Efficient Certificate-Based Proxy Re-encryption Scheme for Data Sharing in Public Clouds

Yang Lu
College of Computer and Information Engineering, Hohai University
Nanjing, Jiangsu 211100 - China
[e-mail: luyangnsd@163.com]
*Corresponding author: Yang Lu

Received February 4, 2015; revised May 7, 2015; accepted June 8, 2015; published July 31, 2015

Abstract

Nowadays, public cloud storage is gaining popularity and a growing number of users are beginning to use the public cloud storage for online data storing and sharing. However, how the encrypted data stored in public clouds can be effectively shared becomes a new challenge. Proxy re-encryption is a public-key primitive that can delegate the decryption right from one user to another. In a proxy re-encryption system, a semi-trusted proxy authorized by a data owner is allowed to transform an encrypted data under the data owner's public key into a re-encrypted data under an authorized recipient's public key without seeing the underlying plaintext. Hence, the paradigm of proxy re-encryption provides a promising solution to effectively share encrypted data. In this paper, we propose a new certificate-based proxy re-encryption scheme for encrypted data sharing in public clouds. In the random oracle model, we formally prove that the proposed scheme achieves chosen-ciphertext security. The simulation results show that it is more efficient than the previous certificate-based proxy re-encryption schemes.

Keywords: Public cloud storage; encrypted data sharing; certificate-based proxy re-encryption; random oracle model; chosen-ciphertext security

This research was supported by the National Natural Science Foundation of China [grant number 61272542]. We would like to thank the anonymous referees for their helpful comments.

http://dx.doi.org/10.3837/tiis.2015.07.021 **ISSN : 1976-7277**

1. Introduction

Cloud computing has increasingly become a technology trend due to its key properties, such as cost saving and on-demand provisioning. There is an emerging trend that increasingly more users are beginning to use the public cloud storage for online data storing and sharing. However, many users still hesitate to move their data into a cloud, since they worry about their sensitive information being leaked by a cloud service provider (CSP). To preserve the confidentiality of the data stored at a cloud storage server, one can encrypt the data before sending it to the server. However, traditional encryption paradigm makes it difficult for flexibly sharing encrypted data between different users. For data sharing, there are two ways for a data owner to choose under the traditional encryption paradigm: he encrypts all data with a single symmetric key and gives his friends the symmetric key directly; or he downloads the encrypted data from the cloud storage, decrypts them, re-encrypts them using a new symmetric key and re-uploads the re-encrypted data along with the new symmetric key encrypted under his friend's public key to cloud. Obviously, the first way violates the least privilege principle since all data are leaked to his friends. For the second way, there are practical concerns on efficiency. Because the data owner has to re-encrypt the data and then re-upload to cloud, it brings heavy computation load and bandwidth cost to the data owner. In addition, the second way also loses the value of cloud storage.

How the encrypted data can be effectively shared in clouds has become a challenge. So far, a variety of methods (*e.g.* [1-10]) have been introduced in an attempt to deal with this problem. Among these approaches, proxy re-encryption (PRE) provides a promising solution for encrypted-data sharing in public clouds. The notion of PRE was introduced by Blaze *et al*. [11] in Eurocrypt'98. Its goal is to securely delegate the decryption right from one user (the *delegator*) to another (the *delegate*) without relying on trusted third parties. In a PRE scheme, a semi-trusted proxy authorized by the delegator is allowed to transform a ciphertext under the delegator's public key into a new ciphertext under the delegate's public key without seeing the underlying plaintext. More concretely, this cryptographic system works as follows: The delegator generates a proxy re-encryption key and sets it in a proxy. On receiving the ciphertexts under the delegator's public key, the proxy transforms them into the ciphertexts under the delegate's public key using the re-encryption key. Then, the delegate can decrypt the re-encrypted ciphertexts by using its private key directly. PRE can serve as a fundamental cryptographic building block for constructing secure data sharing applications in cloud systems. With a PRE system, a data owner is able to delegate the access rights of the sharing data to others so that they can access these data from the cloud storage directly. Furthermore, PRE introduces minor overhead on cloud users by eliminating any direct interaction between a data owner and its recipients.

Since its advent, PRE has attracted much attention in the research community and a number of schemes have been proposed. However, most of previous PRE schemes were constructed under either traditional public-key encryption (PKE) (*e.g.* [12-16]) or identity-based encryption (IBE) (*e.g.* [17-20]). It is well recognized that traditional PKE suffers from the cumbersome certificate management problem and IBE has inherent key escrow and distribution problems. To solve the key escrow problem in identity-based PRE, Xu *et al*. [3] proposed the notion of certificateless PRE by extending PRE into certificateless public-key cryptography (CL-PKC) that was presented by Al-Riyami and Paterson in Asiacrypt'03 [21]. However, CL-PKC needs a secure communication channel to distribute a

partial private key to each user. Therefore, certificateless PRE inevitably suffers from the key distribution problem similar to identity-based PRE. This feature limits the application of certificateless PRE in public clouds.

To address the problems imposed on the previous approaches, Sur *et al*. [4] introduced the notion of certificate-based PRE (CB-PRE) that follows the idea of certificate-based encryption (CBE) presented by Gentry [22] in Eurocrypt'03. CBE is a public-key encryption primitive that has attracted great interest in the recent years [23-30]. This primitive combines traditional PKE with IBE while preserving some of their most attractive features. As in traditional PKE, each user in CBE generates a pair of public key and private key independently and then requests a certificate from a CA. The difference is that a certificate is pushed only to its owner and acts as a partial decryption key. This additional functionality provides an efficient implicit certificate mechanism so that a user needs both his private key and certificate to perform decryption operations, while the other parties need not obtain the fresh information on this user's certificate status. The feature of implicit certificate makes CBE eliminate third-party queries for the certificate status and simplify the public key revocation problem in traditional PKE. Furthermore, there are no key escrow problem (since CA does not know users' private keys) and key distribution problem (since the certificates can be sent to their owners publicly) in CBE. To the best of our knowledge, two CB-PRE schemes have been proposed in the literature so far. In [4], Sur *et al*. provided a formal security model for CB-PRE schemes and proposed the first CB-PRE scheme that is provably secure in the random oracle model [31]. In [32], Li *et al*. proposed another CB-PRE scheme in the random oracle model. However, both of these two CB-PRE schemes are inefficient due to many costly pairing operations. For example, to re-encrypt a ciphertext, Sur *et al*.'s scheme [4] requires computing seven pairings while Li *et al*.'s scheme [32] requires computing five pairings.

The contribution of this paper is that we develop an efficient CB-PRE scheme with bilinear pairings. The proposed scheme requires computing at most two bilinear pairings in each operation. Compared with the previous CB-PRE schemes, it has obviously advantage in both the computation efficiency and the communication bandwidth. In the random oracle model, we strictly prove that the proposed scheme is chosen-ciphertext secure under the hardness of the bilinear Diffie-Hellman problem.

2. Preliminaries

2.1 Bilinear Map and Complexity Assumption

Let k be a security parameter and p be a k-bit prime number. Let G and G_T be two cyclic groups of prime order p and P be a generator of the group G.

A bilinear pairing is a map e: $G \times G \rightarrow G_T$ that takes two elements U and V in the group G as input and outputs an element $e(U, V)$ in the group G_T. It satisfies the following three properties:

(1) Bilinearity: For all $U, V \in G$ and all $a,b \in Z_p^*$, $e(U^a, V^b) = e(U, V)^{ab}$;

(2) Non-degeneracy: $e(P, P) \neq 1$;

(3) Computability: For all $U, V \in G$, $e(U, V)$ can be efficiently computed.

The security of our CB-PRE scheme is based on the following complexity assumption.

Definition 1 [33]**.** The bilinear Diffie-Hellman (BDH) problem in (G, G_T) is, given a tuple $(P, aP, bP, cP) \in G^4$ for unknown $a,b,c \in Z_p^*$, to compute $e(P, P)^{abc} \in G_T$. The advantage of a

probabilistic polynomial time (PPT) algorithm A_{BDH} in solving the BDH problem in (G, G_T) is defined as

$$Adv(A_{BDH}) = \Pr[A_{BDH}(p, G, G_T, P, aP, bP, cP) = e(P, P)^{abc}]. \quad (1)$$

The BDH assumption is that, for any PPT algorithm A_{BDH}, the advantage $Adv(A_{BDH})$ is negligible.

2.2 Certificate-Based Proxy Re-Encryption

In this paper, a CB-PRE scheme is composed of eight algorithms: (1) System setup algorithm ***Setup***, which is performed by a CA to generate a master secret key and a list of public system parameters; (2) User key generation algorithm ***UserKeyGen***, which is performed by the users to generate their private key and public key pairs; (3) Certificate generation algorithm ***Certify***, which is performed by a CA to generate a certificate for each user in the system; (4) Encryption algorithm ***Encrypt***, which is performed by the delegators to encrypt their data to generate the original ciphertexts; (5) Re-encryption key generation algorithm ***ReKeyGen***, which is performed by the delegators to generate the re-encryption keys; (6) Re-encryption algorithm ***ReEncrypt***, which is performed by a proxy to re-encrypt the original ciphertexts; (7) Normal decryption algorithm ***Decrypt1***, which is performed by the delegators to decrypt the original ciphertexts; (8) Re-encrypted ciphertext decryption algorithm ***Decrypt2***, which is performed by the delegates to decrypt the re-encrypted ciphertexts.

A more concrete functional description of a CB-PRE scheme is as follows:

(1) ***Setup***$(k) \rightarrow (msk, params)$
Input: a security parameter $k \in Z^+$
Output: a master secret key msk and a list of public system parameters $params$

(2) ***UserKeyGen***$(params) \rightarrow (SK_U, PK_U)$
Input: $params$
Output: a private key SK_U and a public key PK_U for a user U with identity id_U

(3) ***Certify***$(params, msk, id_U, PK_U) \rightarrow Cert_U$
Input: $params$, msk, a user U's identity id_U and public key PK_U
Output: a certificate $Cert_U$

(4) ***Encrypt***$(params, M, id_A, PK_A) \rightarrow C_A$
Input: $params$, a message M, a delegator A's identity id_A and public key PK_A
Output: an original ciphertext C_A

(5) ***ReKeyGen***$(params, id_A, SK_A, Cert_A, id_B, PK_B) \rightarrow RK_{A \rightarrow B}$
Input: $params$, a delegator A's identity id_A, private key SK_A and certificate $Cert_A$ and a delegate B's identity id_B and public key PK_B
Output: a re-encryption key $RK_{A \rightarrow B}$

(6) ***ReEncrypt***$(params, C_A, RK_{A \rightarrow B}) \rightarrow C_B$
Input: $params$, a ciphertext C_A and a re-encryption key $RK_{A \rightarrow B}$
Output: a re-encrypt ciphertext C_B under a delegate B's identity id_B and public key PK_B

(7) ***Dencrypt1***$(params, C_A, id_A, SK_A, Cert_A) \rightarrow M$
Input: $params$, an original ciphertext C_A, a delegator A's identity id_A, private key SK_A and certificate $Cert_A$
Output: a message M or an error symbol $\perp$ if the decryption fails

(8) ***Dencrypt2***$(params, C_B, id_B, SK_B, Cert_B, id_A, PK_A) \rightarrow M$
Input: $params$, a re-encrypted ciphertext C_B, a delegate B's identity id_B, private key SK_B and certificate $Cert_B$ and a delegator A's identity id_A and public key PK_A
Output: a message M or an error symbol $\perp$ if the decryption fails

In the above algorithms ***UserKeyGen*** and ***Certify***, a user U may be a delegator A or a delegate B.

For correctness, it is required that, for any message M and any identity id_A and id_B, the following two equations should hold: ***Decrypt1***($params$, ***Encrypt***($params$, M, id_A, PK_A), SK_A, $Cert_A$) = M, ***Decrypt2***($params$, ***ReEncrypt***($params$, ***Encrypt***($params$, M, id_A, PK_A), $RK_{A\rightarrow B}$), id_B, SK_B, $Cert_B$, id_A, PK_A) = M.

As introduced by Sur *et al*. in [4], the security model of CB-PRE schemes is an extension of the model of CBE schemes in which there are two kinds of adversaries, namely Type-I adversary (denoted by A_I) and Type-II adversary (denoted by A_{II}). The Type-I adversary models an uncertified entity while the Type-II adversary models a malicious CA who knows the master secret key. To formalize the security notions for CB-PRE schemes, we first describe the following six oracles. A Type-I or Type-II adversary can adaptively make requests to some of these oracles. We assume that the challenger keeps a history of "query-answer" when interacting with the adversary.

(1) OUserCreate(idU): On input an identity idU, the challenger responds with the public key PKU associated with the identity idU. If the identity idU has not been created, then the challenger generates a public key PKU and a private key SKU respectively and returns PKU. In this case, the identity idU is said to be created. Note that other oracles defined below only respond to an identity which has been created.

(2) OCorrupt(idU): On input an identity idU, the challenger outputs the private key SKU associated with the identity idU.

(3) OCertificate(idU): On input an identity idU, the challenger responds with a certificate CertU. Note that such an oracle is only queried by the Type-I adversary since the Type-II adversary can generate any user's certificate by itself.

(4) $O^{ReKeyGen}(id_A, id_B)$: On input two identities id_A and id_B, the challenger responds with a re-encryption key $RK_{A\rightarrow B}$.

(5) $O^{ReEncrypt}(id_A, id_B, C_A)$: On input two identities id_A, id_B and a ciphertext C_A under the identity id_A and the public key PK_A, the challenger responds with a transformed ciphertext C_B under the identity id_B and the public key PK_B.

(6) $O^{Decrypt}(id_U, C_U)$: On input an identity id_U and a ciphertext C_U, the challenger responds with the decryption of the ciphertext C_U.

The chosen-ciphertext security of CB-PRE schemes can be formally defined by the following adversarial game "**IND-CBPRE-CCA2 Game**", in which a Type-I or Type-II adversary $A_X \in \{A_I, A_{II}\}$ interacts with a challenger.

Setup. On input a security parameter k, the challenger runs the algorithm ***Setup***(k) to generate a master secret key msk and a list of public parameters $params$. It then sends $params$ to the adversary A_X. If A_X is a Type-II adversary, the challenger also sends the master secret key msk to it.

Phase 1. In this phase, the adversary A_X can adaptively query the oracles $O^{CreateUser}$, $O^{Corrupt}$, $O^{Certificate}$, $O^{ReKeyGen}$, $O^{ReEncrypt}$ and $O^{Decrypt}$ if it is a Type-I adversary or the oracles $O^{CreateUser}$, $O^{PrivateKey}$, $O^{ReKeyGen}$, $O^{ReEncrypt}$ and $O^{Decrypt}$ if it is a Type-II adversary. The challenger responds as described above.

Challenge. Once the adversary A_X decides that Phase 1 is over, it outputs an identity id_{ch} and two equal-length messages (M_0, M_1) on which it wants to be challenged. The challenger

picks a random bit $b \in \{0,1\}$, computes $C_{ch} = Encrypt(params, M_b, id_{ch}, PK_{ch})$, and then outputs C_{ch} as the challenge ciphertext to the adversary A_X.

Phase 2. In this phase, the adversary A_X issues a second sequence of queries as in Phase 1.

Guess. After all queries, the adversary A_X outputs a guess $b' \in \{0,1\}$ for the bit b. We say that the adversary A_X wins the game if $b = b'$ and the following restrictions are simultaneously satisfied: (1) (id_{ch}, C_{ch}) and its derivatives cannot be submitted to the oracle $O^{Decrypt}$; (2) id_{ch} cannot be submitted to the oracle $O^{Certificate}$ if A_X is a Type-I adversary or the oracle $O^{PrivateKey}$ if A_X is a Type-II adversary. The advantage of the adversary A_X is defined to be

$$Adv(A_X) = 2|\Pr[\, b = b' \,] - 1/2|. \tag{2}$$

For our definition to make sense, we consider the notion of derivative of the challenge ciphertext [13].

Definition 2. Assume that id_{ch} is the challenge identity and C_{ch} is the challenge ciphertext in the above games, (id_U, C_U) is said to be a derivative of (id_{ch}, C_{ch}) if either (1) the adversary A_X has queried the oracle $O^{ReEncrypt}$ on (id_{ch}, id_U, C_{ch}) to get a new ciphertext C_U, or (2) the adversary A_X has queried the oracle $O^{ReKeyGen}(id_{ch}, id_U)$ to get the re-encryption key $RK_{ch \to U}$ and C_U is the result of ***ReEncrypt***($params$, C_{ch}, $RK_{ch \to U}$).

It is clear that in the above game we should disallow the queries to $O^{Decrypt}$ not only on the challenge ciphertext (id_{ch}, C_{ch}) as usual, but also on any derivative of (id_{ch}, C_{ch}). Otherwise, the adversary A_X can easily win the game by making a query to $O^{ReEncrypt}$ or $O^{ReKeyGen}$ corresponding to (id_{ch}, C_{ch}).

Definition 3. A CB-PRE scheme is said to be secure against adaptive chosen-ciphertext attacks (or IND-CBPRE-CCA2 secure) if no PPT adversary has non-negligible advantage in the above game.

3. Description of the Proposed CB-PRE Scheme

Motivated by Green and Ateniese's identity-based PRE scheme [17], we propose a new CB-PRE scheme. The proposed scheme consists of the following eight algorithms:

(1) ***Setup***(k). The CA chooses a k-bit prime number p, generates two cyclic groups G and G_T of order p such that there exists a bilinear paring map e: $G \times G \to G_T$. It randomly chooses a generator $P \in G$ and a master secret key $s \in Z_p^*$, and sets $P_{pub} = P^s$. Additionally, it selects five cryptographic hash functions H_1: $\{0,1\}^* \times G \to G$, H_2: $\{0,1\}^n \times G_T \times \{0,1\}^* \times G \to Z_p^*$, H_3: $\{0,1\}^* \times G \times G \to G$, H_4: $G_T \to \{0,1\}^n$ and H_5: $\{0,1\}^* \times \{0,1\}^* \times G_T \times G \to G$, where n is the bit-length of the message to be encrypted. The public system parameters are $params = \{p, G, G_T, e, n, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$ and the master secret key is $msk = s$.

(2) ***UserKeyGen***(params). A user U with identity id_U chooses a random value $x_U \in Z_p^*$ as his private key SK_U and computes his public key $PK_U = P^{x_U}$.

(3) ***Certify***($params$, msk, id_U, PK_U). The CA computes $Cert_U = Q_U^s$ as a certificate for a user U with identity id_U and public key PK_U, where $Q_U = H_1(id_U, PK_U)$. The user U can check the validness of $Cert_U$ by verifying whether $e(P, Cert_U) = e(P_{pub}, Q_U)$.

(4) ***Encrypt***(*params*, M, id_A, PK_A). To encrypt a message $M \in \{0,1\}^n$, the delegator A randomly chooses $\sigma \in G_T$, sets $r = H_2(M, \sigma, id_A, PK_A)$, and then computes an original ciphertext $C_A = (U_A, V_A, W_A) = (P^r, \sigma \cdot e(P_{pub}, Q_A)^{-r} \cdot e(PK_A, R_A)^{-r}, M \oplus H_4(\sigma))$, where $Q_A = H_1(id_A, PK_A)$ and $R_A = H_3(id_A, PK_A, P_{pub})$.

The algorithm *Encrypt* does not require any paring computations once $e(P_{pub}, Q_A)$ and $e(PK_A, R_A)$ have been pre-computed.

(5) ***ReKeyGen***(params, id_A, SK_A, $Cert_A$, id_B, PK_B). To generate a proxy re-encryption key $RK_{A\to B}$, the delegator A computes $K_1 = e(Cert_A, Q_B)$, $K_2 = (PK_B)^{SK_A}$ and $K_3 = (R_A)^{SK_A} \cdot Cert_A$, and then sets $RK_{A\to B} = H_5(id_A, id_B, K_1, K_2) \cdot K_3$, where $Q_B = H_1(id_B, PK_B)$ and $R_A = H_3(id_A, PK_A, P_{pub})$.

(6) ***ReEncrypt***(*params*, C, $RK_{A\to B}$). To convert an original ciphertext $C_A = (U_A, V_A, W_A)$ under identity id_A and public key PK_A into a re-encrypted ciphertext C_B under identity id_B and public key PK_B using the proxy re-encryption key $RK_{A\to B}$, the proxy sets $U_B = U_A$ and $W_B = W_A$ respectively, computes $V_B = V_A \cdot e(U_A, RK_{A\to B})$ and then sets $C_B = (U_B, V_B, W_B)$.

(7) ***Decrypt1***(*params*, C_A, id_A, SK_A, $Cert_A$). To decrypt an original ciphertext $C_A = (U_A, V_A, W_A)$, the delegator A first computes $\sigma' = V_A \cdot e(U_A, (R_A)^{SK_A} \cdot Cert_A)$ and $M' = W_A \oplus H_4(\sigma')$, where $R_A = H_3(id_A, PK_A, P_{pub})$. It then checks whether $U_A = P^{r'}$ where $r' = H_2(M', \sigma', id_A, PK_A)$. If this check holds, it outputs M', otherwise outputs $\perp$.

(8) ***Decrypt2***(params, C_B, id_B, SK_B, $Cert_B$, id_A, PK_A). To decrypt a re-encrypted ciphertext $C_B = (U_B, V_B, W_B)$ from a delegator A with identity id_A and public key PK_A, the delegate B first computes $\sigma' = V_B \cdot e(U_B, H_5(id_A, id_B, e(Q_A, Cert_B), (PK_A)^{SK_B})^{-1})$ and $M' = W_B \oplus H_4(\sigma')$, where $Q_A = H_1(id_A, PK_A)$. It then checks whether $U_B = P^{r'}$ where $r' = H_2(M', \sigma', id_A, PK_A)$. If this check holds, it outputs M', otherwise outputs $\perp$.

4. Analysis of the Proposed CB-PRE Scheme

4.1 Correctness

The correctness of the proposed CB-PRE scheme can be verified as follows:

If C_A is an original ciphertext, *i.e.*, $C_A = (U_A, V_A, W_A)$, then we have

$$\begin{aligned}\sigma' &= V_A \cdot e(U_A, (R_A)^{SK_A} \cdot Cert_A)\\ &= \sigma \cdot e(P_{pub}, Q_A)^{-r} \cdot e(PK_A, R_A)^{-r} \cdot e(P^r, (R_A)^{SK_A}) \cdot e(P^r, Cert_A)\\ &= \sigma \cdot e(P^{\alpha}, Q_A)^{-r} \cdot e(P^{SK_A}, R_A)^{-r} \cdot e(P^r, (R_A)^{SK_A}) \cdot e(P^r, Q_A^{\alpha})\\ &= \sigma.\end{aligned}$$

If C_B is a re-encrypted ciphertext from a delegator A with identity id_A and public key PK_A, *i.e.*, $C_B = (U_B, V_B, W_B) = (U_A, V_A \cdot e(U_A, RK_{A\to B}), W_A)$, then we have

$$\begin{aligned}\sigma' &= V_B \cdot e(U_A, H_5(id_A, id_B, e(Q_A, Cert_B), (PK_A)^{SK_B})^{-1})\\ &= V_A \cdot e(U_A, H_5(id_A, id_B, e(Cert_A, Q_B), (PK_B)^{SK_A})) \cdot (R_A)^{SK_A} \cdot Cert_A)\\ &\qquad \cdot e(U_A, H_5(id_A, id_B, e(Q_A, Cert_B), (PK_A)^{SK_B})^{-1})\\ &= V_A \cdot e(U_A, (R_A)^{SK_A}) \cdot e(U_A, Cert_A)\\ &= \sigma \cdot e(P_{pub}, Q_A)^{-r} \cdot e(PK_A, R_A)^{-r} \cdot e(P^r, (R_A)^{SK_A}) \cdot e(P^r, Q_A^{\alpha})\\ &= \sigma \cdot e(P^{\alpha}, Q_A)^{-r} \cdot e(P^{SK_A}, R_A)^{-r} \cdot e(P^r, (R_A)^{SK_A}) \cdot e(P^r, Q_A^{\alpha})\end{aligned}$$

$= \sigma$.

Hence, the normal decryption and the re-encrypted ciphertext decryption are both correct.

4.2 Security Proof

Theorem 1. In the random oracle model, our CB-PRE scheme is IND-CBPRE-CCA2 secure under the BDH assumption.

This theorem can be proved by combining the following **Lemma 1** and **Lemma 2**.

Lemma 1. Assume that a Type-I adversary A_I has an advantage ε against our CB-PRE scheme when asking at most q_{uc} queries to the oracle $O^{UserCreate}$, q_{cr} queries to the oracle $O^{Corrupt}$, q_{cer} queries to the oracle $O^{Certificate}$, q_{rk} queries to the oracle $O^{ReKeyGen}$, q_{ren} queries to the oracle $O^{ReEncrypt}$, q_{dec} queries to the oracle $O^{Decrypt}$ and q_i queries to the random oracles H_i (i = 1,2,3,4,5) respectively, then there exists an algorithm A_{BDH} to solve the BDH problem with advantage

$$\varepsilon' \geq \frac{\varepsilon}{q_1 q_4}\left(1-\frac{q_{ren}}{2^k}\right)\left(1-\frac{q_{dec}}{2^k}\right). \tag{3}$$

Proof. We show how to construct an algorithm A_{BDH} to solve the BDH problem. Assume that the algorithm A_{BDH} is given a random instance (P, P^a, P^b, P^c) of the BDH problem in (G, G_T) and asked to compute $e(P, P)^{abc}$. In order to solve the given problem, A_{BDH} needs to simulate a challenger and all oracles for the adversary A_I.

In the setup phase, the algorithm A_{BDH} sets $P_{pub} = P^a$ and randomly chooses an index $\theta \in \{1,2,\ldots,q_1\}$. Then, it starts **IND-CBPRE-CCA2 Game** by supplying the adversary A_I with $params = \{p, G, G_T, e, n, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$, where $H_1 \sim H_5$ are random oracles controlled by A_{BDH}. Note that the master key msk is the value a that is unknown to A_{BDH}.

Now, the algorithm A_{BDH} starts to respond various queries as follows:

H_1 queries: We assume that the adversary A_I's queries to the random oracle H_1 are distinct. The algorithm A_{BDH} maintains a list H_1List of tuples (id_i, PK_i, Q_i, $Cert_i$). On receiving such a query on (id_i, PK_i), it does the following: (1) If id_i already appears on the list H_1List in a tuple (id_i, PK_i, Q_i, $Cert_i$), then it outputs Q_i to the adversary A_I. (2) Else if the query is on the θ-th distinct identity id_θ, it sets $h_{1i} = P^b$, inserts a new tuple (id_i, PK_i, Q_i, $\perp$) into the list H_1List and then returns Q_i. (3) Otherwise, it randomly chooses $s_i \in Z_p^*$, sets $Q_i = P^{s_i}$ and $Cert_i = (P^a)^{s_i}$, inserts a new tuple (id_i, PK_i, Q_i, $Cert_i$) into the list H_1List and then returns Q_i.

H_2 queries: The algorithm A_{BDH} maintains a list H_2List of tuples (M, σ, id_i, PK_i, r). On receiving such a query on (M, σ, id_i, PK_i), it checks whether (M, σ, id_i, PK_i) already appears on the list H_2List in a tuple (M, σ, id_i, PK_i, r). If so, it outputs r. Otherwise, it outputs a random value $r \in Z_p^*$ to A_I and inserts a new tuple (M, σ, id_i, PK_i, r) into the list H_2List.

H_3 queries: The algorithm A_{BDH} maintains a list H_3List of tuples (id_i, PK_i, t_i, R_i). On receiving such a query on (id_i, PK_i, P_{pub}), it checks whether (id_i, PK_i) already appears on the list H_3List in a tuple (id_i, PK_i, t_i, R_i). If so, it returns R_i to A_I directly. Otherwise, it randomly chooses $t_i \in Z_p^*$, sets $R_i = P^{t_i}$, inserts a new tuple (id_i, PK_i, t_i, R_i) into H_3List and returns R_i to A_I.

H_4 queries: The algorithm A_{BDH} maintains a list H_4List of tuples (σ, h_4). On receiving such a query on σ, it checks whether σ already appears on the list H_4List in a tuple (σ, h_4). If so, it

returns h_4 to A_I directly. Otherwise, it returns a random value $h_4 \in \{0, 1\}^n$ and inserts a new tuple (σ, h_4) into the list H_4List.

H_5 queries: The algorithm A_{BDH} maintains a list H_5List of tuples (id_i, id_j, K_1, K_2, h_{5ij}). On receiving such a query on (id_i, id_j, K_1, K_2), it checks whether (id_i, id_j, K_1, K_2) already appears on the list H_5List in a tuple (id_i, id_j, K_1, K_2, h_{5ij}). If so, it returns h_{5ij} to A_I directly. Otherwise, it returns a random value $h_{5ij} \in G$ and inserts a new tuple (id_i, id_j, K_1, K_2, h_{5ij}) into the list H_5List.

$O^{UserCreate}$ queries: The algorithm A_{BDH} maintains a list *KeyList* of tuples (id_i, PK_i, SK_i). On receiving such a query on id_i, it checks whether the identity id_i already appears on the list *KeyList* in a tuple (id_i, PK_i, SK_i). If so, it returns PK_i to A_I directly. Otherwise, it randomly chooses $x_i \in Z_p^*$ as SK_i, computes $PK_i = P^{x_i}$, inserts a new tuple (id_i, PK_i, SK_i) into the list *KeyList* and then returns PK_i.

$O^{Corrupt}$ queries: On receiving such a query on id_i, A_{BDH} searches id_i in the list *KeyList* to find a tuple (id_i, PK_i, SK_i) and returns SK_i to A_I.

$O^{Certificate}$ queries: On receiving such a query on id_i, A_{BDH} aborts if $id_i = id_\theta$. Otherwise, it searches id_i in the list H_1List to find a tuple (id_i, PK_i, Q_i, $Cert_i$) and returns $Cert_i$ to A_I.

$O^{ReKeyGen}$ queries: On receiving such a query on (id_i, id_j), A_{BDH} aborts if $id_i = id_\theta$. Otherwise, it respectively retrieves the private key SK_i and certificate $Cert_i$ associated with the identity id_i and the public key PK_j associated with the identity id_j, then computes ***ReKeyGen***(*params*, id_i, SK_i, $Cert_i$, id_j, PK_j) and outputs the result to A_I.

$O^{ReEncrypt}$ queries: On receiving such a query on (id_i, id_j, $C_i = (U_i, V_i, W_i)$), A_{BDH} does the following: (1) If $id_i = id_\theta$, it searches in the list H_2List for a tuple (M, σ, id_i, PK_i, r) such that $U_i = P^r$, $V_i = \sigma \cdot e(P_{pub}, H_1(id_i, PK_i))^r \cdot e(PK_i, H_3(id_i, PK_i, P_{pub}))^r$ and $W_i = M \oplus H_4(\sigma)$. The query is rejected if no such tuple is found. Otherwise, it sets $V_j = \sigma \cdot e(U_i, H_5(id_i, id_j, e(H_1(id_i, PK_i), Cert_j), (PK_i)^{SK_j}))$ and returns $C_j = (id_i, U_i, V_j, W_i)$ as the re-encryption ciphertext to A_I. (2) Otherwise, it makes a query $O^{ReKeyGen}(id_i, id_j)$ to obtain a re-encryption key $RK_{i\to j}$ and then returns the result of ***ReEncrypt***(*params*, C_i, $RK_{i\to j}$) to A_I. Note that a valid ciphertext submitted to this oracle is rejected with probability smaller than $q_{ren}/2^k$ across the whole game.

$O^{Decrypt}$ queries: On receiving such a query on (id_i, C_i), A_{BDH} does the following: (1) If $id_i = id_\theta$ and $C_i = (U_i, V_i, W_i)$ is an original ciphertext, it searches in the list H_2List for a tuple (M, σ, id_i, PK_i, r) such that $U_i = rP$, $V_i = \sigma \cdot e(P_{pub}, H_1(id_i, PK_i))^r \cdot e(PK_i, H_3(id_i, PK_i, P_{pub}))^r$ and $W_i = M \oplus H_4(\sigma)$. If no such tuple is found, it rejects this query. Otherwise, it returns M in this tuple to A_I. (2) Else if $id_i = id_\theta$ and $C_i = (id_j, U_j, V_i, W_j)$ is a transformed ciphertext, it queries the oracle $O^{ReKeyGen}$ on (id_j, id_i) to obtain a re-encryption key $RK_{j\to i}$ and computes $V_j = V_i / RK_{j\to i}$. It then searches in the list H_2List for a tuple (M, σ, id_j, PK_j, r) such that $U_j = rP$, $V_j = \sigma \cdot e(P_{pub}, H_1(id_j, PK_j))^r \cdot e(PK_j, H_3(id_j, PK_j, P_{pub}))^r$ and $W_j = M \oplus H_4(\sigma)$. If no such tuple is found, B rejects this query. Otherwise, it returns M in this tuple to A_I. (3) Otherwise, it obtains SK_i and $Cert_i$ associated with the identity id_i and then returns the result of ***Decrypt***(*params*, C_i, SK_i, $Cert_i$). Note that a valid ciphertext submitted to this oracle is rejected with probability smaller than $q_{dec}/2^k$.

In the challenge phase, A_I outputs (M_0, M_1, id_{ch}) on which it wants to be challenged. If $id_{ch} \neq id_\theta$, then A_{BDH} aborts. Otherwise, it sets $U_{ch} = cP$, randomly chooses $V_{ch} \in G_T$, $W_{ch} \in \{0, 1\}^n$, and returns $C_{ch} = (U_{ch}, V_{ch}, W_{ch})$ as the challenge ciphertext to A_I.

In the guess phase, A_I outputs a bit b' which is ignored by A_{BDH}. Observe that the decryption of C_{ch} is $W_{ch} \oplus H_4(V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch}))$ where $R_{ch} = H_3(id_{ch}, PK_{ch}, P_{pub})$.

To produce a result, A_{BDH} randomly picks a tuple (σ, h_4) from the list H_4List, retrieves the value t_{ch} from the tuple $(id_{ch}, PK_{ch}, t_{ch}, R_{ch})$ in the list H_3List and returns

$$T = V_{ch} / (\sigma \cdot e(PK_{ch}, t_{ch}U_{ch})) \tag{4}$$

as the solution to the given BDH problem.

Note that if $\sigma = V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch})$, then $V_{ch} = \sigma \cdot e(P_{pub}, Q_{ch})^c \cdot e(PK_{ch}, R_{ch})^c$, where $Q_{ch} = H_1(id_{ch}, PK_{ch})$. Thus, we have

$$T = V_{ch} / (\sigma \cdot e(PK_{ch}, t_{ch}U_{ch})) = e(P_{pub}, Q_{ch})^c = e(aP, bP)^c = e(P, P)^{abc}. \tag{5}$$

We now estimate A_{BDH}'s advantage in solving the BDH problem.

Let **Fail** denote the event that the above simulation fails and $\mathbf{QueryH_4}$ the event that A_I makes a query $H_4(V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch})$. From the above simulation, the event **Fail** occurs if any one of the following five events occurs: (1) $\mathbf{E_1}$: In the challenge phase, A_I does not choose id_θ as the challenge identity; (2) $\mathbf{E_2}$: A_I queries the oracle $O^{Certificate}$ on id_θ; (3) $\mathbf{E_3}$: A_I queries the oracle $O^{ReKeyGen}$ on (id_θ, id_j); (4) $\mathbf{E_4}$: A_{BDH} rejects a valid ciphertext submitted to the oracle $O^{ReEncrypt}$; (5) $\mathbf{E_5}$: A_{BDH} rejects a valid ciphertext submitted to the oracle $O^{Decrypt}$.

We clearly have that $\Pr[\neg\mathbf{E_1}] = 1/q_1$ and $\neg\mathbf{E_1}$ implies $\neg\mathbf{E_2}$ and $\neg\mathbf{E_3}$. We also already observed that $\Pr[\mathbf{E_4}] \le q_{ren}/2^k$ and $\Pr[\mathbf{E_5}] \le q_{dec}/2^k$. Thus, the probability that the above simulation does not fail is

$$\Pr[\neg\mathbf{Fail}] = \Pr[\neg\mathbf{E_1} \wedge \neg\mathbf{E_2} \wedge \neg\mathbf{E_3} \wedge \neg\mathbf{E_4} \wedge \neg\mathbf{E_5}] \ge \frac{1}{q_1}(1-\frac{q_{ren}}{2^k})(1-\frac{q_{dec}}{2^k}). \tag{6}$$

Let **Event** be the event $\mathbf{QueryH_4} \mid \neg\mathbf{Fail}$. It is clear that if **Event** does not happen, then A_I does not gain any advantage greater than 1/2 in guessing b. Namely, we have the probability $\Pr[b = b' \mid \neg\mathbf{Event}] = 1/2$. Hence, by splitting the probability $\Pr[b = b']$, we obtain $\Pr[b = b'] = \Pr[b = b' \mid \neg\mathbf{Event}] \cdot \Pr[\neg\mathbf{Event}] + \Pr[b = b' \mid \mathbf{Event}] \cdot \Pr[\mathbf{Event}] \le \Pr[\neg\mathbf{Event}]/2 + \Pr[\mathbf{Event}] = 1/2 + \Pr[\mathbf{Event}]/2$. By the definition of the advantage in **IND-CBPRE-CCA2 Game**, we have $\varepsilon \le 2|\Pr[b = b'] - 1/2| \le \Pr[\mathbf{Event}] \le \Pr[\mathbf{QueryH_4}]/\Pr[\neg\mathbf{Fail}]$. Hence, we get

$$\Pr[\mathbf{QueryH_4}] \ge \varepsilon\Pr[\neg\mathbf{Fail}] \ge \frac{\varepsilon}{q_1}(1-\frac{q_{ren}}{2^k})(1-\frac{q_{dec}}{2^k}). \tag{7}$$

Finally, we get the announced bound on A_{BDH}'s advantage in solving the BDH problem by noting that A_{BDH} selects the correct tuple from the list H_4List with probability $1/q_4$.

Lemma 2. Assume that a Type-II adversary A_{II} has an advantage ε against our CB-PRE scheme when asking at most q_{uc} queries to the oracle $O^{UserCreate}$, q_{cr} queries to the oracle $O^{Corrupt}$, q_{rk} queries to the oracle $O^{ReKeyGen}$, q_{ren} queries to the oracle $O^{ReEncrypt}$, q_{dec} queries to the oracle $O^{Decrypt}$ and q_i queries to the random oracles H_i (i = 1,2,3,4,5) respectively, then there exists an algorithm A_{BDH} to solve the BDH problem with advantage

$$\varepsilon' \geq \frac{\varepsilon}{q_{uc} q_4}(1-\frac{q_{ren}}{2^k})(1-\frac{q_{dec}}{2^k}) . \quad (8)$$

Proof. We show how to construct an algorithm A_{BDH} to solve the BDH problem. Assume that A_{BDH} is given a random instance (P, aP, bP, cP) of the BDH problem in (G, G_T) and asked to compute $e(P, P)^{abc}$. In order to solve the given problem, A_{BDH} needs to simulate a challenger and all oracles for the adversary A_{II}.

In the setup phase, A_{BDH} randomly chooses $\alpha \in Z_p^*$ and sets $P_{pub} = \alpha P$. Furthermore, it randomly chooses an index $\theta \in \{1,2,\ldots,q_{uc}\}$. Then, it starts **IND-CBPRE-CCA2 Game** by supplying A_{II} with $msk = \alpha$ and $params = \{p, G, G_T, e, n, P, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$, where $H_1 \sim H_5$ are random oracles controlled by A_{BDH}.

During the query-answer phase, A_{BDH} responds A_{II}'s queries to the oracles H_2, H_4, H_5, $O^{ReKeyGen}$, $O^{ReEncrypt}$ and $O^{Decrypt}$ as in the proof of Lemma 1 and other queries as follows:

H_1 queries: A_{BDH} maintains a list H_1List of tuples (id_i, PK_i, Q_i). On receiving such a query on (id_i, PK_i), A_{BDH} checks whether (id_i, PK_i) already appears on the list H_1List in a tuple (id_i, PK_i, Q_i). If so, then it returns Q_i to A_{II}. Otherwise, it returns a random value $Q_i \in Z_p^*$ to A_{II} and inserts a new tuple (id_i, PK_i, Q_i) into the list H_1List.

H_3 queries: A_{BDH} maintains a list H_3List of tuples (id_i, PK_i, R_i). On receiving such a query on (id_i, PK_i, P_{pub}), A_{BDH} does the following: (1) If (id_i, PK_i) already appears on H_3List in a tuple (id_i, PK_i, R_i), it returns R_i to A_{II} directly. (2) Else if $id_i = id_\theta$, it returns $R_i = bP$ to A_{II} and inserts a new tuple (id_i, PK_i, R_i) into the list H_3List. (3) Otherwise, it returns a random element $R_i \in G$ to A_{II} and inserts a new tuple (id_i, PK_i, R_i) into the list H_3List.

$O^{UserCreate}$ queries: A_{BDH} maintains a list $KeyList$ of tuples (id_i, PK_i, SK_i). On receiving such a query on id_i, A_{BDH} does the following: (1) If id_i already appears on $KeyList$ in a tuple (id_i, PK_i, SK_i), it returns PK_i to A_{II} directly. (2) Else if the query is on the θ-th distinct identity id_θ, it returns $PK_\theta = bP$ to A_{II} and inserts (id_θ, PK_θ, $\perp$) into the list $KeyList$. Note that the private key corresponding to PK_θ is $SK_\theta = b$ which is unknown to A_{BDH}. (3) Otherwise, it randomly chooses $x_i \in Z_p^*$ as SK_i, computes $PK_i = x_iP$, inserts a new tuple (id_i, PK_i, SK_i) into the list $KeyList$ and then returns PK_i to A_{II}.

In the challenge phase, A_{II} outputs (M_0, M_1, id_{ch}) on which it wants to be challenged. If $id_{ch} \neq id_\theta$, then A_{BDH} aborts. Otherwise, it sets $U_{ch} = cP$, randomly chooses $V_{ch} \in G_T$, $W_{ch} \in \{0, 1\}^n$, and returns $C_{ch} = (U_{ch}, V_{ch}, W_{ch})$ as the challenge ciphertext to A_{II}.

In the guess phase, A_{II} outputs a bit b' which is ignored by A_{BDH}. Observe that the decryption of C_{ch} is $W_{ch} \oplus H_4(V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch}))$ where $R_{ch} = H_3(id_{ch}, PK_{ch}, P_{pub})$. To produce a result, A_{BDH} randomly picks a tuple (σ, h_4) from the list H_4List and returns

$$T = V_{ch} / (\sigma \cdot e(\alpha U_{ch}, Q_{ch})) \quad (9)$$

as the solution to the given BDH problem, where $Q_{ch} = H_1(id_{ch}, PK_{ch})$.

Note that if $\sigma = V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch})$, then $V_{ch} = \sigma \cdot e(P_{pub}, Q_{ch})^c \cdot e(PK_{ch}, R_{ch})^c$. Thus, we have

$$T = V_{ch} / (\sigma \cdot e(\alpha U_{ch}, Q_{ch})) = e(PK_{ch}, R_{ch})^c = e(aP, bP)^c = e(P, P)^{abc}. \quad (10)$$

We now estimate A_{BDH}'s advantage in solving the BDH problem.

Let **Fail** denote the event that the above simulation fails and **QueryH$_4$** the event that A_{II} makes a query $H_4(V_{ch} / e(U_{ch}, Cert_{ch} + SK_{ch}R_{ch})$. From the above simulation, the event **Fail** occurs if any one of the following events occurs: (1) **E$_1$**: In the challenge phase, A_{II} does not choose id_θ as the challenge identity; (2) **E$_2$**: A_{II} queries the oracle $O^{Corrupt}$ on id_θ; (3) **E$_3$**: A_{II} queries the oracle $O^{ReKeyGen}$ on (id_θ, id_j); (4) **E$_4$**: A_{BDH} rejects a valid ciphertext submitted to the oracle $O^{ReEncrypt}$; (5) **E$_5$**: A_{BDH} rejects a valid ciphertext submitted to the oracle $O^{Decrypt}$.

We clearly have that $\Pr[\neg\mathbf{E_1}] = 1/q_{uc}$ and $\neg\mathbf{E_1}$ implies $\neg\mathbf{E_2}$ and $\neg\mathbf{E_3}$. As in the proof of Lemma 1, we have $\Pr[\mathbf{E_4}] \le q_{ren}/2^k$ and $\Pr[\mathbf{E_5}] \le q_{dec}/2^k$. Thus, the probability that the above simulation does not fail is

$$\Pr[\neg\mathbf{Fail}] = \Pr[\neg\mathbf{E_1} \wedge \neg\mathbf{E_2} \wedge \neg\mathbf{E_3} \wedge \neg\mathbf{E_4} \wedge \neg\mathbf{E_5}] \ge \frac{1}{q_{uc}}(1-\frac{q_{ren}}{2^k})(1-\frac{q_{dec}}{2^k}). \quad (11)$$

Let **Event** be the event **QueryH$_4$** | $\neg$**Fail**. It is clear that if **Event** does not happen, then A_{II} does not gain any advantage greater than 1/2 in guessing b. Namely, we have the probability $\Pr[b = b' | \neg\mathbf{Event}] = 1/2$. Hence, by splitting the probability $\Pr[b = b']$, we obtain $\Pr[b = b'] = \Pr[b = b' | \neg\mathbf{Event}]\cdot\Pr[\neg\mathbf{Event}] + \Pr[b = b' | \mathbf{Event}]\cdot\Pr[\mathbf{Event}] \le \Pr[\neg\mathbf{Event}]/2 + \Pr[\mathbf{Event}] = 1/2 + \Pr[\mathbf{Event}]/2$. By the definition of the advantage in **IND-CBPRE-CCA2 Game**, we have $\varepsilon \le 2|\Pr[b = b'] - 1/2| \le \Pr[\mathbf{Event}] \le \Pr[\mathbf{QueryH_4}]/\Pr[\neg\mathbf{Fail}]$. Hence, we get

$$\Pr[\mathbf{QueryH_4}] \ge \varepsilon\Pr[\neg\mathbf{Fail}] \ge \frac{\varepsilon}{q_{uc}}(1-\frac{q_{ren}}{2^k})(1-\frac{q_{dec}}{2^k}). \quad (12)$$

Finally, we get the announced bound on A_{BDH}'s advantage in solving the BDH problem by noting that A_{BDH} selects the correct tuple from the list H_4List with probability $1/q_4$.

4.3 Performance Comparison

To evaluate the performance of the proposed scheme, we provide a comparison of it and the previous two CB-PRE schemes [4, 32]. Without considering pre-computation, the details of the compared schemes are listed in **Table 1**. We denote pairing, exponentiation in G_T, exponentiation in G, map-to-point hash and general hash by P, E_T, E, H_M and H respectively, the bit length of an element in G, an element in G_T and a message by $|G|$, $|G_T|$ and n respectively. In Sur *et al.*'s scheme [4], k_0 denotes the bit-length of the random values used to encrypt the data, which should be at least 160 in order to obtain a reasonable security.

Table 1. Performance of the CB-PRE schemes

Compared Items	Ours	Sur *et al.*'s [4]	Li *et al.*'s [32]
Encrypt	$2P+2E_T+1E+2H_M+2H$	$2P+2E_T+3E+3H_M+1H$	$3P+2E_T+3E+3H_M+2H$
ReKeyGen	$1P+2E+3H_M$	$2P+2E_T+3E+4H_M$	$2P+1E_T+5E+4H_M$
ReEncrypt	$1P$	$7P+1H_M$	$5P+1H_M$
*Decrypt*1	$1P+2E+1H_M+2H$	$2P+1E_T+2E+1H_M+2H$	$4P+2E+2H_M+2H$
*Decrypt*2	$2P+2E+1H_M+2H$	$4P+1E_T+1E+3H_M+2H$	$4P+1E_T+1E+2H_M+2H$
Re-encryption key size	$\|G\|$	$3\|G\|$	$\|G_T\|+4\|G\|$
Original ciphertext size	$\|G_T\|+\|G\|+n$	$3\|G\|+n+k_0$	$2\|G_T\|+2\|G\|+n$
Re-encryption ciphertext size	$\|G_T\|+\|G\|+n$	$2\|G_T\|+2\|G\|+n+k_0$	$2\|G_T\|+2\|G\|+n$

Table 2. Simulation results of the CB-PRE schemes (1024-bit security level)

Compared Items		Ours	Sur *et al.*'s [4]	Li *et al.*'s [32]
Computation cost (ms)	*Encrypt*	63.16	78.96	99.18
	ReKeyGen	41.92	82.02	89.45
	ReEncrypt	20.04	143.32	103.24
	*Decrypt*1	35.84	61.19	98.96
	*Decrypt*2	55.88	100.97	97.93
Communication cost (bits)	Re-encryption key	512	1536	3072
	Original ciphertext	1536 + n	1536 + n + k_0	3072 + n
	Re-encryption ciphertext	1536 + n	3072 + n + k_0	3072 + n

To give a more intuitive comparison, we implement these CB-PRE schemes using the standard cryptographic library MIRACAL [34]. Our experimental platform is a PIV 3-GHZ processor with 512-MB memory and a Windows XP operation system. To achieve the 1024-bit (2048-bit) RSA level security, the Tate pairing defined over the super-singular elliptic curve E/F_p: $y^2 = x^3 + x$ with embedding degree 2 is used, where p is a 512-bit (1024-bit) prime. The simulation results are given in **Table 2** and **Table 3**.

Table 3. Simulation results of the CB-PRE schemes (2048-bit security level)

Compared Items		Ours	Sur *et al.*'s [4]	Li *et al.*'s [32]
Computation cost (ms)	*Encrypt*	522.28	632.89	814.50
	ReKeyGen	333.52	654.30	706.31
	ReEncrypt	181.38	1293.71	928.08
	*Decrypt*1	290.96	508.49	853.04
	*Decrypt*2	471.32	867.15	845.83
Communication cost (bits)	Re-encryption key	1024	3072	6144
	Original ciphertext	3072 + n	3072 + n + k_0	6144 + n
	Re-encryption ciphertext	3072 + n	6144 + n + k_0	6144 + n

The above comparison shows that our scheme is more efficient than the previous two CB-PRE schemes in both the computation cost and the communication cost. Actually, the computation performance of our scheme can be further optimized. If the pairings $e(P_{pub}, Q_A)$ and $e(PK_A, R_A)$ are pre-computed, then the algorithm *Encrypt* of our scheme requires computing only two exponentiations in G_T, one exponentiation in G and two general hashes to encrypt a message.

5. Conclusion

In this paper, we develop an efficient CB-PRE scheme from pairings and prove it to achieve chosen-ciphertext security in the random oracle model. Compared with the previous CB-PRE schemes, the proposed scheme enjoys obvious advantage in both the computation efficiency and the communication bandwidth. However, as pairing operation is extremely disliked by the power-constrained devices, it would be interesting to construct CB-PRE schemes that do not depend on parings. In addition, the security of our scheme can only be achieved in the random

oracle model. So, another interesting problem is to construct secure CB-PRE schemes in the standard model.

References

[1] J.M. Do, Y.J. Song and N. Park, "Attribute based proxy re-encryption for data confidentiality in cloud computing environments," in *Proc. of 2011 First ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, pp. 248-251, 2011. Article (CrossRef Link).

[2] G. Wang, Q. Liu and J. Wu, "Achieving fine-grained access control for secure data sharing on cloud servers," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 12, pp. 1443-1464, 2011. Article (CrossRef Link).

[3] L. Xu, X. Wu and X. Zhang, "CL-PKE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proc. of the 7th ACM Symposium on Information, Computer and Communications Security*, pp. 87-88, 2012. Article (CrossRef Link).

[4] C. Sur, Y. Park, S.U. Shin, K.H. Rhee and C. Seo, "Certificate-based proxy re-encryption for public cloud storage," in *Proc. of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 159-166, 2013. Article (CrossRef Link).

[5] S.H. Lee and I.Y. Lee, "A secure index management scheme for providing data sharing in cloud storage," *Journal of Information Processing Systems*, vol. 9, no. 2, pp. 287-300, 2013. Article (CrossRef Link).

[6] S.H. Seo, M. Nabeel, X. Ding and E. Bertino, "An efficient certificateless encryption for secure data sharing in public clouds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 9, pp. 2107-2119, 2013. Article (CrossRef Link).

[7] K. Liang, M.H. Au, J.K. Liu, W. Susilo, D.S. Wong, G. Yang, T.V.X. Phuong and Q. Xie, "A DFA-based functional proxy re-encryption scheme for secure public cloud data sharing," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 10, pp. 1667-1680, 2014. Article (CrossRef Link).

[8] Q. Liu, G. Wang and J. Wu, "Time-based proxy re-encryption scheme for secure data sharing in a cloud environment," *Information Sciences*, vol. 258, pp. 355-370, 2014. Article (CrossRef Link).

[9] N.D. Han, L. Han, D.M. Tuan, H.P. In and M. Jo, "A scheme for data confidentiality in cloud-assisted wireless body area networks," *Information Sciences*, vol. 284, pp. 157-166, 2014. Article (CrossRef Link).

[10] J.W. Li, J. Li, Z. Liu and C. Jia, "Enabling efficient and secure data sharing in cloud computing," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 5, pp. 1052-1066, 2014. Article (CrossRef Link).

[11] M. Blaze, G. Bleumer and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. of Advances in Cryptology - Eurocrypt 1998*, pp. 127-144, 1998. Article (CrossRef Link).

[12] 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*, vol. 9, no. 1, pp. 1-30, 2006. Article (CrossRef Link).

[13] R. Canetti and S. Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. of the 14th ACM conference on Computer and Communications Security*, pp. 185-194, 2007. Article (CrossRef Link).

[14] R.H. Deng, J. Weng, S. Liu and K. Chen, "Chosen-ciphertext secure proxy re-encryption without pairings," in *Proc. of CANS 2008*, pp. 1-17, 2008. Article (CrossRef Link).

[15] B. Libert and D. Vergnaud, "Unidirectional chosen-ciphertext secure proxy re-encryption," in *Proc. of Public Key Cryptography - PKC 2008*, pp. 360-379, 2008. Article (CrossRef Link).

[16] J. Shao and Z. Cao, "CCA-secure proxy re-encryption without pairings," in *Proc. of Public Key*

Cryptography - PKC 2009, pp. 357-376, 2009. Article (CrossRef Link).
[17] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *Proc. of ACNS 2007*, pp. 288-306, 2007. Article (CrossRef Link).
[18] C.K. Chu and W.G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *Proc. of Information Security 2007*, pp. 189-202, 2007. Article (CrossRef Link).
[19] T. Matsuo, "Proxy re-encryption systems for identity-based encryption," in *Proc. of Pairing-Based Cryptography - Pairing 2007*, pp 247-267, 2007. Article (CrossRef Link).
[20] S. Luo, Q. Shen and Z. Chen, "Fully secure unidirectional identity-based proxy re-encryption," in *Proc. of Information Security and Cryptology - ICISC 2011*, pp. 109-126, 2012. Article (CrossRef Link).
[21] S.S. Al-Riyami and K.G. Paterson, "Certificateless public key cryptography," in *Proc. of Advances in Cryptology - Asiacrypt 2003*, pp. 452-473, 2003.Article (CrossRef Link).
[22] C. Gentry, "Certificate-based encryption and the certificate revocation problem," in *Proc. of Advances in Cryptology - Eurocrypt 2003*, pp. 272-293, 2003. Article (CrossRef Link).
[23] C. Sur, C. D. Jung and K. H. Rhee, "Multi-receiver certificate-based encryption and application to public key broadcast encryption," in *Proc. of 2007 ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security*, pp. 35-40, 2007. Article (CrossRef Link).
[24] D. Galindo, P. Morillo and C. Ràfols, "Improved certificate-based encryption in the standard model," *Journal of Systems and Software*, vol. 81, no. 7, pp. 1218-1226, 2008. Article (CrossRef Link).
[25] J. K. Liu and J. Zhou, "Efficient certificate-based encryption in the standard model," in *Proc. of 6th Int. Conf. on Security and Cryptography for Networks*, pp. 144-155, 2008. Article (CrossRef Link).
[26] Y. Lu, J. Li and J. Xiao, "Constructing efficient certificate-based encryption with paring," *Journal of Computers*, vol. 4, no. 1, pp. 19-26, 2009. Article (CrossRef Link).
[27] Z. Shao, "Enhanced certificate-based encryption from pairings," *Computers and Electrical Engineering*, vol. 37, no. 2, pp. 136-146, 2011. Article (CrossRef Link).
[28] J. Yao, J. Li and Y. Zhang, "Certificate-based encryption scheme without pairing," *KSII Transactions on Internet and Information Systems*, vol. 7, no. 6, pp. 1480-1491, 2013. Article (CrossRef Link).
[29] T. Hyla, W. Maćków and J. Pejaś, "Implicit and explicit certificates-based encryption scheme," in *Proc. of the 13th IFIP TC8 International Conference on Computer Information Systems and Industrial Management*, pp. 651-666, 2014. Article (CrossRef Link).
[30] Y. Lu and J. Li, "Efficient construction of certificate-based encryption secure against public key replacement attacks in the standard model," *Journal of Information Science and Engineering*, vol. 30, no. 5, pp. 1553-1568, 2014. Article (CrossRef Link).
[31] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," in *Proc. of* 1st *ACM Conf. on Communications and Computer Security*, pp. 62-73, 1993. Article (CrossRef Link).
[32] J. Li, X. Zhao and Y. Zhang, "Certificate-based conditional proxy re-encryption," in *Proc. of the 8th International Conference on Network and System Security*, pp. 299-310, 2014. Article (CrossRef Link).
[33] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Proc. of Advances in Cryptology - Crypto 2001*, pp. 213-229, 2001. Article (CrossRef Link).
[34] MIRACL, Multiprecision integer and rational arithmetic cryptographic library, http://certivox.org/display/EXT/MIRACL. Article (CrossRef Link).

Yang Lu received the Ph.D. degree from PLA University of Science and Technology in 2009. He has been working in HoHai University from 2003. Currently, he is an Assistant Professor in College of Computer and Information Engineering. His major research interests include information security and cryptography, network security and cloud security, etc. He has published more than 30 scientific papers in international conferences and journals.