15	3	19

BP – Bilinear Pairings, EXP – Exponentiations.

Encryption case 1 refers to the encryption without the condition. Encryption case 2 refers to the encryption with the condition. Decryption case 1 refers to the decryption of the re-encrypted ciphertext (first level ciphertext) and Decryption case 2 refers to the decryption of the encrypted ciphertext (second level ciphertext).

Although the number of exponentiations in our scheme is slightly more, it is insignificant when compared to the reduction in number of bilinear pairings. Thus, our scheme is more efficient than the existing one. We then formally prove the security of our scheme. We have slightly modified the security model in [14], as discussed in Section 3.

The C-PRE scheme in [14] has a break as given in [17]. Scheme in [17] has combined the two keys into a single key. Having the keys separate has an advantage. The delegation power of the proxy can be controlled. One of the two keys can be given to the proxy for partial re-encryption and the other key can be given to a third party for full re-encryption. Since the scheme in [14] has a break, our scheme is the only existing scheme having this unique property.

2. PRELIMINARIES

Bilinear Groups and Bilinear Pairings: Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups with the same prime order q. A bilinear pairing is a map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties.

- Bilinearity: We have $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ $\forall\, g_1, g_2 \in \mathbb{G}$ and $\forall\, a,b \in \mathbb{Z}_q^*$;
- Non-degeneracy: There exist $g_1, g_2 \in \mathbb{G}$ such that $\hat{e}(g_1, g_2) \neq 1$;
- Computability: There exists an efficient algorithm to compute $\hat{e}(g_1, g_2)$ $\forall\, g_1, g_2 \in \mathbb{G}$.

Modified Computational Diffie-Hellman Problem: Let G be a cyclic multiplicative group with a prime order q. Let g be the generator of $\mathbb{G}$, The mCDH problem in $\mathbb{G}$ is as follows:

Given $(g, g^{\frac{1}{a}}, g^a, g^b)$ for some a, b $\in \mathbb{Z}_q^*$, compute W = $g^{ab} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has an advantage ε in solving mCDH in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}(g, g^{\frac{1}{a}}, g^a, g^b) = g^{ab}\right] \geq \varepsilon$$

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

Modified Computational Bilinear Diffie-Hellman Problem: Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups with the same prime order q. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an admissible bilinear map and let g be the generator of $\mathbb{G}$. The mCBDH problem in ($\mathbb{G}$, $\mathbb{G}_T$, e) is as follows:

Given $(g, g^{\frac{1}{a}}, g^a, g^b, g^c)$ for some a, b, c $\in \mathbb{Z}_q^*$, compute W = $\hat{e}(g,g)^{abc} \in \mathbb{G}_T$. An algorithm $\mathcal{A}$ has an advantage ∈ in solving mCBDH in ($\mathbb{G}$, $\mathbb{G}_T$, e) if

$$\Pr\left[\mathcal{A}(g, g^{\frac{1}{a}}, g^a, g^b, g^c) = \hat{e}(g,g)^{abc}\right] \geq \varepsilon$$

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

3. MODEL OF CONDITIONAL PROXY RE-ENCRYPTION

We give the definitions and security notions for C-PRE systems in this section.

3.1. Definition of C-PRE systems

A unidirectional C-PRE scheme consists of seven algorithms which are described as follows:

Global Setup (λ): The global setup algorithm takes a security parameter λ as input and outputs the global parameters *param*. The parameters in *param* are implicitly given as input to the following algorithms.

KeyGen (i): The key generation algorithm takes the user index i as input and generates a public key(pk_i) and a secret key(sk_i) for user U_i.

ReKeyGen (sk_i, pk_j): The partial re-encryption key generation algorithm takes a secret key sk_i and another public key pk_j as input and outputs the partial re-encryption key $rk_{i \to j}$. This algorithm is run by U_i. Here sk_j is not taken as input which indeed makes the scheme unidirectional.

CKeyGen (sk_i, w): The condition key generation algorithm takes a secret key sk_i and a condition w as input and outputs the condition key $ck_{i, w}$. This algorithm is run by U_i.

Encrypt (pk, m, w): The encryption algorithm takes a public key pk, a message m and a condition w as input and outputs the ciphertext ζ associated with w under pk. Here $m \in \mathcal{M}$ where $\mathcal{M}$ denotes the message space.

ReEncrypt ($rk_{i \to j}$, $ck_{i, w}$,ζ_i): The re-encryption algorithm takes a partial re-encryption key $rk_{i \to j}$, a condition key $ck_{i, w}$ associated with condition w and a ciphertext ζ_i under the public key pk_i as input and outputs the re-encrypted ciphertext ζ_j under the public key pk_j. This algorithm is run by the proxy.

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

Correctness: For any $m \in \mathcal{M}$, any condition w, any $(pk_i, sk_i) \leftarrow$ KeyGen (i), $(pk_j, sk_j) \leftarrow$ KeyGen (j), and
ζ_i = Encrypt (pk_i, m, w),
Pr [Decrypt(sk_i, ζ_i) = m] = 1, and
Pr [Decrypt(sk_j, ReEncrypt ($rk_{i, j}$, $ck_{i, w}$, ζ_i)) = m] = 1.
while for any other condition w′ and user j′ with $w' \neq w$ and $j' \neq j$, we have
Pr [Decrypt(sk_j, ReEncrypt ($rk_{i, j}$, $ck_{i, w'}$, ζ_i)) = $\perp$] = 1−neg(λ)
Pr [Decrypt(sk_j, ReEncrypt ($rk_{i, j'}$, $ck_{i, w}$, ζ_i)) = $\perp$] = 1−neg(λ).

3.2 Security Notions

The following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ is used to define the semantic security of our C-PRE scheme against chosen ciphertext attacks.

Setup. $\mathcal{C}$ takes a security parameter λ and runs the algorithm GlobalSetup(λ) and gives the resulting global parameters *param* to $\mathcal{A}$.
Phase 1. $\mathcal{A}$ adaptively issues queries q_1, …, q_m where q_i is one of the following:

- *Uncorrupted key generation query:* $\mathcal{C}$ first runs algorithm KeyGen (i) to obtain the public/secret key pair (pk_i, sk_i), and then gives pk_i to $\mathcal{A}$.
- *Corrupted key generation query:* $\mathcal{C}$ first runs algorithm KeyGen (j) to obtain the public/secret key pair (pk_j, sk_j), and then gives (pk_j, sk_j) to $\mathcal{A}$.
- *Partial re-encryption key generation query* (pk_i, pk_j): $\mathcal{C}$ runs the algorithm ReKeyGen(sk_i, pk_j) and returns the generated re-encryption key $rk_{i \to j}$ to $\mathcal{A}$. Here sk_i is the secret key corresponding to pk_i.
- *Condition key generation query* (pk_i, w): $\mathcal{C}$ runs the algorithm CKeyGen(sk_i, w) and returns the generated condition key $ck_{i, w}$ to $\mathcal{A}$.

- *Re-encryption query* (pk_i, pk_j, w, ζ_i): $\mathcal{C}$ runs the algorithm ReEncrypt(ReKeyGen(sk_i, pk_j), CKeyGen(sk_i, w), ζ_i) and returns the generated ciphertext ζ_j to A.
- *Decryption query* (pk, w, ζ) or (p_k, ζ): $\mathcal{C}$ runs the algorithm Decrypt(sk, ζ) and returns its result to $\mathcal{A}$. Here (pk, w, ζ) and (pk, ζ) are queries on original ciphertexts and re-encrypted ciphertexts respectively.

For the last four queries it is required that pk, pk_i and pk_j are generated beforehand by the KeyGen algorithm.

Challenge. Once $\mathcal{A}$ decides Phase 1 is over, it outputs a target public key pk_{i*}, a target condition w^* and two equal-length plaintexts m_0, $m_1 \in \mathcal{M}$. $\mathcal{C}$ flips a random coin $\delta \in \{0, 1\}$, and sets the challenge ciphertext to be $\zeta^* = \text{Encrypt}(pk_{i*}, m_\delta, w^*)$, which is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ adaptively issues queries as in Phase 1, and $\mathcal{C}$ answers them as before.

Guess: Finally, $\mathcal{A}$ outputs a guess $\delta' \in \{0, 1\}$ and wins the game if $\delta' = \delta$. Adversary $\mathcal{A}$ is subject to the following restrictions during the above game.

1. $\mathcal{A}$ cannot issue corrupted key generation queries on i* to obtain the target secret key sk_{i*}.
2. $\mathcal{A}$ can issue decryption queries on neither (pk_{i*}, w^*, ζ^*) nor (pk_j, ReEncrypt($rk_{i*} \rightarrow j$, $ck_{i*,w*}$, ζ^*)).
3. $\mathcal{A}$ cannot issue re-encryption queries on (pk_{i*}, pk_j, w^*, ζ^*) if pk_j appears in a previous corrupted key generation query.
4. $\mathcal{A}$ cannot obtain the partial re-encryption key $rk_{i* \rightarrow j}$ if pk_j appears in a previous corrupted key generation query.

We refer to the above adversary $\mathcal{A}$ as an IND-CPRE-CCA adversary. $\mathcal{A}$'s advantage in attacking our CPRE scheme is defined as $\text{Adv}_{\text{C-PRE},\mathcal{A}}^{\text{IND-CPRE-CCA}} = \left|\Pr[\delta' = \delta] - 1/2\right|$, where the probability is taken over the random coins consumed by the adversary and the challenger. As in [14], we also distinguish between two types of IND-CPRE-CCA adversaries as follows:

- Type I IND-CPRE-CCA adversary: In the game, adversary $\mathcal{A}$ does not obtain the re-encryption key $rk_{i* \rightarrow j}$ with pk_j corrupted.
- Type II IND-CPRE-CCA adversary: In the game, adversary $\mathcal{A}$ does not obtain both the condition key $ck_{i*, w*}$ and the re-encryption key $rk_{i* \rightarrow j}$ with pk_j corrupted.

4. AN EFFICIENT C-PRE SCHEME

Here we present our efficient C-PRE scheme and then prove its security.

4.1 Construction

Our proposed scheme consists of the following seven main algorithms and one auxiliary algorithm for checking the validity of the ciphertext.

Global Setup (λ) : This algorithm takes the security parameter λ as input. Then two primes p and q are chosen such that $q \mid p-1$ where q is a λ bit prime. Then the algorithm generates (q, $\mathbb{G}$, $\mathbb{G}_T$, e) where $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups with prime order q and e is a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. Let g be the generator of group $\mathbb{G}$, which is a subgroup of $\mathbb{Z}_q^*$ with order q. Choose hash functions as follows:

$H_1 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G} \rightarrow \{0,1\}^{l_0+l_1}$, $H_4 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_5 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_6 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}$, and $H_7 : \mathbb{G}_T \rightarrow \{0,1\}^{l_0+l_1}$.

param = ((q, $\mathbb{G}$, $\mathbb{G}_T$, e), g, H_1, ..., H_7). l_0 and l_1 are determined by λ and the message space $\mathcal{M}$ is $\{0,1\}^{l_0}$.

KeyGen (i): This algorithm randomly picks $sk_i = (x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*)$ and sets $pk_i = (g^{x_{i,1}}, g^{x_{i2}})$.

ReKeyGen(sk_i, pk_j) : The re-encryption key $rk_{i \rightarrow j}$ is generated as follows:

1. Pick $h \xleftarrow{\$} \{0,1\}^{l_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{l_1}$ and compute $v = H_1(h, \pi)$.
2. Compute $V = g^v$ and $W = H_3(pk_{j,2}^v) \oplus (h \| \pi)$.
3. Compute $rk_{i \rightarrow j}^{(1)} = \dfrac{h}{x_{i,1} H_5(pk_{i,2}) + x_{i,2}}$ and return $rk_{i \rightarrow j} = (rk_{i \rightarrow j}^{(1)}, V, W)$.

CKeyGen(sk_i, w) : This algorithm outputs the condition key $ck_{i,w} = H_6(w, pk_i)^{\frac{1}{x_{i,1}}}$.

Encrypt(pk_i, m, w) : This algorithm encrypts a message m with condition w for pk_i as follows:

1. Pick s, $z \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $B = pk_{i,1}^s$ and $D = pk_{i,1}^z$.
2. Pick $r' \xleftarrow{\$} \{0,1\}^{l_1}$. Compute $r = H_2(m, r', pk_i, w)$ and $A = (pk_{i,1}^{H_5(pk_{i,2})} pk_{i,2})^r$.
3. Compute $C = H_3(g^r) \oplus (m \| r') \oplus H_7(\hat{e}(g, H_6(w, pk_i))^s)$.
4. Compute $E = s + zH_4(A, B, C, D) \bmod q$.
5. Output the ciphertext $\zeta_i = (A, B, C, D, E)$.

Validity(): This algorithm implicitly takes all the inputs of the calling algorithm as its input and works as follows:

If $pk_{i,1}^E \neq B.D^{H_4(A,B,C,D)}$ return $\perp$.

ReEncrypt($rk_{i \rightarrow j}$, $ck_{i,w}$, ζ_i, pk_i, pk_j): This algorithm re-encrypts ζ_i to ζ_j as follows:

1. Return $\perp$ if Validity() returns $\perp$.
2. Compute $A' = A^{rk_{i \rightarrow j}^{(1)}}$ and $C' = C \oplus H_7(\hat{e}(B, ck_{i,w}))$.
3. Output the transformed ciphertext as $\zeta_j = (A', C', V, W)$.

Decrypt(sk_i, ζ_i): Parse the ciphertext ζ_i. Decryption of ζ_i is done as follows:

- ζ is the original ciphertext in the form $\zeta = (A, B, C, D, E)$.
  1. Return $\perp$ if Validity() returns $\perp$.
  2. Compute $(m \| r') = C \oplus H_3(A^{\frac{1}{x_{i,1} H_5(pk_{i,2}) + x_{i,2}}}) \oplus H_7(\hat{e}(B, H_6(w, pk_i))^{\frac{1}{x_{i,1}}})$.
  3. If $A = (pk_{i,1}^{H_5(pk_{i,2})} pk_{i,2})^{H_2(m, r', pk_i, w)}$ holds, return m; else return $\perp$.
- ζ is the re-encrypted ciphertext in the form $\zeta = (A', C', V, W)$.
  1. Compute $(h \| \pi) = W \oplus H_3(V^{sk_{i,2}})$ and $(m \| r') = C' \oplus H_3(A'^{\frac{1}{h}})$.
  2. If $V = g^{H_1(h, \pi)}$ and $A' = g^{hH_2(m, r', pk_i, w)}$ hold, return m; else return $\perp$.

Correctness: The proxy must have both the right re-encryption key and the condition key to re-encrypt a ciphertext to the delegatee. Otherwise, the delegatee will not be able to decrypt the ciphertext with non-negligible probability. Suppose a proxy has the re-encryption key $rk_{i \to j}$ and the condition key $ck_{i, w'}$ ($w' \neq w$), he will generate the re-encrypted ciphertext $\zeta_j = (A', C', V, W)$ as

$$A' = g^{rh}$$

$$\begin{aligned} C' &= H_3(g^r) \oplus (m \| r') \oplus H_7(\hat{e}(g, H_6(w, pk_i))^s) \oplus H_7(\hat{e}(B, ck_{i,w'})) \\ &= H_3(g^r) \oplus (m \| r') \oplus H_7(\hat{e}(g, H_6(w, pk_i))^s) \oplus H_7(\hat{e}(g^{sx_{i,1}}, H_6(w', pk_i)^{\frac{1}{x_{i,1}}})) \\ &= H_3(g^r) \oplus (m \| r') \oplus H_7(\hat{e}(g, H_6(w, pk_i))^s) \oplus H_7(\hat{e}(g, H_6(w', pk_i))^s) \end{aligned}$$

$$V = g^v$$

$$W = H_3(pk_{j,2}^v) \oplus (h \| \pi).$$

Note that the two H_7 terms do not cancel each other implying that $C' \oplus H_3(A'^{\frac{1}{g}})$ in the decryption algorithm will not reveal the message m with overwhelming probability. The resulting value will also not pass the condition checks. Hence the delegatee cannot decrypt the re-encrypted ciphertext with high probability.

Security intuitions: It is impossible for the adversary to manipulate the ciphertext. This is because the validity of the original ciphertext can be publicly verified by the Validity() algorithm. Thus our scheme can ensure chosen-ciphertext security. Even if the conditional key w is changed to another value w′ by the adversary, the scheme is secure because w is a parameter for H_2 and when w changes the value of r also changes.

4.2. Security

The proposed C-PRE scheme is IND-CPRE-CCA secure in random oracle model. This follows directly from Theorem 1 and Theorem 2.

Theorem 1. Our scheme is IND-CPRE-CCA secure in the random oracle model, assuming the mCDH assumption holds in group $\mathbb{G}$ and the Schnorr signature is EUF-CMA secure. Concretely, if there exists a Type I adversary $\mathcal{A}$, who asks at most q_{H_i} random oracle queries to H_i with $i \in \{1, 2, ..., 7\}$, and breaks the $(t, q_u, q_c, q_{rk}, q_{ck}, q_{re}, q_d, \varepsilon)$-IND-CPRE-CCA of our scheme, then, for any $0 < \psi < \varepsilon$, there exists

1. either an algorithm $\mathcal{B}$ which can break the (t', ε')-mCDH assumption in $\mathbb{G}$ with
$$\begin{aligned} t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6} + q_{H_7} + q_u + q_c + q_{rk} + q_{ck} + q_{re} + q_d) O(1) \\ & + (2q_c + 2q_u + 6q_{rk} + q_{ck} + (q_{re} + 1)(2q_d + (2q_{H_2} + 2q_{H_3})q_d))\, t_{exp} + (q_{re} + q_d)\, t_p \end{aligned}$$
$$\varepsilon' \geq \frac{1}{q_{H_3}} \left(\frac{\varepsilon - \psi}{\varepsilon(1 + q_{rk})} - \frac{q_{H_2} + q_{H_4} + (q_{H_2} + q_{H_3})(q_{re} + q_d)}{2^{l_0 + l_1}} - \frac{2(q_{re} + q_d)}{q} \right)$$
where t_{exp} denotes the running time of an exponentiation in group $\mathbb{G}$ and t_p denotes the running time of a pairing in groups $(\mathbb{G}, \mathbb{G}_T)$.
2. or an attacker who breaks the EUF-CMA security of the Schnorr signature with advantage ψ within time t'.

PROOF. Without loss of generality, we assume that the Schnorr signature is (t′, ψ)-EUF-CMA secure for some probability $0 < \psi < \varepsilon$. If there exists a t-time adversary $\mathcal{A}$ who can break the IND-CPRE-CCA security of our scheme with advantage ε-ψ, then we show how to construct an algorithm $\mathcal{B}$ which can break the (t′, ε′)-mCDH assumption in $\mathbb{G}$. Suppose $\mathcal{B}$ is given as input a challenge tuple $(g, g^{\frac{1}{a}}, g^a, g^b)$ with unknown $a, b \xleftarrow{\$} \mathbb{Z}_q^*$. Algorithm $\mathcal{B}$'s goal is to output g^{ab}. Algorithm $\mathcal{B}$ first gives (q, $\mathbb{G}$, g, H_1, ..., H_7, l_0, l_1) to $\mathcal{A}$. Next, $\mathcal{B}$ acts as a challenger and plays the IND-CPRE-CCA game with adversary $\mathcal{A}$ in the following way:

Hash Oracle Queries: At any time adversary $\mathcal{A}$ can issue random oracle queries H_i with $i \in \{1, ..., 7\}$. Algorithm $\mathcal{B}$ maintains seven hash lists H_i^{list} with $i \in \{1, ..., 7\}$ which are initially empty, and responds as below:

1. H_1 *queries*: If $H_1(h, \pi)$ has appeared on the H_i^{list} in a tuple (h, π, v), return the predefined value v. Otherwise, choose $v \xleftarrow{\$} \mathbb{Z}_q^*$ and add tuple (h, π, v) to H_i^{list} and respond with $H_1(h, \pi) = v$.
2. H_2 *queries*: If $H_2(m, r', pk_i, w)$ has appeared on the H_2^{list} in a tuple (m, r′, pk_i, w, r), return the predefined value r. Otherwise, choose $r \xleftarrow{\$} \mathbb{Z}_q^*$ and add tuple (m, r′, pk_i, w, r) to H_2^{list} and respond with $H_2(m, r', pk_i, w) = r$.
3. H_3 *queries*: If $H_3(R)$ has appeared on the H_3^{list} [$R \in \mathbb{G}$] in a tuple (R, β), return the predefined value β. Otherwise, choose $\beta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add tuple (R, β) to H_3^{list} and respond with $H_3(R) = \beta$.
4. H_4 *queries*: If $H_4(A, B, C, D)$ has appeared on the H_4^{list} in a tuple (A, B, C, D, γ), return the predefined value γ. Otherwise, choose $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$, add tuple (A, B, C, D, γ) to H_4^{list} and respond with $H_4(A, B, C, D) = \gamma$.
5. H_5 *queries*: If $H_5(pk)$ has appeared on the H_5^{list} in a tuple (pk, τ), return the predefined value τ. Otherwise, choose $\tau \xleftarrow{\$} \mathbb{Z}_q^*$, add tuple (pk, τ) to H_5^{list} and respond with $H_5(pk) = \tau$.
6. H_6 *queries*: If $H_6(w, pk)$ has appeared on the H_6^{list} in a tuple (w, pk, t, S), return the predefined value S. Otherwise, choose $t \xleftarrow{\$} \mathbb{Z}_q^*$, compute $S = g^t$, add the tuple (w, pk, t, S) to H_6^{list} and respond with $H_6(w, pk) = S$.
7. H_7 *queries*: If $H_7(U)$ has appeared on the H_7^{list} [$U \in \mathbb{G}_T$] in a tuple (U, η), return the predefined value η. Otherwise, choose $\eta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add tuple (U, η) to H_7^{list} and respond with $H_7(U) = \eta$.

Phase 1. In this phase, adversary $\mathcal{A}$ issues a series of queries subject to the restrictions of the Type I IND-CPRE-CCA game. $\mathcal{B}$ maintains three lists K^{list}, R^{list} and C^{list} which are initially empty, and answers these queries for $\mathcal{A}$ as follows:

- *Uncorrupted key generation query.* $\mathcal{B}$ picks $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$. Next, using the Coron's technique [6], it flips a biased coin $c_i \in \{0, 1\}$ that yields 1 with probability θ and 0 otherwise. If $c_i = 1$, it defines $pk_i = (g^{x_{i,1}}, g^{x_{i,2}})$; else $pk_i = ((g^{\frac{1}{a}})^{x_{i,1}}, (g^{\frac{1}{a}})^{x_{i,2}})$. Then, it adds the tuple $(pk_i, x_{i,1}, x_{i,2}, c_i)$ to K^{list} and returns pk_i.
- *Corrupted key generation query.* $\mathcal{B}$ picks $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$ and defines $pk_i = (g^{x_{i,1}}, g^{x_{i,2}})$, c_i = '–'. Then, it adds the tuple $(pk_i, x_{i,1}, x_{i,2}, c_i)$ to K^{list} and returns $(pk_i, (x_{i,1}, x_{i,2}))$.
- *Re-encryption key generation query* (pk_i, pk_j). If R^{list} has an entry for (pk_i, pk_j), return the predefined re-encryption key to $\mathcal{A}$. Otherwise, algorithm $\mathcal{B}$ acts as follows:
  1. Recover tuples $(pk_i, x_{i,1}, x_{i,2}, c_i)$ and $(pk_j, x_{j,1}, x_{j,2}, c_j)$ from K^{list}.
  2. Pick $h \xleftarrow{\$} \{0,1\}^{l_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{l_1}$; compute $v = H_1(h, \pi)$, $V = g^v$ and $W = H_3(pk_{j,2}^v) \oplus (h \| \pi)$.
  3. Construct the first component $rk_{i\to j}^{(1)}$ according to the following cases:
     - $c_i = 1$ or c_i = '–': Define $rk_{i\to j}^{(1)} = \dfrac{h}{x_{i,1} H_5(pk_{i,2}) + x_{i,2}}$.
     - $(c_i = 0 \wedge c_j = 1)$ or $(c_i = 0 \wedge c_j = 0)$: Pick $rk_{i\to j}^{(1)} \xleftarrow{\$} \mathbb{Z}_q^*$.
     - $(c_i = 0 \wedge c_j$ = '–'): Output "failure" and **abort**.
  4. If $\mathcal{B}$ does not **abort**, add $(pk_i, pk_j, (rk_{i\to j}^{(1)}, V, W), h)$ into list R^{list}, return $(rk_{i\to j}^{(1)}, V, W)$.
- *Condition key query* (pk_i, w). If C^{list} has an entry for (pk_i, w), return the predefined condition key $ck_{i,w}$ to $\mathcal{A}$. Otherwise algorithm $\mathcal{B}$ acts as follows:
  1. Recover tuples $(pk_i, x_{i,1}, x_{i,2}, c_i)$ from K^{list} and (w, pk_i, t, S) from H_6^{list}.
  2. It constructs the condition key $ck_{i,w}$ for adversary $\mathcal{A}$ according to the following cases:
     - $c_i = 1$ or c_i = '–': Algorithm $\mathcal{B}$ responds with $ck_{i,w} = S^{\frac{1}{x_{i,1}}}$.
     - $c_i = 0$: Algorithm $\mathcal{B}$ responds with $ck_{i,w} = (g^a)^{\frac{1}{x_{i,1}}}$ which is same as $S^{\frac{1}{sk_{i,1}}}$.
  3. Add $(pk_i, w, ck_{i,w})$ to C^{list}.
- *Re-encryption query* $(pk_i, pk_j, w, \zeta_i)\}$. Algorithm $\mathcal{B}$ parses $\zeta_i = (A, B, C, D, E)$. Return $\perp$ if Validity() returns $\perp$. Otherwise it constructs the condition key $ck_{i,w}$ by issuing a condition key query (pk_i, w) and does the following:
  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^{list}.
  2. If $(c_i = 0 \wedge c_j$ = '–') does not hold, issue a re-encryption key generation query (pk_i, pk_j) to obtain $rk_{i\to j}$, and then ReEncrypt($rk_{i\to j}$, $ck_{i,w}$, ζ_i, pk_i, pk_j) to $\mathcal{A}$.
  3. Else $\mathcal{B}$ does the following.
     - Pick $h \xleftarrow{\$} \{0,1\}^{l_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{l_1}$ and compute $v = H_1(h, \pi)$.
     - Compute $V = g^v$ and $W = H_3(pk_{j,2}^v) \oplus (h \| \pi)$.
     - Since the ciphertext is valid, issue a decryption query (pk_i, ζ_i) and get message m.
     - Pick $r' \xleftarrow{\$} \{0,1\}^{l_1}$. Compute $r = H_2(m, r', pk_i, w)$.
     - Compute $A' = g^{rh}$ and $C' = H_3(g^r) \oplus (m \| r')$.

- Add tuple (h, π, v) to H_1^{list} and tuple (m, r', pk_i, w, r) to H_2^{list}, if they are not present in their respective lists.
- Return (A', C', V, W) to A as the re-encrypted ciphertext.

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

- ζ_i is an original ciphertext $\zeta_I = (A, B, C, D, E)$:
  1. Return $\perp$ if Validity() returns $\perp$.
  2. Construct condition key $ck_{i,w'}$ as in the condition key query and define $C = C' \oplus H_7(\hat{e}(B, ck_{i,w'}))$.
  3. Search tuples $(m, r', pk, w, r) \in H_2^{list}$ and $(R, \beta) \in H_3^{list}$ such that $pk_i = pk$, $w = w'$, $\beta \oplus (m \| r') = C$, $g^r = R$ and $(pk_{i,1}^{H_5(pk_{i,2})} pk_{i,2})^r = A$.
  4. If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.
- ζ_i is a re-encrypted ciphertext $\zeta_i = (A', C', V, W)$:
  1. Search tuples $(m, r', pk, w, r) \in H_2^{list}$, $(h, \pi, v) \in H_1^{list}$, $(R, \beta) \in H_3^{list}$ and $(R', \beta') \in H_3^{list}$ such that $pk_i = pk$, $w = w'$, $g^{rh} = A'$, $g^r = R$, $\beta \oplus (m \| r') = C'$, $g^v = V$, $\beta' \oplus (h \| \pi) = W$ and $pk_{i,2}^v = R'$.
  2. If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a public key $pk_{i^*} = (pk_{i^*,1}, pk_{i^*,2})$, a condition w^* and two equal-length messages $m_0, m_1 \in \{0,1\}^{l_0+l_1}$. Algorithm $\mathcal{B}$ responds as follows:

1. Recover tuple $(pk_{i^*}, x_{i^*,1}, x_{i^*,2}, c^*)$ from K^{list}. If $c^* \neq 0$, $\mathcal{B}$ outputs "failure" and **aborts**. Otherwise, $\mathcal{B}$ proceeds to execute the following steps.
2. Pick $s^*, z^* \xleftarrow{\$} \mathbb{Z}_q^*$ and compute $B^* = (g^{\frac{1}{a}})^{s^* x_{i^*,1}}$ and $D^* = (g^{\frac{1}{a}})^{z^* x_{i^*,1}}$.
3. Pick $C^* \xleftarrow{\$} \{0,1\}^{l_0+l_1}$.
4. Compute $A^* = (g^b)^{x_{i^*,1} H_5(pk_{i^*,2}) + x_{i^*,2}}$ and $E^* = s^* + z^* H_4(A^*, B^*, C^*, D^*) \bmod q$.
5. Construct the condition key ck_{i^*,w^*}, as in the condition key query.
6. Pick a random bit $\delta \xleftarrow{\$} \{0,1\}$ and $r' \xleftarrow{\$} \{0,1\}^{l_1}$. Implicitly define $H_2(m_\delta, r', pk_{i^*}, w^*) = ab$ and
   $H_3(g^{ab}) = C^* \oplus (m_\delta \| r') \oplus H_7(\hat{e}(B^*, ck_{i^*,w^*}))$ (note that $\mathcal{B}$ knows neither ab nor g^{ab}).
7. Return $\zeta^* = (A^*, B^*, C^*, D^*, E^*)$ as the challenged ciphertext to adversary A.

Observe that the challenge ciphertext ζ^* is identically distributed as the real one from the construction. To see this, letting $r^* = ab$, we have

$$A^* = (g^b)^{x_{i^*,1} H_5(pk_{i^*,2}) + x_{i^*,2}} = ((g^{\frac{1}{a}})^{x_{i^*,1} H_5(pk_{i^*,2}) + x_{i^*,2}})^{ab}$$
$$= (pk_{i^*,1}^{H_5(pk_{i^*,2})} pk_{i^*,2})^{r^*}$$
$$B^* = ((g^{\frac{1}{a}})^{s^* x_{i^*,1}} = pk_{i^*,1}^{s^*}$$
$$C^* = H_3(g^{ab}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(B^*, ck_{i^*,w^*}))$$
$$= H_3(g^{ab}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}((g^{\frac{1}{a}})^{s^* x_{i^*,1}}, (g^a)^{\frac{1}{x_{i^*,1}}}))$$

$= H_3(g^{ab}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(g, g^y)^{s^*})$

$= H_3(g^{r^*}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(g, H_6(w^*, pk_{i^*,w^*}))^{s^*})$

$D^* = (g^{\frac{1}{a}})^{z^* x_{i^*,1}} = pk_{i^*,1}^{z^*}$

$E^* = s^* + z^* H_4(A^*, B^*, C^*, D^*) \bmod q$

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

Guess. Eventually, adversary $\mathcal{A}$ returns a guess $\delta' \in \{0, 1\}$ $\mathcal{B}$. Algorithm $\mathcal{B}$ randomly picks a tuple (R, β) from the H_3^{list} and outputs R as the solution to the given problem instance.

Analysis. Now let's analyse the simulation. From the constructions of H_1, H_5, H_6 and H_7, it is clear that the simulations of these oracles are perfect. Let $AskH_4^*$ be the event that A queried (A^*, B^*, C^*, D^*) to H_4 before challenge phase. The simulation of H_4 is perfect as long as $AskH_4^*$ did not occur. Since C^* is randomly chosen from $\{0,1\}^{l_0+l_1}$ by the challenger in the challenge phase, we have $\Pr[AskH_4^*] = \frac{q_{H_4}}{2^{l_0+l_1}}$. Let $AskH_2^*$ be the event that $(m_\delta, r', pk_{i^*}, w^*)$ has been queried to H_2 and $AskH_3^*$ be the event that g^{ab} has been queried to H_3. The simulations of H_2 and H_3 are perfect as long as $AskH_2^*$ and $AskH_3^*$ did not occur, where δ and r' are chosen by $\mathcal{B}$ in the challenge phase.

$\mathcal{B}$'s responses to $\mathcal{A}$'s uncorrupted/corrupted key generation queries are perfect. Let Abort denote the event of $\mathcal{B}$'s aborting during the simulation of the re-encryption key queries or in the challenge phase. We have $\Pr[\neg Abort] \geq \theta^{q_{rk}}(1-\theta)$, which is maximized at $\theta_{opt} = \frac{q_{rk}}{1+q_{rk}}$. Using θ_{opt}, the probability $\Pr[\neg Abort]$ is at least $\frac{1}{e(1+q_{rk})}$.

The simulation of the re-encryption key queries is same as the real one, except for the case ($c_i = 0 \wedge c_j = 1$) or ($c_i = 0 \wedge c_j = 0$), in which the component $rk_{i\to j}^{(1)}$ is randomly chosen. If Abort does not happen, this is computationally indistinguishable from the real world because:

1. Secret key sk_j is unknown to $\mathcal{A}$ since $c_j \neq$ '–'.
2. h is encrypted under pk_j using the "hashed" ElGamal encryption scheme. So, if $\mathcal{A}$ can distinguish $rk_{i\to j}$ from $rk'_{i\to j}$, it means that $\mathcal{A}$ can determine (V, W) is an encryption of h or h′, which breaks the CCA security of the "hashed" ElGamal based on the CDH assumption.

The re-encryption queries are also perfect, unless $\mathcal{A}$ can submit valid original ciphertexts without querying H_2 or H_3 (denote this event by REErr). This is because we issue a decryption query in the third case of the re-encryption query. We will calculate Pr[REErr] shortly.

The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. $\mathcal{A}$ can submit valid original ciphertexts without querying H_2 or H_3 (denote this event by DErr). Let Valid be the event that the ciphertext is

valid. Let $AskH_3$ and $AskH_2$ be the events g^r has been queried to H_3 and (m, r′, w) has been queried to H_2 respectively. We have,

$$\begin{aligned}\Pr[\text{Valid} \mid \neg \text{AskH}_2] &= \Pr[\text{Valid} \wedge \text{AshH}_3 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \neg \text{AskH}_3 \mid \neg \text{AskH}_2] \\ &\leq \Pr[\text{AskH}_3 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \mid \neg \text{AskH}_3 \wedge \neg \text{AskH}_2] \\ &\leq \frac{q_{H_3}}{2^{l_0+l_1}} + \frac{1}{q}\end{aligned}$$

Similarly, we have $\Pr[\text{Valid} \mid \neg \text{AskH}_3] \leq \frac{q_{H_2}}{2^{l_0+l_1}} + \frac{1}{q}$. Thus we have,

$$\begin{aligned}\Pr[\text{Valid} \mid (\neg \text{AskH}_2 \vee \text{AskH}_3)] &\leq \Pr[\text{Valid} \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \mid \neg \text{AskH}_3] \\ &\leq \frac{q_{H_2} + q_{H_3}}{2^{l_0+l_1}} + \frac{2}{q}\end{aligned}$$

Let DErr be the event that Valid |(¬ $AskH_2$ ∨ ¬ $AskH_3$) happens during the entire simulation. Then since A issues utmost q_d decryption oracles, we have $\Pr[\text{DErr}] \leq \frac{(q_{H_2} + q_{H_3})q_d}{2^{l_0+l_1}} + \frac{2q_d}{q}$.

By the definition of REErr as stated above, since $\mathcal{A}$ issues utmost q_{re} re-encryption oracles, we have

$$\Pr[\text{REErr}] \leq \frac{(q_{H_2} + q_{H_3})q_{re}}{2^{l_0+l_1}} + \frac{2q_{re}}{q}.$$

Now, let Good denote the event

$$(\text{AskH}_3^* \vee (\text{AskH}_2^* \mid \neg \text{AskH}_3^*) \vee \text{AskH}_4^* \vee \text{REErr} \vee \text{DErr}) \mid \neg \text{Abort}.$$

If Good does not happen, due to the randomness of the output of the random oracle H_3, it is clear that $\mathcal{A}$ cannot gain any advantage greater than $\frac{1}{2}$ in guessing δ. Thus we have Pr[δ′ = δ | ¬ Good] = $\frac{1}{2}$. Hence by splitting Pr[δ′ = δ], we have

$$\begin{aligned}\Pr[\delta' = \delta] &= \Pr[\delta' = \delta \mid \neg \text{Good}]\Pr[\neg \text{Good}] + \Pr[\delta' = \delta \mid \text{Good}]\Pr[\text{Good}] \\ &\leq \tfrac{1}{2}\Pr[\neg \text{Good}] + \Pr[\text{Good}] \\ &\leq \tfrac{1}{2} + \tfrac{1}{2}\Pr[\text{Good}] \\ \Pr[\delta' = \delta] &\geq \Pr[\delta' = \delta \mid \neg \text{Good}]\Pr[\neg \text{Good}] \\ &= \tfrac{1}{2} - \tfrac{1}{2}\Pr[\text{Good}]\end{aligned}$$

By definition of the advantage for the IND-CPRE-CCA adversary, we then have

$$\begin{aligned}\varepsilon - \psi &= \left|2 \times \Pr[\delta' = \delta] - 1\right| \\ &\leq \Pr[\text{Good}] \\ &= \Pr[(\text{AskH}_3^* \vee (\text{AskH}_2^* \mid \neg \text{AskH}_3^*) \vee \text{AskH}_4^* \vee \text{REErr} \vee \text{DErr}) \mid \neg \text{Abort}] \\ &= \frac{\Pr[(\text{AskH}_3^*] + \Pr[\text{AskH}_2^* \mid \neg \text{AskH}_3^*] + \Pr[\text{AskH}_4^*]}{\Pr[\neg \text{Abort}]} + \frac{\Pr[\text{REErr}] + \Pr[\text{DErr}]}{\Pr[\neg \text{Abort}]}.\end{aligned}$$

Substituting values which have been computed, we get

$$\Pr[\mathrm{AskH}_3^*] \geq \Pr[\neg\mathrm{Abort}] \cdot (\varepsilon - \psi) - \Pr[\mathrm{AskH}_2^* \mid \neg\mathrm{AskH}_3^*] - \Pr[\mathrm{AskH}_4^*] - \Pr[\mathrm{Reerr}] - \Pr[\mathrm{Derr}]$$

$$\geq \frac{\varepsilon-\psi}{\varepsilon(1+q_{rk})} - \frac{q_{H_2}}{2^{l_0+l_1}} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{(q_{H_2}+q_{H_3})q_{re}}{2^{l_0+l_1}} - \frac{2q_{re}}{q} - \frac{(q_{H_2}+q_{H_3})q_d}{2^{l_0+l_1}} - \frac{2q_d}{q}$$

$$= \frac{\varepsilon-\psi}{\varepsilon(1+q_{rk})} - \frac{q_{H_2}+q_{H_4}+(q_{H_2}+q_{H_3})(q_{re}+q_d)}{2^{l_0+l_1}} - \frac{2(q_{re}+q_d)}{q}$$

If AskH_3^* happens, algorithm $\mathcal{B}$ will be able to solve mCDH instance. Therefore we get,

$$\varepsilon' \geq \frac{1}{q_{H_3}} \Pr[\mathrm{AskH}_3^*]$$

$$\geq \frac{1}{q_{H_3}} \left(\frac{\varepsilon-\psi}{\varepsilon(1+q_{rk})} - \frac{q_{H_2}+q_{H_4}+(q_{H_2}+q_{H_3})(q_{re}+q_d)}{2^{l_0+l_1}} - \frac{2(q_{re}+q_d)}{q} \right)$$

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

$$t' \leq t + (q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{H_6}+q_{H_7}+q_u+q_c+q_{rk}+q_{ck}+q_{re}+q_d)O(1)$$
$$+ (2q_c+2q_u+6q_{rk}+q_{ck}+(q_{re}+1)(2q_d+(2q_{H_2}+2q_{H_3})q_d))\, t_{exp} + (q_{re}+q_d)\, t_p$$

This completes the proof of Theorem 1.

Theorem 2. Our scheme is IND-CPRE-CCA secure in the random oracle model, assuming the mCBDH assumption holds in groups $\mathbb{G}$, $\mathbb{G}_T$ and the Schnorr signature is EUF-CMA secure. Concretely, if there exists a Type II adversary $\mathcal{A}$, who asks at most q_{H_i} random oracle queries to H_i with $i \in \{1, 2, ..., 7\}$, and breaks the $(t, q_u, q_c, q_{rk}, q_{ck}, q_{re}, q_d, \varepsilon)$-IND-CPRE-CCA of our scheme, then, for any $0 < \psi < \varepsilon$, there exists

1. either an algorithm $\mathcal{B}$ which can break the (t', ε')-mCBDH assumption in $\mathbb{G}$ with
$$t' \leq t + (q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{H_6}+q_{H_7}+q_u+q_c+q_{rk}+q_{ck}+q_{re}+q_d)O(1)$$
$$+ (2q_c+2q_u+6q_{rk}+q_{ck}+(q_{re}+1)(2q_d+(2q_{H_2}+2q_{H_3})q_d))\, t_{exp} + (q_{re}+q_d)\, t_p$$
$$\varepsilon' \geq \frac{1}{q_{H_7}} \left(\frac{\varepsilon-\psi}{\varepsilon(1+q_{rk})} + \frac{\varepsilon-\psi}{\varepsilon(1+q_{ck})} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{(q_{H_2}q_{H_3}+q_{H_3}q_{H_7}+q_{H_7}q_{H_2})(q_{re}+q_d)}{4^{l_0+l_1}} \right.$$
$$\left. - \frac{(q_{H_2}+q_{H_3}+q_{H_7})(q_{re}+q_d)}{2^{l_0+l_1-1}} - \frac{3(q_{re}+q_d)}{q} \right)$$
where t_{exp} denotes the running time of an exponentiation in group $\mathbb{G}$ and t_p denotes the running time of a pairing in groups $(\mathbb{G}, \mathbb{G}_T)$.
2. or an attacker who breaks the EUF-CMA security of the Schnorr signature with advantage ψ within time t'.

PROOF. Without loss of generality, we assume that the Schnorr signature is (t', ψ)-EUF-CMA secure for some probability $0 < \psi < \varepsilon$. If there exists a t-time adversary $\mathcal{A}$ who can break the IND-CPRE-CCA security of our scheme with advantage $\varepsilon-\psi$, then we show how to construct an algorithm $\mathcal{B}$ which can break the (t', ε')-mCBDH assumption in $\mathbb{G}$. Suppose $\mathcal{B}$ is given as input a challenge tuple $(g, g^{\frac{1}{a}}, g^a, g^b, g^c)$ with unknown $a, b, c \xleftarrow{\$} \mathbb{Z}_q^*$. Algorithm $\mathcal{B}$'s goal is to output $\hat{e}(g,g)^{abc}$. Algorithm $\mathcal{B}$ first gives $(q, \mathbb{G}, g, H_1, ..., H_7, l_0, l_1)$ to $\mathcal{A}$. Next, $\mathcal{B}$ acts as a challenger and plays the IND-CPRE-CCA game with adversary $\mathcal{A}$ in the following way:

Hash Oracle Queries. At any time adversary $\mathcal{A}$ can issue random oracle queries H_i with $i \in \{1, ..., 7\}$. Algorithm $\mathcal{B}$ maintains seven hash lists H_i^{list} with $i \in \{1, ..., 7\}$ which are initially empty, and responds as below:

- *H_1 queries*: If $H_1(h, \pi)$ has appeared on the H_1^{list} in a tuple (h, π, v), return the predefined value v. Otherwise, choose $v \xleftarrow{\$} \mathbb{Z}_q^*$ and add tuple (h, π, v) to H_1^{list} and respond with $H_1(h, \pi) = v$.
- *H_2 queries*: If $H_2(m, r', pk_i, w)$ has appeared on the H_2^{list} in a tuple (m, r', pk_i, w, r), return the predefined value r. Otherwise, choose $r \xleftarrow{\$} \mathbb{Z}_q^*$ and add tuple (m, r', pk_i, w, r) to H_2^{list} and respond with $H_2(m, r', pk_i, w) = r$.
- *H_3 queries*: If $H_3(R)$ has appeared on the H_3^{list} [$R \in G$] in a tuple (R, β), return the predefined value β. Otherwise, choose $\beta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add tuple (R, β) to H_3^{list} and respond with $H_3(R) = \beta$.
- *H_4 queries*: If $H_4(A, B, C, D)$ has appeared on the H_4^{list} in a tuple (A, B, C, D, γ), return the predefined value γ. Otherwise, choose $\gamma \xleftarrow{\$} \mathbb{Z}_q^*$, add tuple (A, B, C, D, γ) to H_4^{list} and respond with $H_4(A, B, C, D) = \gamma$.
- *H_5 queries*: If $H_5(pk)$ has appeared on the H_5^{list} in a tuple (pk, τ), return the predefined value τ. Otherwise, choose $\tau \xleftarrow{\$} \mathbb{Z}_q^*$, add tuple (pk, τ) to H_5^{list} and respond with $H_5(pk) = \tau$.
- *H_6 queries*: If $H_6(w, pk)$ has appeared on the H_6^{list} in a tuple $(w, pk, t, S, coin)$, return the predefined value S. Otherwise, choose $t \xleftarrow{\$} \mathbb{Z}_q^*$. Next, using the Coron's technique [6], flip a random biased coin [$coin \in \{0, 1\}$] that yields 0 with a probability θ and 1 with probability $1-\theta$. If coin = 0, compute $S = g^t$. Otherwise, compute $S = (g^b)^t$. Add the tuple $(w, pk, t, S, coin)$ to H_6^{list} and respond with $H_6(w, pk) = S$.
- *H_7 queries*: If $H_7(U)$ has appeared on the H_7^{list} [$U \in \mathbb{G}_T$] in a tuple (U, η), return the predefined value η. Otherwise, choose $\eta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add tuple (U, η) to H_7^{list} and respond with $H_7(U) = \eta$.

Phase 1. In this phase, adversary $\mathcal{A}$ issues a series of queries subject to the restrictions of the Type II IND-CPRE-CCA game. $\mathcal{B}$ maintains three lists K^{list}, R^{list} and C^{list} which are initially empty, and answers these queries for $\mathcal{A}$ as follows:

- Uncorrupted key generation query. $\mathcal{B}$ picks $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$. Next, it defines $c_i = 0$ and $pk_i = ((g^{\frac{1}{a}})^{x_{i,1}}, (g^{\frac{1}{a}})^{x_{i,2}})$. Then it adds the tuple $(pk_i, x_{i,1}, x_{i,2}, c_i)$ to K^{list} and returns pk_i. Here the bit c_i is used to denote whether the secret key with respect to pk_i is corrupted, i.e., $c_i = 0$ means uncorrupted and $c_i = 1$ means corrupted.
- Corrupted key generation query. $\mathcal{B}$ picks $x_{i,1}, x_{i,2} \xleftarrow{\$} \mathbb{Z}_q^*$ and defines $pk_i = (g^{x_{i,1}}, g^{x_{i,2}})$, $c_i = 1$. Then, it adds the tuple $(pk_i, x_{i,1}, x_{i,2}, c_i)$ to K^{list} and returns $(pk_i, (x_{i,1}, x_{i,2}))$.
- Re-encryption key generation query (pk_i, pk_j). If R^{list} has an entry for (pk_i, pk_j), return the predefined re-encryption key to $\mathcal{A}$. Otherwise, algorithm $\mathcal{B}$ acts as follows:
  1. Recover tuples $(pk_i, x_{i,1}, x_{i,2}, c_i)$ and $(pk_j, x_{j,1}, x_{j,2}, c_j)$ from K^{list}.

2. Pick $h \xleftarrow{\$} \{0,1\}^{l_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{l_1}$; compute $v = H_1(h, \pi)$, $V = g^v$ and $W = H_3(pk_{j,2}^v) \oplus (h \| \pi)$.
3. Construct the first component $rk_{i\to j}^{(1)}$ according to the following cases:
   - $c_i = 1$: Define $rk_{i\to j}^{(1)} = \frac{h}{x_{i,1}H_5(pk_{i,2}) + x_{i,2}}$.
   - $(c_i = 0 \wedge c_j = 0)$: Define $rk_{i\to j}^{(1)} = \frac{h}{x_{i,1}H_5(pk_{i,2}) + x_{i,2}}$. Here define $h = ah'$ so that $rk_{i\to j}^{(1)} = \frac{h'}{\frac{x_{i,1}H_5(pk_{i,2})+x_{i,2}}{a}}$ where $h' \in \{0,1\}^{l_0}$.
   - $(c_i = 0 \wedge c_j = 1)$: Output "failure" and **abort**.
4. If $\mathcal{B}$ does not **abort**, add $(pk_i, pk_j, (rk_{i\to j}^{(1)}, V, W), h)$ into list R^{list}, return $(rk_{i\to j}^{(1)}, V, W)$.

– Condition key query (pk_i, w). If C^{list} has an entry for (pk_i, w), return the predefined condition key $ck_{i,w}$ to $\mathcal{A}$. Otherwise algorithm $\mathcal{B}$ acts as follows:
1. Recover tuples (pk_i, $x_{i,1}$, $x_{i,2}$, c_i) from K^{list} and (w, pk_i, t, S, coin) from H_6^{list}.
2. It constructs the condition key $ck_{i,w}$ for adversary $\mathcal{A}$ according to the following cases:
   - $c_i = 1$: Algorithm $\mathcal{B}$ responds with $ck_{i,w} = S^{\frac{1}{x_{i,1}}}$.
   - $(c_i = 0 \wedge coin = 0)$: Algorithm $\mathcal{B}$ responds with $ck_{i,w} = (g^a)^{\frac{1}{x_{i,1}}}$ which is same as $S^{\frac{1}{sk_{i,1}}}$.
   - $(c_i = 0 \wedge coin = 1)$: Output "failure" and **abort**.
3. If $\mathcal{B}$ does not abort, add (pk_i, w, $ck_{i,w}$) to C^{list}.

– Re-encryption query (pk_i, pk_j, w, ζ_i). Algorithm $\mathcal{B}$ parses ζ_i = (A, B, C, D, E). Return $\perp$ if Validity() returns $\perp$. Otherwise it does the following:
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^{list}.
2. If ($c_i = 0$) does not hold, issue a condition key generation query (pk_i, w) to obtain $ck_{i,w}$ and a re-encryption key query (pk_i, pk_j) to obtain $rk_{i\to j}$, and then ReEncrypt($rk_{i\to j}$, $ck_{i,w}$, ζ_i, pk_i, pk_j) to $\mathcal{A}$.
3. Else $\mathcal{B}$ does the following.
   - Pick $h \xleftarrow{\$} \{0,1\}^{l_0}$ and $\pi \xleftarrow{\$} \{0,1\}^{l_1}$ and compute $v = H_1(h, \pi)$.
   - Compute $V = g^v$ and $W = H_3(pk_{j,2}^v) \oplus (h \| \pi)$.
   - Since the ciphertext is valid, issue a decryption query (pk_i, ζ_i) and get message m.
   - Pick $r' \xleftarrow{\$} \{0,1\}^{l_1}$. Compute $r = H_2(m, r', pk_i, w)$.
   - Compute $A' = g^{rh}$ and $C' = H_3(g^r) \oplus (m \| r')$.
   - Add tuple (h, π, v) to H_1^{list} and tuple (m, r', pk_i, w, r) to H_2^{list}, if they are not present in their respective lists.
   - Return (A', C', V, W) to $\mathcal{A}$ as the re-encrypted ciphertext.

– Decryption query (pk_i, w′, ζ_i) or (pk_i, ζ_i)}. $\mathcal{B}$ recovers tuple (pk_i, $x_{i,1}$, $x_{i,2}$, c) from K^{list} and (w, pk_i, t, S, coin) from H_6^{list}. If c = 1, algorithm B runs Decrypt(($x_{i,1}$, $x_{i,2}$), ζ_i)) and returns the result to $\mathcal{A}$. Otherwise, algorithm $\mathcal{B}$ works according to the following two cases:

- ζ_i is an original ciphertext $\zeta_i = (A, B, C, D, E)$:
  1. Return $\perp$ if Validity() returns $\perp$.
  2. Search tuples $(m, r', pk, w, r) \in H_2^{list}$, $(R, \beta) \in H_3^{list}$ and (U, γ) from H_7^{list} such that $pk_i = pk$, $w = w'$, $\beta \oplus (m \| r') \oplus \gamma = C$, $g^r = R$ and $(pk_{i,1}^{H_5(pk_{i,2})} pk_{i,2})^r = A$.
  3. If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.
- ζ_i is a re-encrypted ciphertext $\zeta_I = (A', C', V, W)$:
  1. Search tuples $(m, r', pk, w, r) \in H_2^{list}$, $(h, \pi, v) \in H_1^{list}$, $(R, \beta) \in H_3^{list}$ and $(R', \beta') \in H_3^{list}$ such that $pk_i = pk$, $w = w'$, $g^{rh} = A'$, $g^r = R$, $\beta \oplus (m \| r') = C'$, $g^v = V$, $\beta' \oplus (h \| \pi) = W$ and $pk_{i,2}^v = R'$.
  2. If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a public key $pk_{i^*} = (pk_{i^*,1}, pk_{i^*,2})$, a condition w^* and two equal-length messages $m_0, m_1 \in \{0,1\}^{l_0+l_1}$. Algorithm $\mathcal{B}$ responds as follows:

1. Recover tuple $(w^*, pk_{i^*}, t^*, S^*, coin^*)$ from H_6^{list}. If $coin^* \neq 1$, $\mathcal{B}$ outputs "failure" and **aborts**. Otherwise, $\mathcal{B}$ proceeds to execute the following steps.
2. Pick $u^*, e^* \overset{\$}{\leftarrow} \mathbb{Z}_q^*$ and compute $B^* = (g^c)^{\frac{x_{i^*,1}}{t}}$ and $D^* = ((B^*)^{-1} pk_{i^*,1}^{u^*})^{\frac{1}{e^*}}$.
3. Pick $C^* \overset{\$}{\leftarrow} \{0,1\}^{l_0+l_1}$ and $r' \overset{\$}{\leftarrow} \{0,1\}^{l_1}$.
4. Pick a random bit $\delta \overset{\$}{\leftarrow} \{0,1\}$ and compute $r^* = H_2(m_\delta, r', pk_{i^*}, w^*)$.
5. Compute $A^* = ((g^{\frac{1}{a}})^{x_{i^*,1}H_5(pk_{i^*,2})+x_{i^*,2}})^{r^*}$ and $E^* = u^*$.
6. Define $H_4(A^*, B^*, C^*, D^*) = e^*$.
7. Implicitly define $H_7(\hat{e}(g,g)^{abc}) = C^* \oplus (m_\delta \| r') \oplus H_3(g^r)$ (note that $\mathcal{B}$ does not know $\hat{e}(g,g)^{abc}$).
8. Note that $\hat{e}(g,g)^{abc} = \hat{e}(g^{\frac{cx_{i^*,1}}{t}}, g^{bt})^{\frac{a}{x_{i^*,1}}} = \hat{e}(B^*, H_6(w^*, pk_{i^*}))^{\frac{1}{sk_{i^*}}}$.
9. Return $\zeta^* = (A^*, B^*, C^*, D^*, E^*)$ as the challenged ciphertext to adversary $\mathcal{A}$.

Observe that the challenge ciphertext ζ^* is identically distributed as the real one from the construction. To see this, letting $s^* = \frac{ac}{t}$, we have

$$A^* = ((g^{\frac{1}{a}})^{x_{i^*,1}H_5(pk_{i^*,2})+x_{i^*,2}})^{r^*} = (pk_{i^*,1}^{H_5(pk_{i^*,2})} pk_{i^*,2})^{r^*}$$

$$B^* = (g^c)^{\frac{x_{i^*,1}}{t}} = (g^{\frac{x_{i^*,1}}{a}})^{\frac{ac}{t}} = pk_{i^*,1}^{s^*}$$

$$C^* = H_3(g^{r^*}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(g,g)^{abc})$$

$$= H_3(g^{r^*}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(g, g^{bt})^{\frac{ac}{t}})$$

$$= H_3(g^{r^*}) \oplus (m_\delta \| r') \oplus H_7(\hat{e}(g, H_6(w^*, pk_{i^*,w^*}))^{s^*})$$

$$D^* = ((B^*)^{-1} pk_{i^*,1}^{u^*})^{\frac{1}{e^*}}$$

$E^* = u^*$

Since u^* and e^* are random, adversary cannot distinguish D^* and E^* from the real one.

Phase 2. Adversary $\mathcal{A}$ continues to issue queries as in Phase 1, with the restrictions prescribed in the IND-CPRE-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 (U, γ) from the H_7^{list} and outputs U as the solution to the given problem instance.

Analysis. Now let's analyse the simulation. From the constructions of H_1, H_2, H_3, H_5 and H_6, it is clear that the simulations of these oracles are perfect. Let $AskH_4^*$ be the event that A queried (A*, B*, C*, D*) to H_4 before challenge phase. The simulation of H_4 is perfect as long as $AskH_4^*$ did not occur. Since C* is randomly chosen from $\{0,1\}^{l_0+l_1}$ by the challenger in the challenge phase, we have $\Pr[AskH_4^*] = \frac{q_{H_4}}{2^{l_0+l_1}}$. Let $AskH_7^*$ be the event that $\hat{e}(g,g)^{abc}$ has been queried to H_7. The simulation of H_7 is perfect as long as $AskH_7^*$ did not occur.

$\mathcal{B}$'s responses to $\mathcal{A}$'s uncorrupted/corrupted key generation queries are perfect. Let Abort denote the event of $\mathcal{B}$'s aborting during the simulation of the re-encryption key queries, condition key queries or in the challenge phase. We have $\Pr[\neg Abort\} \geq \theta^{q_{rk}}(1-\theta) + \theta^{q_{ck}}(1-\theta)$, which is maximized when each of the two terms are maximized. First term maximizes at $\theta_{opt1} = \frac{q_{rk}}{1+q_{rk}}$ and the second term maximizes at $\theta_{opt2} = \frac{q_{ck}}{1+q_{ck}}$. Thus the probability Pr[¬Abort] is at least $\frac{1}{e(1+q_{rk})} + \frac{1}{e(1+q_{ck})}$. Here we assume that probability of a key being uncorrupted is same as θ in H_6 queries and 1–θ if it is a corrupted one.

The simulation of the re-encryption key queries is same as the real one, except for the case ($c_i = 0 \wedge c_j = 0$), in which the component $rk_{i\to j}^{(1)}$ is chosen by choosing h randomly, where h is defined as ah′. If Abort does not happen, this is computationally indistinguishable from the real world because :

1. Secret key sk_j is unknown to $\mathcal{A}$ since $c_j \neq 1$.
2. h is encrypted under pk_j using the "hashed" ElGamal encryption scheme. So, if $\mathcal{A}$ can distinguish $rk_{i\to j}$ from $rk'_{i\to j}$, it means that $\mathcal{A}$ can determine (V, W) is an encryption of h or h′, which breaks the CCA security of the "hashed" ElGamal based on the CDH assumption.

The re-encryption queries are also perfect, unless $\mathcal{A}$ can submit valid original ciphertexts without querying H_2 or H_3 or H_7 (denote this event by REErr). This is because we issue a decryption query in the third case of the re-encryption query. We will calculate Pr[REErr] shortly.

The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. A can submit valid original ciphertexts without querying H_2 or H_3 or H_7 (denote this event by DErr). Let Valid be the event that the ciphertext is valid. Let $AskH_7$, $AskH_3$, and $AskH_2$ be the events $\hat{e}(g,g)^{abc}$ has been queried to H_7, g^r has been queried to H_3, and (m, r′, w) has been queried to H_2 respectively . We have,

$$\begin{aligned}
&\Pr[\text{Valid} \mid \neg \text{AskH}_2] \\
&\quad = \Pr[\text{Valid} \wedge \text{AskH}_3 \wedge \text{AskH}_7 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \text{AskH}_3 \wedge \neg \text{AskH}_7 \mid \neg \text{AskH}_2] \\
&\qquad + \Pr[\text{Valid} \wedge \neg \text{AskH}_3 \wedge \text{AskH}_7 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \text{AskH}_3 \wedge \neg \text{AskH}_7 \mid \neg \text{AskH}_2] \\
&\quad = \Pr[\text{Valid} \wedge \text{AskH}_3 \wedge \text{AskH}_7 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \wedge \text{AskH}_3 \mid \neg \text{AskH}_7 \wedge \neg \text{AskH}_2] \\
&\qquad + \Pr[\text{Valid} \wedge \neg \text{AskH}_7 \mid \neg \text{AskH}_3 \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \mid \neg \text{AskH}_3 \wedge \neg \text{AskH}_7 \wedge \neg \text{AskH}_2] \\
&\quad \le \frac{q_{H_3} q_{H_7}}{4^{l_0+l_1}} + \frac{q_{H_3} + q_{H_7}}{2^{l_0+l_1}} + \frac{1}{q}
\end{aligned}$$

Similarly, we have

$$\Pr[\text{Valid} \mid \neg \text{AskH}_3] \le \frac{q_{H_2} q_{H_7}}{4^{l_0+l_1}} + \frac{q_{H_2} + q_{H_7}}{2^{l_0+l_1}} + \frac{1}{q} \text{ and } \Pr[\text{Valid} \mid \neg \text{AskH}_7] \le \frac{q_{H_2} q_{H_3}}{4^{l_0+l_1}} + \frac{q_{H_2} + q_{H_3}}{2^{l_0+l_1}} + \frac{1}{q}$$

Thus we have,

$$\begin{aligned}
&\Pr[\text{Valid} \mid (\neg \text{AskH}_2 \vee \neg \text{AskH}_3 \vee \neg \text{AskH}_7)] \\
&\quad = \Pr[\text{Valid} \mid \neg \text{AskH}_2] + \Pr[\text{Valid} \mid \neg \text{AskH}_3] + \Pr[\text{Valid} \mid \neg \text{AskH}_7] \\
&\quad \le \frac{q_{H_2} q_{H_3} + q_{H_3} q_{H_7} + q_{H_7} q_{H_2}}{4^{l_0+l_1}} + \frac{q_{H_2} + q_{H_3} + q_{H_7}}{2^{l_0+l_1-1}} + \frac{3}{q}
\end{aligned}$$

Let DErr be the event that Valid|(¬ $\text{AskH}_2 \vee \neg$ $\text{AskH}_3 \vee \neg$ AskH_7) happens during the entire simulation. Then since $\mathcal{A}$ issues utmost q_d decryption oracles, we have

$$\Pr[\text{DErr}] \le \frac{(q_{H_2} q_{H_3} + q_{H_3} q_{H_7} + q_{H_7} q_{H_2}) q_d}{4^{l_0+l_1}} + \frac{(q_{H_2} + q_{H_3} + q_{H_7}) q_d}{2^{l_0+l_1-1}} + \frac{3q_d}{q}.$$

By the definition of REErr as stated above, since $\mathcal{A}$ issues utmost q_{re} re-encryption oracles, we have

$$\Pr[\text{REErr}] \le \frac{(q_{H_2} q_{H_3} + q_{H_3} q_{H_7} + q_{H_7} q_{H_2}) q_{re}}{4^{l_0+l_1}} + \frac{(q_{H_2} + q_{H_3} + q_{H_7}) q_{re}}{2^{l_0+l_1-1}} + \frac{3q_{re}}{q}.$$

Now, let Good denote the event $(\text{AskH}_7^* \vee \text{AskH}_4^* \vee \text{REErr} \vee \text{DErr}) \mid \neg \text{Abort}$. If Good does not happen, due to the randomness of the output of the random oracle H_7, it is clear that $\mathcal{A}$ cannot gain any advantage greater than $\frac{1}{2}$ in guessing δ. Thus we have $\Pr[\delta' = \delta \mid \neg \text{ Good}] = \frac{1}{2}$. Hence by splitting $\Pr[\delta' = \delta]$, we have

$$\begin{aligned}
&\Pr[\delta' = \delta] = \Pr[\delta' = \delta \mid \neg \text{Good}] \Pr[\neg \text{Good}] + \Pr[\delta' = \delta \mid \text{Good}] \Pr[\text{Good}] \\
&\qquad \le \tfrac{1}{2} \Pr[\neg \text{Good}] \Pr[\text{Good}] \\
&\qquad \le \tfrac{1}{2} + \tfrac{1}{2} \Pr[\text{Good}] \\
&\Pr[\delta' = \delta] \ge \Pr[\delta' = \delta \mid \neg \text{Good}] \Pr[\neg \text{Good}] \\
&\qquad = \tfrac{1}{2} - \tfrac{1}{2} \Pr[\text{Good}]
\end{aligned}$$

By definition of the advantage for the IND-CPRE-CCA adversary, we then have

$$\begin{aligned}\varepsilon - \psi &= \left|2\times \Pr[\delta' = \delta] - 1\right| \\ &\le \Pr[\text{Good}] \\ &= \Pr[(\text{AskH}_7^* \vee \text{AskH}_4^* \vee \text{REErr} \vee \text{DErr}) \mid \neg\text{Abort}] \\ &= \frac{\Pr[(\text{AskH}_7^* \vee \text{AskH}_4^* \vee \text{REErr} \vee \text{DErr}) \mid \neg\text{Abort}]}{\Pr[\neg\text{Abort}]}\end{aligned}$$

Substituting values which have been computed, we get

$$\begin{aligned}\Pr[\text{AskH}_7^*] &\ge \Pr[\neg\text{Abort}]\cdot(\varepsilon-\psi) - \Pr[\text{AskH}_4^*] - \Pr[\text{REErr}] - \Pr[\text{DErr}] \\ &\ge \frac{\in -\psi}{\varepsilon(1+q_{rk})} + \frac{\in -\psi}{\varepsilon(1+q_{ck})} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{(q_{H_2}q_{H_3} + q_{H_3}q_{H_7} + q_{H_7}q_{H_2})(q_{re}+q_d)}{4^{l_0+l_1}} \\ &\quad - \frac{(q_{H_2}+q_{H_3}+q_{H_7})(q_{re}+q_d)}{2^{l_0+l_1-1}} - \frac{3(q_{re}+q_d)}{q}\end{aligned}$$

If AskH_7^* happens, algorithm $\mathcal{B}$ will be able to solve mCBDH instance. Therefore we get,

$$\begin{aligned}\varepsilon' &\ge \frac{1}{q_{H_7}}\Pr[AskH_7^*] \\ &\ge \frac{1}{q_{H_7}}\frac{\in -\psi}{\varepsilon(1+q_{rk})} + \frac{\in -\psi}{\varepsilon(1+q_{ck})} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{(q_{H_2}q_{H_3} + q_{H_3}q_{H_7} + q_{H_7}q_{H_2})(q_{re}+q_d)}{4^{l_0+l_1}} \\ &\quad - \frac{(q_{H_2}+q_{H_3}+q_{H_7})(q_{re}+q_d)}{2^{l_0+l_1-1}} - \frac{3(q_{re}+q_d)}{q}\end{aligned}$$

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

$$\begin{aligned}t' \le t &+ (q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{H_6}+q_{H_7}+q_u+q_c+q_{rk}+q_{ck}+q_{re}+q_d)O(1) \\ &+ (2q_c+2q_u+6q_{rk}+q_{ck}+(q_{re}+1)(2q_d+(2q_{H_2}+2q_{H_3})q_d))\,t_{exp} + (q_{re}+q_d)\,t_p\end{aligned}$$

This completes the proof of Theorem 2.

5. CONCLUSION

In this paper, we proposed a more efficient CCA secure unidirectional C-PRE scheme with less number of bilinear pairings. The scheme is more elegant when compared to its counterparts. We have proved the security of the scheme in the random oracle model under appropriate security definitions. There are still many open problems to be solved, such as designing CCA secure C-PRE scheme in the standard model, C-PRE in other settings like identity based and certificateless cryptography.

REFERENCES

[1] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In ACM Conference on Computer and Communications Security 2007, pages 185–194, 2007.

[2] Jun Shao and Zhenfu Cao. CCA-Secure Proxy Re-encryption without Pairings. In Public Key Cryptography 2009, volume 5443 of LNCS, pages 357–376, 2009.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In Internet Society (ISOC): NDSS 2005, pages 29–43, 2005.

[4] H. Khurana and R. Koleva. Scalable security and accounting services for content-based publish subscribe systems. International Journal of E-Business Research, 2006.

[5] 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), pages 1–30, 2006.
[6] Jean-Sebastien Coron. On the Exact Security of Full Domain Hash. In CRYPTO, volume 1880 of LNCS, pages 229–235, 2000.
[7] T.S. Heydt-Benjamin, H. Chae, B. Defend, and K. Fu. Privacy for public transportation. In PET 2006, volume 4258 of LNCS, pages 1–19, 2005.
[8] Cheng-Kang Chu, Jian Weng, Sherman S. M. Chow, Jianying Zhou, and Robert H. Deng. Conditional Proxy Broadcast Re-Encryption. In ACISP 2009, volume 5594 of LNCS, pages 327–342, 2009.
[9] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In EUROCRYPT 1988, volume 1403 of LNCS, pages 127–144, 1998.
[10] Claus-Peter Schnorr. Efficient Identification and Signatures for Smart Cards. In CRYPTO 1989, volume 435 of LNCS, pages 239–252, 1989.
[11] A. Talmy and O. Dobzinski. Abuse freedom in access control schemes. In AINA 2006, pages 77–86, 2006.
[12] Jian Weng, Sherman S.M. Chow, Yanjiang Yang, and Robert H. Deng. Efficient Unidirectional Proxy Re-Encryption. Cryptology ePrint Archive, Report 2009/189 (2009) http://eprint.iacr.org/.
[13] A. Ivan and Y. Dodis. Proxy cryptography revisited. In Internet Society (ISOC): NDSS 2003, 2003.
[14] Jian Weng, Robert H. Deng, Xuhua Ding, Cheng-Kang Chu, and Junzuo Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In ASIACCS, pages 322–332, 2009.
[15] Masahiro Mambo and Eiji Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. IEICE Trans. Fund. Elect. Communications and CS, E80-A/1:54–63, 1997.
[16] S. Hohenberger, G.N. Rothblum, A. Shelat, and V. Vaikuntanathan. Securely obfuscating re-encryption. In TCC 2007, volume 4392 of LNCS, pages 233–252, 2007.
[17] Jian Weng, Yanjiang Yang, Qiang Tang, Robert H. Deng, and Feng Bao. Efficient Conditional Proxy Re-encryption with Chosen-Ciphertext Security. In ISC 2009, volume 5735 of LNCS, pages 151–166, 2009.
[18] Matthew Green and Giuseppe Ateniese. Identity-Based Proxy Re-encryption. In ACNS 2007, volume 4521 of LNCS, pages 288–306, 2007.
[19] Y-P. Chiu, C-L. Lei, and C-Y. Huang. Secure multicast using proxy encryption. In ICICS 2005, volume 3783 of LNCS, pages 280–290, 2005.
[20] H. Khurana and H-S. Hahm. Certified mailing lists. In ASIACCS 2006, pages 46–58, 2006.
[21] G. Taban, A.A. C'ardenas, and V.D. Gligor. Towards a secure and interoperable drm architecture. In ACM DRM 2006, pages 69–78, 2006.
[22] Smith. Tony. Dvd jon: buy drm-less tracks from apple itunes. 2005. http://www.theregister.co.uk/2005/03/18/itunes pymusique.
[23] H. Khurana, A. Slagell, and R. Bonilla. Sels: A secure e-mail list service. In ACM SAC 2005, pages 306–313, 2005.

Authors

S.Sree Vivek

He is currently a PhD scholar in the Department of Computer Science and Engineering, of IIT Madras. His research focuses on Provably Secure Public Key Cryptosystems.

S.Sharmila Deva Selvi

She is currently a PhD scholar in the Department of Computer Science and Engineering, of IIT Madras. Her research focuses on Provably Secure Public Key Cryptosystems.

V. Radhakishan

He is currently pursuing a Bachelor of Technology in the Department of Computer Science and Engineering, of National Institute of Technology, Tiruchirappalli. His research interests include 1) Public key cryptography 2) Randomized Algorithms and 3) Graph theory.

C. Pandu Rangan

He is currently a Professor in the Department of Computer Science and Engineering of IIT Madras. His research focuses on the design of pragmatic algorithms. His research interests include 1) Restricting the problem domain 2) Approximate algorithm design 3) Randomized algorithms 4)Parallel and VLSI algorithms and 5) Cryptography Applications. He is also a Fellow of Indian National Academy of Engineering (FNAE).