Efficient and Secure Certificateless Proxy Re-Encryption

Ya Liu[1,2], Hongbing Wang[3] and Chunlu Wang[4,5]
[1] Department of Computer Science and Engineering, University of Shanghai for Science and Technology
Shanghai, 200093 – P.R. China
[2] Engineering Research Center of Optical Instrument and System, Ministry of Education , Shanghai Key Lab of Modern Optical System, University of Shanghai for Science and Technology, 200093 – P.R. China
[e-mail: liuya@usst.edu.cn]
[3] Shanghai Key Laboratory of Data Science, Software School, Fudan University
Shanghai, 201203 - P.R. China
[e-mail: wanghongbing@fudan.edu.cn]
[4] School of Computer Science, Beijing University of Posts and Telecommunications
Beijing, 100876 - P.R. China
[e-mail: wangcl@bupt.edu.cn]
[5] Key Laboratory of Trustworthy Distributed Computing and Service(BUPT), Ministry of Education
Beijing, 100876 - P.R. China
[e-mail: wangcl@bupt.edu.cn]
*Corresponding author: Hongbing Wang

Received September 29, 2016; revised December 23, 2016; accepted February 1, 2017; published April 30, 2017

Abstract

In this paper, we present an IND-CCA2 secure certificateless proxy re-encryption scheme in the random oracle model. A certificateless public key cryptography simplifies the certificate management in a traditional public key infrastructure and the built-in key escrow feature in an identity-based public key cryptography. Our scheme shares the merits of certificateless public key encryption cryptosystems and proxy re-encryption cryptosystems. Our certificateless proxy re-encryption scheme has several practical and useful properties - namely, multi-use, unidirectionality, non-interactivity, non-transitivity and so on. The security of our scheme bases on the standard bilinear Diffie-Hellman and the decisional Bilinear Diffie-Hellman assumptions.

Keywords: Proxy re-encryption, bilinear pairing, unidirectional, multi-use, chosen-ciphertext security, random oracles

The first author was supported by the National Natural Science Foundation of China (Nos. 61402288,61370068) and Shanghai Engineering Research Center Project (No. GCZX14014, No. C14001). The second author was partially supported by NSFC (Grant Nos. 61472084, 61272012, U1536205), Shanghai Innovation Action Project No. 16DZ1100200, and Project funded by China Postdoctoral Science Foundation. The third author was partially supported by the National Natural Science Foundation of China (No. 61370068).

https://doi.org/10.3837/tiis.2017.04.023 **ISSN : 1976-7277**

1. Introduction

For decades of years, cryptographic researchers have been engaging in developing proxy re-encryption (PRE) schemes, since PRE is realized to have wide use in digital era. Usually, we need to transform our ciphertexts to others in order to let them fulfill the decryption for us when we are busy or not convenient to access to internet. The main idea of PRE is the transformation of ciphertexts. Before the invention of PRE, to change a ciphertext under one key to another needs to decrypt the ciphertext first, then encrypt it using the new key. Intuitively, this implies the exposition of the plaintext, and is not desirable in many situations. While, PRE gives a perfect solution for ciphertext transformation. In a proxy re-encryption, a semi-trusted proxy equipped with some additional information can convert a ciphertext encrypted under a delegator's key into an encryption of the same message under a delegatee's key, while, during the translation, no information is revealed to the proxy except for the necessary one to allow it to perform its transformation. At the very least, the proxy should not be able to learn the keys of the participants or the content of the message it re-encrypts. The concept of PRE is brought out by Blaze et. [1] in 1989, and the first PRE scheme which is based on ElGamal [2] system and achieves IND-CPA secure is also given in [1]. Blaze et.'s work [1] promotes the development of PRE research. There now exist public key PRE schemes, such as [1,3-7]. However, the main problem [8] which exists in a public key cryptosystem (PKC) also exists in a public key PRE cryptosystem. The most noted issues are associated with certificate management, including revocation, storage, distribution, etc..

In order to simplify the above mentioned problem, i.e., the certificate management in PKC, Shamir [9] proposed the idea of identity-based cryptosystem(IBC). In an IBC, a user's public key is his identifier, and his private key is generated by the private key generator (PKG) using PKG's master secret key and the user's identifier. Since a user's public key is uniquely determined by the user's identifer, and can be publicly obtained, there are no need of certificates in an IBC anymore. An identity-based proxy re-encrypiton (ID-PRE) shares this advantage of identity-based encryption. In 2006, Green and Ateniese [10] proposed the first ID-PRE scheme. Though the certificate management is simplified in ID-PKC or ID-PRE, there arises another problem in these two systems. Due to the exorbitant dependence on a PKG to generate all the private keys of users, ID-PKC has the problem of key escrow, and this problem is certainly remained in an ID-PRE. For PKG, who uses a system-wide master-key to generate all private keys in ID-PKC, can decrypt any ciphertext in an ID-PRE scheme. Moreover, in ID-PRE cryptosystems, since the PKG knows all the users' private keys, he can generate any proxy re-encryption keys.

In order to eliminate the requirement of the use of certificates in a traditional public key infrastructure (PKI) and the built-in key escrow feature in an ID-PKC, Al-Riyami and Paterson proposed certificateless public key cryptography (CL-PKC) in [11], followed by some works such as [11-20]. In a CL-PKC, a trusted third party named Key Generating Center (KGC) is used to supply a partial private key D_A to entity A. The KGC computes D_A from ID_A (an identifier of entity A) and a master-key owned only by the KGC itself. The KGC must ensure that the partial private keys are delivered securely to the correct entities. On receiving the partial private key D_A , A combines it with some secret information to generate its actual private key S_A, so A's private key is not available to the KGC. A also combines its secret information with the KGC's public parameters to generate its public key P_A. A's public key might be made available to other entities by transmitting it along with messages or by placing

it in a public directory. CL-PKC has more applications, see for examples [11]. A certificateless proxy re-encryption (CL-PRE) inherits these merits of CL-PKC. In a CL-PRE, the key of an entity is generated in the same way as in a CL-PKC, i.e., the entity chooses a secret information and combines it with the partial private key supplied by the KGC according to its identifier to generate its actual private key, and also combines its secret information with the KGC's public parameters to generate its public key. Thus, there is no certificate management and key escrow problems existed in a CL-PRE. Recent work about PRE.shows that PRE has widely used in currently hot-studied cloud computing and cloud storage [21-26].

1.1 Related Work

In 2010, Sur et.al. [21] constructed a certificateless proxy re-encryption scheme which is IND-CCA secure in the random oracle model using bilinear pairing. In 2011 and 2012, two IND-CPA secure CL-PREs [22-23] were proposed. The schemes are also implemented using bilinear paring in the random oracle model.

In this paper, we construct a certificateless proxy re-encryption scheme which solves the certificate management problem in a PRE scheme and the key escrow problem in an ID-PRE scheme simultaneously. Moreover, our CL-PRE satisfies more useful properties such as multi-use, unidirectionality, non-interactivity and non-transitivity, etc. Compared with the existing CL-PRE schemes [21-23], our scheme wins either in security level (CCA vs. CPA [22-23]) or more PRE attributions and more standard computational assumption [21]. (See **Table 1**)

Table 1. Properties of PRE

Attributions	SJPR([21])	XWZ12[22,23]	Our CL-PRE
Unidirectional	Yes	Yes	Yes
Multi-use	No	No	Yes
Non-interactive	Yes	Yes	Yes
Non-transitive	Yes	Yes	Yes
Secure	CCA	CPA	CCA
Without RO	No	No	No

1.2 Paper Organization

In section 2, we introduce necessary preliminaries, including definitions, computational assumptions. In Section 3, we describe the security model for CL-PRE. Then, our CL-PRE is constructed in Section 4, followed by the security proof in Section 5. We analyze the performance of the existing CL-PRE schemes and ours in Section 6. Finally, We draw our concluding remarks in Section 7.

2. Preliminaries

Definition 1 (Bilinear Map) Let $\mathbb{G}_1$ be an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same prime order. Further, we let P denote a generator of $\mathbb{G}_1$. Assume that the discrete logarithm problems in both $\mathbb{G}_1$ and $\mathbb{G}_2$ are intractable. We say that e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear or pairing map if it satisfies the following properties:

1. Bilinear: given $Q, W, T \in \mathbb{G}_1$, we have
   $e(Q, W+T) = e(Q, W) \cdot e(Q, T),\ e(Q+W, T) = e(Q, T) \cdot e(W, T)$.

2. Non-degeneate: $e(P,P) \neq 1_{\mathbb{G}_2}$, i.e., if P generates $\mathbb{G}_1$, then $e(P,P)$ generates $\mathbb{G}_2$.
3. Computable: The map e is efficiently computable.

We denote $\mathcal{IG}$ as an algorithm that, on input the security parameter 1^κ, outputs the parameters for a bilinear map as $(q, \mathbb{G}_1, \mathbb{G}_2, e)$, where q $\in \Theta(2^\kappa)$.

Definition 2 (Bilinear Diffie-Hellman (BDH) Problem) Let $\mathbb{G}_1, \mathbb{G}_2, e$ and P be parameters defined as above. The BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given $\langle P, aP, bP, cP \rangle \in \mathbb{G}_1^4$, (where $a, b, c \in \mathbb{Z}_q^*$), compute $e(P,P)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ε in solving the BDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$, if $\Pr[\mathcal{A}(\langle P, aP, bP, cP \rangle) = e(P,P)^{abc}] = \varepsilon$.

Definition3 (Decisional Bilinear Diffie-Hellman (DBDH) Problem) Let $\mathbb{G}_1, \mathbb{G}_2, e$ and P be as above. The DBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ is as follows: Given a tuple of $\langle P, aP, bP, cP, T \rangle \in \mathbb{G}_1^4 \times \mathbb{G}_2$ (where $a, b, c \in \mathbb{Z}_q^*$), decide whether $T = e(P,P)^{abc}$ holds.

Let κ be a security parameter of sufficient size. Formally, we say that the DBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$, if for all probabilistic polynomial time algorithms (PPT) $\mathcal{A}$, the following condition is true, where $v(\cdot)$ is defined as a negligible function.:

$$\left| \begin{matrix} \Pr\left[a, b, c \leftarrow_R \mathbb{Z}_q^*; 1 \leftarrow \mathcal{A}\left(P, aP, bP, cP, e(P,P)^{abc}\right)\right] - \\ \Pr\left[a, b, c \leftarrow_R \mathbb{Z}_q^*; T \leftarrow_R \mathbb{G}_2; 1 \leftarrow \mathcal{A}(P, aP, bP, cP, T)\right] \end{matrix} \right| \leq v(\kappa),$$

3. Models and Security

1.3 3.1 Certificateless Proxy Re-encryption

Definition 4 (Certificateless Proxy Re-encryption (CL-PRE)) A CL-PRE scheme is specified by the following nine algorithms:

1. **Setup(1^κ):** On input of security parameter $\boldsymbol{\kappa}$, it returns the system's parameters par and master-key msk. Usually, this algorithm is run by KGC. We assume that par is publicly and authentically available, but that only the KGC knows msk [11].
2. **Partial-Private-Key(par, msk, id_i)**: On input of par, msk, and an identifier $id_i \in \{0,1\}^*$, it returns a partial private key D_{id_i}.
3. **Set-Secret-Value(par, id_i)**: On input of par and an identifier $id_i \in \{0,1\}^*$, the algorithm outputs a secret value x_i for entity with identifier id_i.
4. **Set-Private-Key (par,D_{id_i},x_i)**: On input of par, the partial private key D_{id_i} of id_i, and that entity's secret value x_i, it returns private key S_{id_i} for entity with identifier id_i.
5. **Set-Public-Key (par,x_i)**: On input of par and the secret value x_i of entity with identifier id_i, the algorithm outputs the public key P_{id_i} for that entity.
6. **Enc (par, M, P_{id_i}, id_i)**: On input of par, a message $M \in \mathcal{M}$, an entity's public key P_{id_i} and its identifier id_i, the algorithm outputs either a ciphertext $C \in \mathcal{C}$ or $\perp$.
7. **Set-ReKey(par, S_{id_i}, P_{id_j})**: On input par, the private key S_{id_i} of the delegator id_i, and the public key P_{id_j} of the delegatee id_j, it outputs a re-encryption key $rk_{i \to j}$ $(i \neq j)$.
8. **ReEnc(par, $rk_{i \to j}$, C_{id_i})**: On input par, a ciphertext C_{id_i} under the delegator id_i, and a re-encryption key $rk_{i \to j}$ $(i \neq j)$, it outputs a ciphertext C_{id_j} under the delegatee id_j.
9. **Dec (par, C_{id_i}, S_{id_i})**: On input of par, $C_{id_i} \in \mathcal{C}$, and a private key S_{id_i}, the algorithm outputs a message $M \in \mathcal{M}$ or $\perp$ (a symbol indicating a decryption failure).

Consistency. We say that a CL-PRE scheme is consistent if for any valid entities id_i, id_j, and its corresponding public/private key pair (P_{id_i}, S_{id_i}), (P_{id_j}, S_{id_j}), which are generated by **Set-Public-Key** and **Set-Private-Key** respectively, and the corresponding re-encryption key $rk_{i\to j}$ which is generated by **Set-ReKey**, the following equations hold for all $M \in \mathcal{M}$:

$$\mathbf{Dec}(\text{par}, \mathbf{Enc}(\text{par}, M, P_{id_i}, id_i), S_{id_i}) = M;$$

$$\mathbf{Dec}\left(\text{par}, \mathbf{ReEnc}\left(\text{par}, rk_{i\to j}, \mathbf{Enc}(\text{par}, M, P_{id_i}, id_i)\right), S_{id_i}\right) = M.$$

1.4 3.2 Chosen-Ciphertext Security for CL-PRE

First, we define two kinds of adversaries who may involve in a security game.

- **Type I adversary:** A type I adversary $\mathcal{A}_I$ can issue partial private key extraction queries, private key extraction queries, re-encryption key extraction queries, re-encryption queries, decryption queries, can request public keys and request to replace public keys with values of his choice. However, he does not have access to master-key.
- **Type II adversary:** A Type II adversary $\mathcal{A}_{II}$ is equipped with the master-key, thus, he can compute partial private key for himself. He can issue partial private key extraction queries for others, can issue private key extraction queries, re-encryption key queries, re-encryption queries, decryption queries, can request public keys, but he can not request to replace public keys at any point.

Let $\mathcal{E}$ be a CL-PRE scheme defined as above. We consider the following game, denoted by $Game_{\mathcal{E},\mathcal{A}}^{\text{IND}-\text{CLPr}-\text{CCA2}}$, in which a PPT adversary $\mathcal{A}$ (while $\mathcal{A}$ maybe a Type I or a Type II adversary as defined above) is involved:

1. SETUP. On input of a security parameter κ, the challenger runs the **Setup** algorithm to generate the system's parameters par. The challenger gives par to $\mathcal{A}$. If $\mathcal{A}$ is a Type I adversary, then the challenger keeps the master-key msk to himself. Otherwise, if $\mathcal{A}$ is a Type II adversary, msk is also given to $\mathcal{A}$.
2. PHASE 1. $\mathcal{A}$ makes queries $q_1, \cdots, q_m$, where q_i $(i = 1, \cdots, m)$ is one of the following:
   (a) **Partial Private Key** $\mathcal{O}_{psk}(id_i)$: On input an identifier id_i, the challenger responds $\mathcal{A}$ by running **Partial-Private-Key** to generate the partial private key D_{id_i} for id_i.
   (b) **Private Key** $\mathcal{O}_{\mathbf{sk}}(id_i)$: On input an identifier id_i, if $\mathcal{A}$ is a Type II adversary, or if $\mathcal{A}$ is a Type I adversary and the public key of id_i has not been replaced, the challenger responds $\mathcal{A}$ by running algorithm **Set-Private-Key** to generate the private key S_{id_i} for id_i (running **Set-Secret-Value** for id_i first if necessary). Otherwise, if $\mathcal{A}$ is a Type I adversary and the public key of id_i has been replaced, the challenger returns $\perp$.
   (c) **Request for Public Key** $\mathcal{O}_{pk}(id_i)$: On input an identifier id_i, the challenger responds $\mathcal{A}$ by running the algorithm **Set-Public-Key** to generate the public key P_{id_i} for entity id_i (running **Set-Secret-Value** for id_i first if necessary).
   (d) **Replace Public Key** $\mathcal{O}_{rpk}(id_i, P'_{id_i})$: On input $(id_i, P'_{id_i} = (X'_i, Y'_i))$, where P'_{id_i} is a new public key for id_i chosen by $\mathcal{A}$, the challenger first checks if P'_{id_i} is a valid public key. If not, the challenger returns $\perp$ to $\mathcal{A}$. Otherwise, the challenger accepts the new value for entity id_i. Thereafter, the challenger will use P'_{id_i} in any computation for id_i. Note that a Type II adversary is not allowed to make this request.

(e) **ReKey $\mathcal{O}_{rk}(id_i, id_j)$ ($i \neq j$)**: On input id_i and id_j for $i \neq j$, the challenger responds $\mathcal{A}$ by running the algorithm **Set-ReKey** to generate the re-encryption key $rk_{i\to j}$ from id_i to id_j (running **Set-Private-Key** for id_i first if necessary).

(f) **Reencryption $\mathcal{O}_{reen}(id_i, id_j, C_{id_i})$ ($i \neq j$)**: On input two identifiers id_i and id_j for $i \neq j$, and a ciphertext C_{id_i} under id_i, the challenger first derives a re-encryption key $rk_{i\to j}$ by running **Set-ReKey**, then returns $C_{id_j} = \mathbf{ReEnc}(\text{par}, rk_{i\to j}, C_{id_i})$ to $\mathcal{A}$.

(g) **Decryption $\mathcal{O}_{dec}(id_i, C_{id_i})$**: On input an identifier id_i, and a ciphertext C_{id_i} under it, the challenger's respondence is discussed in the following two cases:

i. **Case** 1: If $\mathcal{A}$ is a Type II adversary, or if $\mathcal{A}$ is a Type I adversary and the public key of entity with identifier id_i has not been replaced, the challenger responds $\mathcal{A}$ by running the algorithm **Set-Private-Key** to obtain the private key S_{id_i}, then runs the algorithm $\mathbf{Dec}(\text{par}, C_{id_i}, S_{id_i})$, and returns the result to $\mathcal{A}$.

ii. **Case 2**: If $\mathcal{A}$ is a Type I adversary and the public key of entity with identifier id_i has been replaced, the challenger responds $\mathcal{A}$ by using special purpose knowledge extractors in our security proofs.

3. CHOICE AND CHALLENGE. $\mathcal{A}$ decides PHASE 1 is over, he presents his choice (id^*, M_0, M_1) with the following restrictions:

(a) $\mathcal{A}$ has not previously issued $\mathcal{O}_{sk}(id^*)$.

(b) If $\mathcal{A}$ is a Type I adversary, he must have not previously issued $\mathcal{O}_{psk}(id^*)$ if he had already request to replace the public key for id^*.

(c) If $\mathcal{A}$ is a Type I adversary, he must have not requested to replace the public key for id^* if he had already issued $\mathcal{O}_{psk}(id^*)$.

(d) $\mathcal{A}$ has not previously issued any $\mathcal{O}_{rk}$ queries which will enable $\mathcal{A}$ to compute a re-encryption key from id^* to some entity which $\mathcal{A}$ holds the private key.

(e) M_0 and M_1 are of the same length from the message space.

Then, the challenger selects $d \in \{0,1\}$ at random, and computes $C^* = \mathbf{Enc}(\text{par}, M_d, P_{id^*}, id^*)$, where P_{id^*} is the current public key of entity with identifier id^*. The challenger gives C^* to $\mathcal{A}$ as the challenge ciphertext.

4. PHASE 2. $\mathcal{A}$ continues to make queries as in PHASE 1 with the following restrictions:

Let $\mathcal{R}$ be a set of ciphertext/identifier pairs, initially containing the single pair (C^*, id^*). For all $C \in \mathcal{C}$ and for all rk given to $\mathcal{A}$ or can be computed by $\mathcal{A}$. Let $\mathcal{R}'$ be the set of all possible values derived via (one or more) consecutive calls to **ReEnc**:

(a) $\mathcal{A}$ is not permitted to issue any $\mathcal{O}_{sk}(id^*)$ query.

(b) If $\mathcal{A}$ is a Type I adversary, he is not permitted to issue any $\mathcal{O}_{psk}(id^*)$ query if he has replaced the public key for id^* in PHASE 1.

(c) If $\mathcal{A}$ is a Type I adversary, he is not permitted to replace the public key for id^* if he had already issued $\mathcal{O}_{psk}(id^*)$.

(d) $\mathcal{A}$ is not permitted to issue any $\mathcal{O}_{dec}(P_{id^*}, C^*)$ query, where P_{id^*} was used to encrypt M_d in CHOICE AND CHALLENGE stage.

(e) $\mathcal{A}$ is not permitted to issue any $\mathcal{O}_{dec}(P_{id}, C)$ query, where $(C, id) \in \mathcal{R} \cup \mathcal{R}'$.

(f) $\mathcal{A}$ is not permitted to issue $\mathcal{O}_{sk}(id)$ query or $\mathcal{O}_{rk}(id_i, id_j)$ query that would permit trivial decryption of any ciphertext in $(\mathcal{R} \cup \mathcal{R}')$.

(g) $\mathcal{A}$ is not permitted to issue $\mathcal{O}_{reen}(id_i, id_j, C_{id_i})$, where $\mathcal{A}$ possesses the keys to trivially decrypt ciphertexts under id_j and $(C_{id_i}, id_i) \in (\mathcal{R} \cup \mathcal{R}')$. On successful execution of any re-encryption query, let C_{id_j} be the result, the challenger adds the pair (C_{id_j}, id_j) to the set $\mathcal{R}$.

5. GUESS. $\mathcal{A}$ outputs a guess d', where $d' \in \{0, 1\}$. If $d = d'$, then $\mathcal{A}$ wins the game.

Let $\mathcal{O}_s = (\mathcal{O}_{psk}, \mathcal{O}_{sk}, \mathcal{O}_{pk}, \mathcal{O}_{rpk}, \mathcal{O}_{rk}, \mathcal{O}_{reen}, \mathcal{O}_{dec})$, and $\mathcal{O}'_s = (\mathcal{O}'_{psk}, \mathcal{O}'_{sk}, \mathcal{O}'_{pk}, \mathcal{O}'_{rpk}, \mathcal{O}'_{rk}, \mathcal{O}'_{reen}, \mathcal{O}'_{dec})$ be the corresponding oracles modified in PHASE 2. With respect to the system's security parameter κ,$\mathcal{A}$'s advantage in the above game $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}^{\mathrm{IND-CLPr-CCA2}}(\kappa)$ is defined as the following:

$$\left| \Pr\left[d' = d \left| \begin{array}{c} d \leftarrow_R \{0,1\}; \\ (\mathrm{par}, \mathrm{msk}) \leftarrow \mathbf{Setup}(1^{\kappa}); \\ (id^*, M_0, M_1, t) \leftarrow \mathcal{A}^{\mathcal{O}_s}(\mathrm{par}); \\ C^* \leftarrow \mathbf{Enc}(\mathrm{par}, M_d, P_{id^*}, id^*); \\ d' \leftarrow \mathcal{A}^{\mathcal{O}'_s}(\mathrm{par}, C^*, t) \end{array} \right. \right] - \frac{1}{2} \right|$$

We say that $\mathcal{E}$ is semantically secure against an adaptive chosen-ciphertext attack IND-CLPr-CCA2, if for all probabilistic polynomial time algorithms $\mathcal{A}$, $\mathrm{Adv}_{\mathcal{E},\mathcal{A}}^{\mathrm{IND-CLPr-CCA2}}(\kappa)$ is negligible with respect to κ.

4. Our Constructions

In this part, we describe our CL-PRE scheme. Our scheme is IND-CLPr-CCA2 secure with appealing features, such as unidirectionality, multi-use, non-interactivity, and non-transitivity.

To achieve public verifiability for ℓ^{th}-*level* $(\ell > 1)$ ciphertexts, Canetti-Halevi-Katz's technique [27] is used, using a strongly unforgeable one-time signature. According to [28], a strongly unforgeable one-time signature consists of a triple of algorithms $\mathrm{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ such that, on input of a security parameter κ, $\mathcal{G}$ generates a one-time key pair (svk, ssk), while, for any message $M, \mathcal{V}(svk, \sigma, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$, and 0 otherwise. It is reasonable to assume that the verification key for each entity is publicly available.

Scheme Description.

Let κ be a security parameter, $\mathcal{IG}$ be a BDH parameter generator with input κ, and $\mathrm{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ be a strongly unforgeable signature scheme. For simplicity, we assume that verification keys of the one-time signature are encoded as elements from $\mathbb{Z}_q^*$. In practice, such verification keys are typically much longer than $|q|$ and a collision-resistent hash function should be applied to map them onto $\mathbb{Z}_q^*$.

We let P_i denote the proxy to create a re-encrypted ciphertext from entity id_i to entity id_{i+1} in a multi-use re-encryption scheme. Now, we describe our CL-PRE scheme as below:

1. **Setup (1^{κ})**: This algorithm runs as follows:
   (a) Run $\mathcal{IG}$ on input κ to generate $\langle \mathbb{G}_1, \mathbb{G}_2, e, q \rangle$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of some prime order q and e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a pairing.
   (b) Choose an arbitrary generator $P \in \mathbb{G}_1$, and two arbitrary elements $P_1, P_2 \in \mathbb{G}_1 \backslash \{P\}$.
   (c) Select a master-key s uniformly at random from $\mathbb{Z}_q^*$ and set $P_0 = sP$.
   (d) Choose a strongly unforgeable one-time signature scheme $\mathrm{Sig} = (\mathcal{G}, \mathcal{S}, \mathcal{V})$.

(e) Choose three cryptographic hash functions $H_1: \{0,1\}^* \to \mathbb{G}_1^*$, $H_2: \{0,1\}^* \to \mathbb{Z}_q^*$, and $H_3: \mathbb{G}_2 \to \mathbb{G}_1$. The system parameters are:

$$par = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, P_1, P_2, Sig, H_1, H_2, H_3 \rangle.$$

The master-key msk is $s \in \mathbb{Z}_q^*$. The message space is $\mathcal{M} = \mathbb{G}_2$.

2. **Partial-Private-Key** (par, msk, id_i): This algorithm takes as input par, msk, and an identifier $id_i \in \{0,1\}^*$, computes $Q_{id_i} = H_1(id_i) \in \mathbb{G}_1^*$; and outputs the partial private key $D_{id_i} = sQ_{id_i} \in \mathbb{G}_1^*$ for entity with identifier id_i:
3. **Set-Secret-Value**(par, id_i): This algorithm takes as input par and an identifier id_i. Then, it selects $x_i \in \mathbb{Z}_q^*$ at random and outputs x_i as the secret value of entity with identifier id_i.
4. **Set-Private-Key**(par, D_{id_i}, x_i): This algorithm takes as inputs par, an entity's partial private key D_{id_i} and its secret value $x_i \in \mathbb{Z}_q^*$. It transforms the entity's partial private key D_{id_i} to its private key S_{id_i} by computing $S_{id_i} = x_i D_{id_i} = x_i s Q_{id_i} \in \mathbb{G}_1^*$.
5. **Set-Public-Key**(par, x_i): This algorithm takes as input par and an entity's secret value $x_i \in \mathbb{Z}_q^*$, then, constructs the public key for that entity as $P_{id_i} = \langle X_{id_i}, Y_{id_i} \rangle$, where $X_{id_i} = x_i P$ and $Y_{id_i} = x_i P_0 = x_i s P$.
6. **Enc**(par, M, P_{id_i}, id_i): To encrypt a message $M \in \mathcal{M}$ for entity with identifier $id_i \in \{0,1\}^*$ and public key $P_{id_i} = \langle X_{id_i}, Y_{id_i} \rangle$, this algorithm performs the following
   (a) Check that $X_{id_i}, Y_{id_i} \in \mathbb{G}_1^*$, and whether the equality $e(X_{id_i}, P_0) = e(Y_{id_i}, P)$ holds. If not, output $\perp$ and aborts encryption. Otherwise, compute $Q_{id_i} = H_1(id_i) \in \mathbb{G}_1^*$.
   (b) Choose $r \in \mathbb{Z}_q^*$ at random.
   (c) Compute and output the *first-level* ciphertext as $C_i^{(1)} = (C_{1,1}, C_{1,2}, C_{1,3})$, where $C_{1,1} = rP$, $C_{1,2} = M \cdot e(Q_{id_i}, Y_{id_i})^r$ and $C_{1,3} = r(H_2(C_{1,1} \| C_{1,2})P_1 + P_2)$.
7. **Set-ReKey**(par, S_{id_i}, P_{id_j}), $(i \neq j)$: Let svk_{P_i} be the verification key of proxy P_i, to generate a re-encryption key for the proxy P_i, the entity with identifier id_i selects $r_i \in_R \mathbb{Z}_q^*$, $K_i \in_R \mathbb{G}_2$ and computes $R_1^{(i)} = r_i P, R_2^{(i)} = K_i \cdot e\left(Q_{id_j}, Y_{id_j}\right)^{r_i}, R_3^{(i)} = svk_{P_i}$,

$$R_4^{(i)} = r_i\left(H_2\left(R_3^{(i)} \cdot R_1^{(i)} \| R_2^{(i)}\right)P_1 + P_2\right), R_5^{(i)} = H_2(K_i) + S_{id_i}^{-1},$$

where $Q_{id_j} = H_1(id_j)$ and $P_{id_j} = \langle X_{id_j}, Y_{id_j} \rangle = \langle x_j P, x_j s P \rangle$. Finally, this algorithm outputs $rk_{i \to j} = \left(R_1^{(i)}, R_2^{(i)}, R_3^{(i)}, R_4^{(i)}, R_5^{(i)}\right)$.
8. ReEnc (par, $rk_{i \to j}, C_{id_i}$), $(i \neq j)$:
   (a) To re-encrypt a *first-level* ciphertext C_{id_i}, denoted by $C_i^{(1)}$, the proxy P_1 does :
      i. Parse $C_i^{(1)}$ as $(C_{1,1}, C_{1,2}, C_{1,3})$, and $rk_{i \to j}$ as $\left(R_1^{(1)}, R_2^{(1)}, R_3^{(1)}, R_4^{(1)}, R_5^{(1)}\right)$.
      ii. Check if $e(P, C_{1,3}) = e(C_{1,1}, H_2(C_{1,1} \| C_{1,2})P_1 + P_2)$ and $e\left(P, R_4^{(1)}\right) = e(R_1^{(1)}, H_2\left(R_3^{(1)} \cdot R_1^{(1)} \| R_2^{(1)}\right)P_1 + P_2)$ hold. If not, return $\perp$.
      iii. Otherwise, compute $C = (C'_{1,1}, C'_{1,2}, C'_{1,3}, C'_{2,1}, C'_{2,2}, C'_{2,3}, C'_{2,4})$, where $C'_{1,1} = C_{1,1}$, $C'_{1,2} = C_{1,2} \cdot e\left(R_5^{(1)}, C_{1,1}\right), C'_{1,3} = C_{1,3}$, $C'_{2,1} = R_1^{(1)}$, $C'_{2,2} = R_2^{(1)}$, $C'_{2,3} = R_3^{(1)}$, $C'_{2,4} = R_4^{(1)}$.

iv. Let ssk_{P_1} be the signing key of the proxy P_1 whose verification key is $R_3^{(1)}$.

v. Run $\mathcal{S}(ssk_{P_1}, (C'_{1,1}, C'_{1,2}, C'_{1,3}, C'_{2,1}, C'_{2,3}, C'_{2,4}))$ to generate a signature on the ciphertext tuple $(C'_{1,1}, C'_{1,2}, C'_{1,3}, C'_{2,1}, C'_{2,3}, C'_{2,4})$, and denote the signature as S_1.

vi. Output the ciphertext $C_j^{(2)} = \langle C, S_1 \rangle$.

(b) To re-encrypt an ℓ^{th}-*level* ($\ell > 1$) ciphertext C_{id_i}, denoted by $C_i^{(\ell)}$, the proxy P_ℓ:

i. Parse $C_i^{(\ell)}$ as $(C_{1,1}, C_{1,2}, C_{1,3}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$, and $rk_{i\to j}$ as $\left(R_1^{(\ell)}, R_2^{(\ell)}, R_3^{(\ell)}, R_4^{(\ell)}, R_5^{(\ell)}\right)$.

ii. Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \mathrm{H}_2(C_{\ell,3} \cdot C_{\ell,1} \parallel C_{\ell,2})P_1 + P_2)$ and $e\left(P, R_4^{(\ell)}\right) = e(R_1^{(\ell)}, \mathrm{H}_2\left(R_3^{(\ell)} \cdot R_1^{(\ell)} \parallel R_2^{(\ell)}\right)P_1 + P_2)$ hold. If not, return $\perp$.

iii. Otherwise, for $\forall k \in [2, \ell]$, check if $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right) = 1$. Whenever one of them fails, return $\perp$. Otherwise, compute $C = (C'_{1,1}, \cdots, C'_{\ell,1}, C'_{\ell,2}, C'_{\ell,3}, C'_{\ell,4}, C'_{\ell,5}, C'_{\ell+1,1}, C'_{\ell+1,2}, C'_{\ell+1,3}, C'_{\ell+1,4})$ where $C'_{\ell,2} = C_{\ell,2} \cdot e(R_5^{(\ell)}, C_{\ell,1})$, $C'_{\ell+1,1} = R_1^{(\ell)}$, $C'_{\ell+1,2} = R_2^{(\ell)}$, $C'_{\ell+1,3} = R_3^{(\ell)}$, $C'_{\ell+1,4} = R_4^{(\ell)}$ and all other elements keep unchanged.

iv. Let ssk_{P_ℓ} be the signing key of the proxy P_ℓ whose verification key is $R_3^{(\ell)}$.

v. Run $\mathcal{S}(ssk_{P_\ell}, (C'_{1,1}, \cdots, C'_{\ell+1,1}, C'_{\ell+1,3}, C'_{\ell+1,4}))$ to generate a signature on the ciphertext tuple $(C'_{1,1}, \cdots, C'_{\ell,1}, C'_{\ell,2}, C'_{\ell,3}, C'_{\ell,4}, C'_{\ell,5}, C'_{\ell+1,1}, C'_{\ell+1,3}, C'_{\ell+1,4})$, and denote the signature as S_ℓ.

vi. Output the ciphertext $C_j^{(\ell+1)} = \langle C, S_\ell \rangle$.

9. **Dec**(par, C_{id_i}, S_{id_i}): If C_{id_i} can not be parsed as $(C_{1,1}, C_{1,2}, C_{1,3})$ for a *first-level* ciphertext, or $(C_{1,1}, \cdots, C_{\ell,1}, \cdots, C_{\ell,5})$ for an ℓ^{th}-*level* ciphertext ($\ell > 1$), then return $\perp$. Otherwise:

(a) If $\ell = 1$, then perform

i. Parse C_{id_i} as $(C_{1,1}, C_{1,2}, C_{1,3})$.

ii. Verify that $e(P, C_{1,3}) = e(C_{1,1}, H_2(C_{1,1} \parallel C_{1,2})P_1 + P_2)$. If not, return $\perp$.

iii. Otherwise, compute and output $M \leftarrow C_{1,2}/e(S_{id_i}, C_{1,1})$.

(b) If $\ell > 1$, perform

i. Parse C_{id_i} as $(C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$.

ii. Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \mathrm{H}_2(C_{\ell,3} \cdot C_{\ell,1} \parallel C_{\ell,2})P_1 + P_2)$. If not, return $\perp$. do the following performance:

iii. Otherwise, for $\forall k \in [2, \ell]$, check if $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right) = 1$. Whenever one of them fails, return $\perp$. Otherwise, go to next step:

iv. Compute $K_\ell \leftarrow C_{\ell,2}/e(S_{id_i}, C_{\ell,1})$.

v. For $i = \ell - 2$ down to 1, compute $K_i \leftarrow C_{i+1,2}/e(\mathrm{H}_3(K_{i+1}), C_{i+1,1})$.

vi. Compute and output $M \leftarrow C_{1,2}/e(\mathrm{H}_3(K_1), C_{1,1})$.

Consistency. Let us consider the following two cases:

- For an original (the *first-level*) ciphertext $C_{id_i}^{(1)} = (C_{1,1}, C_{1,2}, C_{1,3})$, where $C_{1,1} = rP, C_{1,2} = M \cdot e(Q_{id_i}, Y_{id_i})^r, C_{1,3} = r\big(\mathrm{H}_2\big(C_{1,1} \parallel C_{1,2}\big)P_1 + P_2\big)$,it is easy to verify:

$$M = \frac{C_{1,2}}{e(S_{id_i}, C_{1,1})} = \frac{M \cdot e(Q_{id_i}, Y_{id_i})^r}{e(x_i s Q_{id_i}, rP)} = \frac{M \cdot e(Q_{id_i}, x_i sP)^r}{e(Q_{id_i}, x_i sP)^r} = M$$

- For an ℓ^{th}-*level* ($\ell > 1$) ciphertext $C_{id_\ell}^{(\ell)} = (C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$ we have

$C_{1,1} = rP, C_{1,2} = M \cdot e(Q_{id_1}, Y_{id_1})^r \cdot e(\mathrm{H}_3(K_1) + S_{id_1}, rP)$,

$C_{1,3} = r\big(\mathrm{H}_2\big(C_{1,1} \parallel M \cdot e(Q_{id_1}, Y_{id_1})^r\big)P_1 + P_2\big), \cdots,$

$C_{\ell,1} = R_1^{(\ell-1)} = r_{\ell-1}P, C_{\ell,2} = R_2^{(\ell-1)} = K_{\ell-1} \cdot e(Q_{id_\ell}, Y_{id_\ell})^{r_{\ell-1}}$,

$C_{\ell,3} = R_3^{(\ell-1)} = svk_{P_{\ell-1}}, C_{\ell,4} = R_4^{(\ell-1)} = r_{\ell-1}(\mathrm{H}_2\big(C_{\ell,3} \cdot C_{\ell,1} \parallel C_{\ell,2}\big)P_1 + P_2)$,

$C_{\ell,5} = S_{\ell-1} = \mathcal{S}(ssk_{P_{\ell-1}}, (C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,3}, C_{\ell,4}))$.

And re-encryption key $R^{(\ell-1)} = (R_1^{(\ell-1)}, R_2^{(\ell-1)}, R_3^{(\ell-1)}, R_4^{(\ell-1)}, R_5^{(\ell-1)})$, where $R_5^{(\ell-1)} = \mathrm{H}_3(K_{\ell-1}) + S_{id_{\ell-1}}^{-1}$.

First we verify whetehr $e\big(P, C_{\ell,4}\big)$ equals $e(C_{\ell,1}, \mathrm{H}_2\big(C_{\ell,3} \cdot C_{\ell,1} \parallel C_{\ell,2}\big)P_1 + P_2)$. If it holds, then for $\forall k \in [2, \ell]$, we verify whether $\mathcal{V}\left(C_{k,3}, C_{k,5}, \big(C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4}\big)\right)$ equals 1. If all of these equations hold, we compute

$$K_{\ell-1} = \frac{C_{\ell,2}}{e(S_{id_\ell}, C_{\ell,1})} = \frac{K_{\ell-1} \cdot e(Q_{id_\ell}, Y_{id_\ell})^{r_{\ell-1}}}{e(x_\ell s Q_{id_\ell}, r_{\ell-1}P)} = \frac{K_{\ell-1} \cdot e(Q_{id_\ell}, x_\ell sP)^{r_{\ell-1}}}{e(Q_{id_\ell}, x_\ell sP)^{r_{\ell-1}}} = K_{\ell-1},$$

and for i from $\ell - 1$ down to 1, compute

$$K_i = \frac{C_{i+1,2}}{e\big(\mathrm{H}_3(K_{i+1}), C_{i+1,1}\big)} = \frac{K_i \cdot e\big(Q_{id_{i+1}}, Y_{id_{i+1}}\big)^{r_i} \cdot e\big(R_5^{i+1}, r_i P\big)}{e(\mathrm{H}_3(K_{i+1}), r_i P)}$$

$$= \frac{K_i \cdot e\big(x_{i+1} s Q_{id_{i+1}}, P\big)^{r_i} \cdot e\big(\mathrm{H}_3(K_{i+1}) + S_{id_{i+1}}^{-1}, r_i P\big)}{e(\mathrm{H}_3(K_{i+1}), r_i P)}$$

$$= \frac{K_i \cdot e(S_{id_{i+1}}, r_i P) \cdot e(\mathrm{H}_3(K_{i+1}), r_i P) \cdot e(S_{id_{i+1}}^{-1}, r_i P)}{e(\mathrm{H}_3(K_{i+1}), r_i P)} = K_i,$$

Next, we compute M as follows:

$$M = \frac{C_{1,2}}{e\big(\mathrm{H}_3(K_1), C_{1,1}\big)} = \frac{M \cdot e\big(Q_{id_1}, Y_{id_1}\big)^r \cdot e\big(\mathrm{H}_3(K_1) + S_{id_1}^{-1}, rP\big)}{e(\mathrm{H}_3(K_1), rP)}$$

$$= \frac{M \cdot e\big(Q_{id_1}, x_1 sP\big)^r \cdot e(\mathrm{H}_3(K_1), rP) \cdot e(S_{id_1}^{-1}, rP)}{e(\mathrm{H}_3(K_1), rP)} = M \cdot e(S_{id_1}, rP) \cdot e(S_{id_1}^{-1}, rP) = M$$

Theorem 1 Let hash function H_1 be a random oracle. If there exists a PPT adversary $\mathcal{A}$ ($\mathcal{A}$ is a Type I or a Type II adversary) that breaks our CL-PRE scheme described above in the sense of IND-CLPr-CCA2 with non-negligible probability, then, there exists another PPT algorithm $\mathcal{B}$ that can solve the BDH (if $\mathcal{B}$ acts as a Type I adversary) or DBDH (if $\mathcal{B}$ acts as a Type II adversary) problem with non-negligible probability.

5. Security Proofs

1.5 5.1 Public Key Encryption Scheme

We first define a public key encryption scheme **BasicPub**, which will be used as a tool in our security proof for our CL-PRE scheme.

BasicPub is specified by three algorithms: KeyGen, Enc and Dec.

- KeyGen:
  1. Run $\mathcal{IG}$ to generate $\langle \mathbb{G}_1, \mathbb{G}_2, e, q \rangle$ with the usual properties.
  2. Choose a generator $P \in \mathbb{G}_1$, and two random elements $P_1, P_2 \in \mathbb{G}_1 \backslash \{P\}$.
  3. Choose $t_1, t_2 \in_R \mathbb{Z}_q^*$, compute $Q = t_1 P$.
  4. Choose $s, x \in_R \mathbb{Z}_q^*$, and set $P_0 = sP$, $X = xP$, $Y = xsP$.
  5. Choose a cryptographic hash function $\mathrm{H}_2: \{0,1\}^* \to \mathbb{Z}_q^*$.

  The message space and ciphertext space for **BasicPub** are $\mathcal{M} = \mathbb{G}_2$ and $C = \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$. The system's public parameters are

  $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, e, q, t_2, P, P_0, P_1, P_2, X, Y, Q, \mathrm{H}_2)$' and the private key is $S = t_1$.
- Enc: To encrypt a message $M \in \mathcal{M}$, do the following performance:
  1. Verify that the equality $e(X, P_0) = e(Y, P)$ holds. If not, output $\perp$.
  2. Otherwise, choose a random element $r \in \mathbb{Z}_q^*$.
  3. Set the ciphertext as: $C = (C_1, C_2, C_3)$, where $C_1 = rP, C_2 = M \cdot e(Q, Y)^r, C_3 = r(\mathrm{H}_2(t_2 C_1 \| C_2)P_1 + P_2) = e(P, C_3)$.
- Dec: Let $C = (C_1, C_2, C_3) \in \mathcal{C}$ be the ciphertext. To decrypt the ciphertext C using private key S and K_{pub}, do the following performance:
  1. Verify whether $e(C_1, \mathrm{H}_2(t_2 C_1 \| C_2)P_1 + P_2) = e(P, C_3)$ holds. If not, return $\perp$.
  2. Otherwise, compute $C_2 / e(SY, C_1) = M$.

1.6 5.2 Adversaries for BasicPub

A Type I IND-CLPr-CCA2 adversary $\mathcal{A}_I$ for **BasicPub** is a slightly modified version of the usual IND-CLPr-CCA2 adversary, as defined in [29]: $\mathcal{A}_I$ may repeatedly request to replace the public key components $\langle X, Y \rangle$ by a valid pair $\langle X', Y' \rangle$ of its choice. Once a valid public key pair has been replaced, all the challenger's replies to $\mathcal{A}_I$'s decryption queries, as well as the result of encryption of M_d in the **CHOICE AND CHALLENGE** phase should be with respect to the current value of the public key. As usual, the adversary $\mathcal{A}_I$'s task is to output a guess d' for bit d and its advantage $\mathrm{Adv}(\mathcal{A})$ is defined to be $\Pr[d' = d] - \frac{1}{2}$.

1.7 5.3 Statements of Lemmas

Lemma 1 Let H_1 be a random oracle, and there exists an IND-CLPr-CCA2 Type I adversary $\mathcal{A}_I$ against our CL-PRE scheme with non-negligible advantage ε. Suppose $\mathcal{A}_I$ makes q_p partial private key extraction, q_{ex} private key extraction, q_{rk} re-encryption key extraction, q_{reen} re-encryption queries, and q_d decryption queries, then, there exists another PPT algorithm $\mathcal{B}$ which acts as a Type I IND-CLPr-CCA2 adversary against **BasicPub**. The algorithm $\mathcal{B}$ can solve the BDH hard problem, while $\mathcal{B}$'s advantage is at least $\varepsilon / \mathrm{e}(q_p + q_{ex} + q_{rk} + q_{reen} + 1)$.

Lemma 2 Let H_1 be a random oracle and there exists an IND-CLPr-CCA2 Type II adversary $\mathcal{A}_{II}$ against our CL-PRE scheme with non-negligible advantage ε. Suppose $\mathcal{A}_{II}$ makes q_{ex} private key extraction, q_{rk} re-encryption key extraction, q_{reen} re-encryption, and q_d decryption queries, then, there exists a PPT algorithm $\mathcal{B}$ which acts as a Type II adversary.$\mathcal{B}$ can solve the DBDHproblem with the advantage at least $\varepsilon/e(q_{ex} + q_{rk} + q_{reen} + q_d + 1)$.

1.8 5.4 Proofs of Lemmas

We mention that H_2 and H_3 are two one-way, collision-resistent cryptographic hash functions which can be instantiated with SHA-256, etc., so we do not need to simulate them.

Proof 1 Let $\mathcal{A}_I$ be a Type I IND-CLPr-CCA2 adversary against our CL-PRE scheme. We construct from $\mathcal{A}_I$ another PPT algorithm $\mathcal{B}$ that acts as a Type I IND-CLPr-CCA2 adversary against **BasicPub**. $\mathcal{B}$ can solve the BDH problem. We assume that $\mathcal{CH}$ is the challenger available to $\mathcal{B}$. The game begins by supplying $\mathcal{B}$ with a public key

$$K_{pub} = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, t_2, P, P_0, P_1, P_2, X, Y, Q, H_2 \rangle.$$

1. **SETUP.** $\mathcal{B}$ creates the scheme's system parameters $par = \langle \mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, P_1, P_2, \text{Sig}, H_1, H_2, H_3 \rangle$, and gives par to $\mathcal{A}_I$. Here, $\mathbb{G}_1, \mathbb{G}_2, e, q, P, P_0, P_1, P_2, H_2$ are taken from K_{pub}. And Sig, H_1 and H_3 are chosen by $\mathcal{B}$, where H_1 is a random oracle that will be controlled by $\mathcal{B}$.
2. **PHASE 1.** In this phase, $\mathcal{B}$ responds $\mathcal{A}_I$'s following queries as below:
   (a) **H_1 Oracle $\mathcal{O}_{H_1}$**: $\mathcal{B}$ maintains a list of tuples $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i \rangle$, denoted by H_1^{list},which is initially empty. $\mathcal{B}$ responds $\mathcal{A}_I$'s queries H_1 on input $id_i \in \{0,1\}^*$ as:
   - If id_i already appears in H_1^{list} in a tuple of $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i \rangle$, then $\mathcal{B}$ responds with $H_1(id_i) = Q_i \in \mathbb{G}_1^*$.
   - Otherwise, using the techniques of Coron [30], $\mathcal{B}$ flips a biased coin $\alpha_i \in \{0,1\}$ that $\alpha_i = 1$ with probability γ, i.e., $\Pr[\alpha_i = 1] = \gamma$.
     i. If $\alpha_i = 0$, $\mathcal{B}$ selects $b_i, x_i \leftarrow \mathbb{Z}_q^*$, then outputs $Q_i = H_1(id_i) = b_i Q$, and records the tuple $\langle id_i, \alpha_i, Q_i, b_i, x_i, x_i P, x_i P_0 \rangle$ in H_1^{list}.
     ii. If $\alpha_i = 1$, $\mathcal{B}$ selects $b_i, x_i \leftarrow \mathbb{Z}_q^*$, then outputs $Q_i = H_1(id_i) = b_i P$, and records the tuple $\langle id_i, \alpha_i, Q_i, b_i, x_i, x_i P, x_i P_0 \rangle$ in H_1^{list}.

   (b) **Partial Private Key $\mathcal{O}_{psk}$**: On input id_i, $\mathcal{B}$ looks up H_1^{list} to obtain the corresponding value of id_i. If $\alpha_i = 0$,$\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ responds $\mathcal{A}_I$ with $b_i P_0$.

   (c) **Private Key $\mathcal{O}_{sk}$**: On input id_i, if the public key of id_i has been replaced, $\mathcal{B}$ aborts the simulation. Otherwise, $\mathcal{B}$ looks up H_1^{list} to obtain the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ aborts the simulation. If $\alpha_i = 1$, $\mathcal{B}$ responds $\mathcal{A}_I$ with $x_i b_i P_0$.

   (d) **Request for Public Key $\mathcal{O}_{pk}$**: If the request is on id_i, then $\mathcal{B}$ responds $\mathcal{A}_I$ with $\langle X_i, Y_i \rangle$ by accessing H_1^{list}.

   (e) **Replace Public Key $\mathcal{O}_{rpk}$**: Suppose that the request is to replace the public key of id_i with value $\langle X_i', Y_i' \rangle$, $\mathcal{B}$ first checks if the equality $e(X_i', P_0) = e(Y_i', P)$ holds. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ replaces the current entries $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i \rangle$ in H_1^{list} with the new entries $\langle id_i, \alpha_i, Q_i, b_i, *, X_i', Y_i' \rangle$.

(f) **Rekey $\mathcal{O}_{rk}$**: On input (id_i, id_j), $(i \neq j)$, we can assume that the public key for id_i has not been replaced, then, $\mathcal{B}$ does the following performance:

i. $\mathcal{B}$ selects $r_i \in_R \mathbb{Z}_q^*$, $K_i \in_R \mathbb{G}_2$.

ii. $\mathcal{B}$ runs $(svk_{P_i}, ssk_{P_i}) \leftarrow \mathcal{G}(1^\kappa)$ to generate the signature key pair of proxy P_i, .

iii. $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i and id_j.

iv. if $\alpha_i = \alpha_j = 1$, $\mathcal{B}$ computes $R_1^{(i)} = r_i P, R_2^{(i)} = K_i e(b_j P, x_j P_0)^{r_i}, R_3^{(i)} = svk_{P_i}, R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)} R_1^{(i)} \parallel R_2^{(i)}\right) P_1 + P_2), R_5^{(i)} = \mathrm{H}_3(K_i) + (x_i b_i P_0)^{-1}$.

v. if $\alpha_i = 1$, and $\alpha_j = 0$, $\mathcal{B}$ computes $R_1^{(i)} = r_i P, R_2^{(i)} = K_i \cdot e(b_j Q, Y_j)^{r_i}, R_3^{(i)} = svk_{P_i}$, $R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)} R_1^{(i)} \parallel R_2^{(i)}\right) P_1 + P_2), R_5^{(i)} = \mathrm{H}_3(K_i) + (x_i b_i P_0)^{-1}$.

vi. if $\alpha_i = \alpha_j = 0$, $\mathcal{B}$ computes $R_1^{(i)} = r_i P, R_2^{(i)} = K_i \cdot e(b_j Q, Y_j)^{r_i}, R_3^{(i)} = svk_{P_i}, R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)} R_1^{(i)} \parallel R_2^{(i)}\right) P_1 + P_2)$, $R_5^{(i)} = \mathrm{H}_3(K_i) + (x_i' P_0)^{-1}$, where x_i' is selected randomly from $\mathbb{Z}_q^*$.

vii. If $\alpha_i = 0$, and $\alpha_j = 1$, $\mathcal{B}$ aborts the simulation.

viii. $\mathcal{B}$ responds $\mathcal{A}_I$ with $rk_{i\to j} = (R_1^{(i)}, R_2^{(i)}, R_3^{(i)}, R_4^{(i)}, R_5^{(i)})$.

(g) **Reencryption Oracle $\mathcal{O}_{reen}$**: On input (id_i, id_j, C_{id_i}), $\mathcal{B}$ first checks whether C_{id_i} is a *first-level* or an ℓ^{th}-level ciphertext. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ reports failure and aborts the simulation. Otherwise, $\mathcal{B}$ does the following performance:

- For a *first-level* ciphertext $C_{id_i}^{(1)}$, $\mathcal{B}$ dose:
  i. Parse C_{id_i} as $(C_{1,1}, C_{1,2}, C_{1,3})$.
  ii. Check if $e(P, C_{1,3}) = e(C_{1,1}, H_2(C_{1,1} \parallel C_{1,2})P_1 + P_2)$. If not, return $\perp$ to $\mathcal{A}_I$.
  iii. Otherwise, $\mathcal{B}$ obtains the value $rk_{i\to j}$ by executing a **Set-ReKey** query. Then, compute $C_{id_j}^{(2)} = \mathbf{ReEnc}(par, rk_{i\to j}, C_{id_i})$, and return $C_{id_j}^{(2)}$ to $\mathcal{A}_I$.
- For an ℓ^{th}-level ($\ell > 1$) ciphertext $C_{id_i}^{(\ell)}$, $\mathcal{B}$ dose::
  i. Parse C_{id_i} as $(C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$.
  ii. Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \mathrm{H}_2(C_{\ell,3} C_{\ell,1} \parallel C_{\ell,2})P_1 + P_2)$ .If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$.
  iii. For $\forall k \in [2, \ell]$, check $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right) = 1$. Whenever one of them fails, return $\perp$ to $\mathcal{A}_I$.
  iv. Otherwise, $\mathcal{B}$ obtains the value $rk_{i\to j}$ by executing a **Set-ReKey** query. Then, compute $C_{id_j}^{(\ell+1)} = \mathbf{ReEnc}(\mathrm{par}, rk_{i\to j}, C_{id_i}^{(\ell)})$, and return $C_{id_j}^{(\ell+1)}$ to $\mathcal{A}_I$.

(h) **Decryption Oracle $\mathcal{O}_{dec}$**: On input (id_i, C_{id_i}), $\mathcal{B}$ first checks whether C_{id_i} is a *first-level* or an ℓ^{th}-level ($\ell > 1$) ciphertext. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i as $\langle id_i, \alpha_i, Q_i, b_i, *, X_i, Y_i \rangle$.

- For a *first-level* ciphertext $C_{id_i}^{(1)}$:

i. Parse C_{id_i} as $(C_{1,1}, C_{1,2}, C_{1,3})$.

ii. Check if $e(P, C_{1,3}) = e(C_{1,1}, H_2(C_{1,1} \| C_{1,2})P_1 + P_2)$. If not, $\mathcal{B}$ returns $\perp$.

iii. Otherwise, if $\alpha_i = 0$, or if the public key of id_i has been replaced, $\mathcal{B}$ makes a request to $\mathcal{CH}$ to replace the public key components $\langle t_2, X, Y\rangle$ in K_{pub} with new value $\langle b_i^{-1}, X_i, Y_i\rangle$. Then, $\mathcal{B}$ relays the decryption query $(b_i C_{1,1}, C_{1,2}, b_i C_{1,3})$ to $\mathcal{CH}$. It is easy to see that the CL-PRE decryption of $(C_{1,1}, C_{1,2}, C_{1,3})$ under the (unknown) private key of id_i is equal to the **BasicPub** decryption of $(b_i C_{1,1}, C_{1,2}, b_i C_{1,3})$ under the (unknown) private key corresponding to K_{pub}. Hence, $\mathcal{CH}$'s response to $\mathcal{B}$'s request can be relayed to $\mathcal{A}_I$.

iv. If $\alpha_i = 1$, and the public key of id_i has not been replaced, then $\mathcal{B}$ computes $M = C_{1,2}/e(S_{id_i}, C_{1,1})$, where $S_{id_i} = x_i b_i P_0$ for id_i.

- For an ℓ^{th}-level ($\ell > 1$) ciphertext $C_{id_i}^{(\ell)}$, $\mathcal{B}$ dose:

i. Parse C_{id_i} as $(C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$.

ii. Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \mathrm{H}_2(C_{\ell,3} C_{\ell,1} \| C_{\ell,2})P_1 + P_2)$. If not, return $\perp$.

iii. For $\forall k \in [2, \ell]$, check if $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right)$ equals 1. Whenever one of them fails, return $\perp$ to $\mathcal{A}_I$.

iv. Otherwise, if $\alpha_i = 0$, or if the public key of id_i has been replaced, $\mathcal{B}$ makes a request to $\mathcal{CH}$ to replace the public key $\langle t_2, X, Y\rangle$ in K_{pub} with new value $\langle b_i^{-1} C_{\ell,3}, X_i, Y_i\rangle$.Then, $\mathcal{B}$ relays the decryption query $(b_i C_{\ell,1}, C_{\ell,2}, b_i C_{\ell,4})$ to $\mathcal{CH}$. It is easy to see that the CL-PRE decryption of $(C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4})$ under the (unknown) private key of id_i is equal to the **BasicPub** decryption of $(b_i C_{\ell,1}, C_{\ell,2}, b_i C_{\ell,4})$, under the (unknown) private key corresponding to K_{pub}. $\mathcal{CH}$'s response to $\mathcal{B}$'s request is the result of $K_{\ell-1}$.

v. If $\alpha_i = 1$, and the public key of id_i has not been replaced, then $\mathcal{B}$ computes $K_{\ell-1} = C_{\ell,2}/e(S_{id_i}, C_{\ell,1})$, where $S_{id_i} = x_i b_i P_0$ for id_i.

vi. For $j = \ell - 2$ down to 1, $\mathcal{B}$ computes $K_j = C_{j+1,2}/e(\mathrm{H}_3(K_{j+1}), C_{j+1,1})$.

vii. $\mathcal{B}$ computes $M = C_{1,2}/e(\mathrm{H}_3(K_1), C_{1,1})$, and returns M to $\mathcal{A}_I$.

3. **CHOICE AND CHALLENGE.** Once $\mathcal{A}_I$ decides that **PHASE 1** is over, it submits (id^*, M_0, M_1) on which it wishes to be challenged with the following restrictions:

(a) $\mathcal{A}_I$ has not previously issued a query $\mathcal{O}_{sk}(id^*)$.

(b) $\mathcal{A}_I$ has not previously issued a query $\mathcal{O}_{psk}(id^*)$ if he had already request to replace the public key for id^*.

(c) $\mathcal{A}_I$ has not previously requested to replace the public key for id^* if he had already issued a query $\mathcal{O}_{psk}(id^*)$.

(d) $\mathcal{A}_I$ has not previously issued any queries $\mathcal{O}_{rk}$ which will enable $\mathcal{A}_I$ to compute a re-encryption key from id^* to some entity for which $\mathcal{A}_I$ holds the private key.

(e) M_0 and $M_1 \in \mathcal{M}$ are of the same length.

Aassume that id^* has already been queried on H_1. Let $(id^*, \alpha^*, *, b^*, *, X^*, Y^*)$ be the corresponding values. $\mathcal{B}$ produces the challenge ciphertext as follows:

(a) $\mathcal{B}$ makes a request to its challenger $\mathcal{CH}$ to replace the public key components $\langle t_2, X, Y\rangle$ in K_{pub} with the new value $\langle (b^*)^{-1}, X^*, Y^*\rangle$.

(b) $\mathcal{B}$ gives $\mathcal{CH}$ the pair (M_0, M_1) as the messages on which it wishes to be challenged.

(c) $\mathcal{CH}$ chooses $d \in \{0,1\}$ randomly, and encrypts M_d under K_{pub}. Then $\mathcal{CH}$ responds $\mathcal{B}$ with the challenge ciphertext $C' = (C'_{1,1}, C'_{1,2}, C'_{1,3})$.

(d) $\mathcal{B}$ sets $C^*_{1,1} = (b^*)^{-1}C'_{1,1}$, $C^*_{1,2} = C'_{1,2}$, $C^*_{1,3} = (b^*)^{-1}C'_{1,3}$, and returns $C^* = (C^*_{1,1}, C^*_{1,2}, C^*_{1,3})$ to $\mathcal{A}_I$ as the challenge ciphertext.

4. **PHASE 2.** In this phase, $\mathcal{B}$ responds $\mathcal{A}_I$'s queries in the same way as those in **PHASE 1** with some restrictions described by the following dynamic directed graph $G = (V, E)$:

(a) Initialization. (1)Let $E = \emptyset$; and V contain all identities emerged in **PHASE 1**, $V = \{id_1, \cdots, id_n, id^*\}$. (2)Let V_C be the set of vertexes that the corresponding private keys has been given to $\mathcal{A}_I$.(3)For any $\mathcal{O}_{sk}(id_i, id_j)$query appeared in **PHASE 1**, add a directed edge (id_i, id_j) to E. (4)For each vertex $id_i \in V_C$ and each $id_j \in V$ with $id_i \neq id_j$, add a directed edge (id_i, id_j) into E. (See **Fig. 1(a)**.)

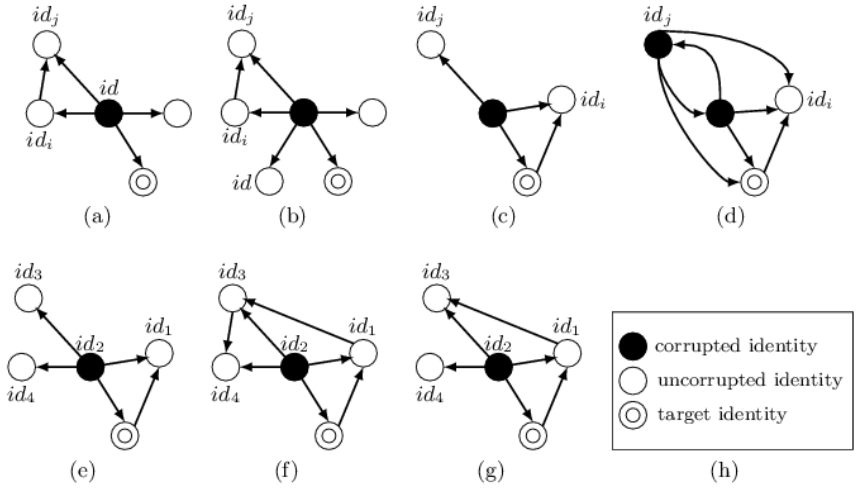

Fig. 1. Dynamic Directed Graph Model of Multi-use Game

(b) In the subsequent simulation, whenever a new identity id emerges, we also update V by setting $V \leftarrow V \cup \{id\}$, and then for each vertex $id_i \in V_C$, we add a directed edge (id_i, id) into E. (See **Fig. 1(b)**)

(c) On $\mathcal{A}_I$'s an $\mathcal{O}_{psk}(id^*)$ query: If $\mathcal{A}_I$ has requested to replace the public key of id^* in **PHASE 1**, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$. Otherwise, $\mathcal{B}$ responds $\mathcal{A}_I$ as in **PHASE 1**.

(d) On $\mathcal{A}_I$'s any query $\mathcal{O}_{sk}(id)$:

i. If there is a directed path from id^* to id in G, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$.

ii. Otherwise, $\mathcal{B}$ responds $\mathcal{A}_I$ as in **PHASE 1**. Then, $\mathcal{B}$ updates V_C by setting $V_C \leftarrow V_C \cup \{id\}$. Next, for each vertex $id_i \in V$, as long as $id \neq id_i$, $\mathcal{B}$ adds a directed edge (id, id_i) into E. (See **Fig. 1(c)** and **Fig. 1(d)**.).

(e) On $\mathcal{A}_I$'s any query of the form $\mathcal{O}_{rk}(id_i, id_j)$ with $id_i \neq id_j$:

i. If there is a directed path from id^* to id_i in G and $id_j \in V_C$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$.

ii. Otherwise, $\mathcal{B}$ computes $rk_{i\to j}$ as in **PHASE 1**, and returns it to $\mathcal{A}_I$. Next, $\mathcal{B}$ adds a directed edge (id_i, id_j) into E. (See **Fig. 1(e)**. and **Fig. 1(f)**.)

(f) On $\mathcal{A}_I$'s any query of the form $\boldsymbol{\mathcal{O}_{reen}}(id_i, id_j, C_{id_i})$ with $id_i \neq id_j$. Firstly, $\mathcal{B}$ checks:

(A) $id_j \in V_C$;

(B) There exists a directed path from id^* to id_i, denoted by path (id^*, id_i), such that

$$C_{id_i} = \text{ReencryptAlongPath}(\text{path}(id^*, id_i), C^*)$$

If both (A) and (B) hold, then $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$; Otherwise, $\mathcal{B}$ does:

i. At first, $\mathcal{B}$ derives a re-encryption key $rk_{i\to j}$ in the same way as in **PHASE 1**.

ii. Then $\mathcal{B}$ computes and returns the ciphertext C_{id_j} as in **PHASE 1** using $rk_{i\to j}$.

iii. If (B) holds, $\mathcal{B}$ adds a directed edge (id_i, id_j) into E. (See **Fig. 1(g)**. and **Fig. 1(f)**.)

(g) On $\mathcal{A}_I$'s any query of the form $\mathcal{O}_{dec}(id, C)$:

i. If $(id, C) = (id^*, C^*)$, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_I$.

ii. If there exists a directed path from id^* to id, denoted by path (id^*, id), such that

$$C = \text{ReencryptAlongPath}(\text{path}(id^*, id), C^*),$$

then $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ responds $\mathcal{A}_I$ as in **PHASE 1**.(See **Fig. 1(g)**.)

5. **GUESS.** Eventually, $\mathcal{A}_I$ outputs its guess bit d' for d. $\mathcal{B}$ outputs its guess as follows:

(a) If $\alpha^* = 1$, $\mathcal{B}$ aborts.

(b) If $\alpha^* = 0$, $\mathcal{B}$ outputs $\mathcal{A}_I$'s guess d' as its guess for **BasicPub**. $\mathcal{B}$ also computes $D = C^*_{1,2}/M'_d$, and outputs $(P, b^* C^*_{1,1}, Q, Y^*, D)$ as a BDH tuple.

[Analysis of probability that $\mathcal{B}$ does not abort.]

1. On an $\mathcal{O}_{psk}(id_i)$, $\mathcal{O}_{sk}(id_i)$, $\mathcal{O}_{rk}(id_i, id_j)$, or an $\mathcal{O}_{reen}(id_i, id_j, c_{id_i})$ query, if $\alpha_i = 1$, then $\mathcal{B}$ does not abort. Suppose that $\mathcal{A}_I$ makes q_p partial private key extraction, q_{ex} private key extraction, q_{rk} re-encryption key extraction, and q_{reen} re-encryption queries respctively, the probability that $\mathcal{B}$ does not abort is $\gamma^{q_p} + \gamma^{q_{ex}} + \gamma^{q_{rk}} + \gamma^{q_{reen}}$.
2. At the end of the **GUESS** phase, when $\mathcal{A}_I$ outputs its guess bit d', $\mathcal{B}$ does not abort if $\alpha_i = 0$. The probability that $\mathcal{B}$ does not abort in this case is $(1-\gamma)$.

Totally, the probability that $\mathcal{B}$ does not abort is $(1-\gamma)\gamma^{q_p+q_{ex}+q_{rk}+q_{reen}}$, which is maximized at $\gamma_{opt} = 1 - 1/(q_p + q_{ex} + q_{rk} + q_{reen} + 1)$. This means the probability that $\mathcal{B}$ does not abort is at least $1/\mathrm{e}(q_p + q_{ex} + q_{rk} + q_{reen} + 1)$, and $\mathcal{B}$'s advantage against **BasicPub** and in solving the BDH problem is at least $\varepsilon/\mathrm{e}(q_p + q_{ex} + q_{rk} + q_{reen} + 1)$.

Proof of Lemma 2:

Proof 2 Let $\mathcal{A}_{II}$ be a Type II IND-CLPr-CCA2 adversary against our CL-PRE. We construct from $\mathcal{A}_{II}$ another PPT algorithm $\mathcal{B}$ that can solve the DBDH hard problem.

Algorithm $\mathcal{B}$ accepts input of a properly distributed tuple $\langle \mathbb{G}_1 = \langle P \rangle, aP, bP, cP, T \rangle \in \mathbb{G}_1^4 \times \mathbb{G}_2$, and outputs 1 if $T = e(P,P)^{abc}$. Otherwise, output 0.

1. **SETUP.** $\mathcal{B}$ creates scheme's system parameters

$$\text{par} = (\mathbb{G}_1, \mathbb{G}_1, e, q, P, P_0, P_1, P_2, Sig, \mathrm{H}_1, \mathrm{H}_3, \mathrm{H}_3)$$

for $\mathcal{A}_{II}$ by selecting $s, t_1, t_2 \in \mathbb{Z}_q^*$ at random, and sets $P_0 = sP$, $P_1 = t_1 P$, $P_2 = t_2 P$. $\mathcal{B}$ then gives par and s to $\mathcal{A}_{II}$. Here, H_1 is a random oracle that will be controlled by $\mathcal{B}$.

2. **PHASE 1.** In this phase, $\mathcal{B}$ responds $\mathcal{A}_{II}$'s following queries:

$\mathbf{H_1}$ Oracle $\mathcal{O}_{\mathbf{H_1}}$: $\mathcal{B}$ maintains a list of tuples $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i\rangle$, denoted by $\mathrm{H}_1^{\mathrm{list}}$. $\mathrm{H}_1^{\mathrm{list}}$ is initially empty. $\mathcal{B}$ responds $\mathcal{A}_{II}$'s queries H_1 on input $id_i \in \{0,1\}^*$ as follows:

- If id_i already appears in $\mathrm{H}_1^{\mathrm{list}}$ in a tuple of $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i\rangle$, then $\mathcal{B}$ responds with $\mathrm{H}_1(id_i) = Q_i \in \mathbb{G}_1^*$.
- Otherwise, using the techniques of Coron [30], $\mathcal{B}$ flips a biased coin $\alpha_i \in \{0,1\}$ that $\alpha_i = 1$ with probability γ and 0 otherwise.
  (a) If $\alpha_i = 0$, $\mathcal{B}$ selects $b_i, x_i \in_R \mathbb{Z}_q^*$, then outputs $Q_i = \mathrm{H}_1(id_i) = b_i(bP)$, and records the tuple $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i = x_i(aP), Y_i = sx_i(aP)\rangle$ in the $\mathrm{H}_1^{\mathrm{list}}$.
  (b) If $\alpha_i = 1$, $\mathcal{B}$ selects $b_i, x_i \in_R \mathbb{Z}_q^*$, then outputs $Q_i = \mathrm{H}_1(id_i) = b_iP$, and records the tuple $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i = x_iP, Y_i = sx_iP\rangle$ in the $\mathrm{H}_1^{\mathrm{list}}$.

Partial Private Key $\mathcal{O}_{psk}$: On input id_i, $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ responds $\mathcal{A}_{II}$ with $sQ_i = sb_i(bP)$. Otherwise, if $\alpha_i = 1$, $\mathcal{B}$ responds $\mathcal{A}_{II}$ with $sQ_i = b_iP_0$.

Private Key $\mathcal{O}_{sk}$: On input id_i (now $\mathcal{A}_{II}$ is not allowed to replace the public key for id_i), $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ aborts the simulation. Otherwise, if $\alpha_i = 1$, $\mathcal{B}$ responds $\mathcal{A}_{II}$ with $x_ib_iP_0$.

Request for Public Key $\mathcal{O}_{pk}$: If the request is on id_i, then $\mathcal{B}$ responds $\mathcal{A}_{II}$ with $\langle X_i, Y_i\rangle$ by accessing $\mathrm{H}_1^{\mathrm{list}}$.

Rekey Oracle $\mathcal{O}_{rk}$: On input $(id_i, id_j), (i \neq j)$, $\mathcal{B}$ does the following:

(a) $\mathcal{B}$ selects $r_i \in_R \mathbb{Z}_q^*$, $K_i \in_R \mathbb{G}_2$.

(b) $\mathcal{B}$ runs $\mathcal{G}(1^\kappa)$ to generate the signature key pair of P_i, that is, $(svk_{P_i}, ssk_{P_i}) \leftarrow \mathcal{G}(1^\kappa)$.

(c) $\mathcal{B}$ looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i and id_j.

(d) If $\alpha_i = \alpha_j = 1$, $\mathcal{B}$ computes $R_1^{(i)} = r_iP$, $R_2^{(i)} = K_i \cdot e(b_jP, x_jP_0)^{r_i}, R_3^{(i)} = svk_{P_i}$, $R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)}R_1^{(i)} \parallel R_2^{(i)}\right)P_1 + P_2)$, $R_5^{(i)} = \mathrm{H}_3(K_i) + (x_ib_iP_0)^{-1}$.

(e) If $\alpha_i = 1$ and $\alpha_j = 0$, $\mathcal{B}$ computes $R_1^{(i)} = r_iP, R_2^{(i)} = K_i \cdot e(b_jbP, Y_j)^{r_i}, R_3^{(i)} = svk_{P_i}, R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)}R_1^{(i)} \parallel R_2^{(i)}\right)P_1 + P_2)$, $R_5^{(i)} = \mathrm{H}_3(K_i) + (x_ib_iP_0)^{-1}$.

(f) If $\alpha_i = \alpha_j = 0$, $\mathcal{B}$ computes $R_1^{(i)} = r_iP, R_2^{(i)} = K_i \cdot e(b_jbP, sx_jaP)^{r_i}, R_3^{(i)} = svk_{P_i}, R_4^{(i)} = r_i(\mathrm{H}_2\left(R_3^{(i)}R_1^{(i)} \parallel R_2^{(i)}\right)P_1 + P_2)$, $R_5^{(i)} = \mathrm{H}_3(K_i) + (x_i'P_0)^{-1}$, where x_i' is selected randomly from $\mathbb{Z}_q^*$.

(g) If $\alpha_i = 0$ and $\alpha_j = 1$, $\mathcal{B}$ aborts the simulation.

(h) $\mathcal{B}$ responds $\mathcal{A}_{II}$ with $rk_{i\to j} = (R_1^{(i)}, R_2^{(i)}, R_3^{(i)}, R_4^{(i)}, R_5^{(i)})$.

Reencryption $\mathcal{O}_{reen}$: On input $(id_i, id_j, C), (i \neq j)$, $\mathcal{B}$ first checks whether C is a *first-level* or an ℓ^{th}*-level* ciphertext. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$. Otherwise, $\mathcal{B}$looks up $\mathrm{H}_1^{\mathrm{list}}$ to obtain the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ reports failure and aborts the simulation. Otherwise:

- For a *first-level* ciphertext:
  (a) Parse C as $(C_{1,1}, C_{1,2}, C_{1,3})$.
  (b) Check if $e(P, C_{1,3}) = e(C_{1,1}, H_2(C_{1,1} \parallel C_{1,2})P_1 + P_2)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$.

(c) Otherwise, $\mathcal{B}$ obtains the value $rk_{i\to j}$ by executing a Set-ReKey query. Then, compute $C^{(2)} = \text{ReEnc}(\text{par}, rk_{i\to j}, C)$, and returns $C^{(2)}$ to $\mathcal{A}_{II}$.

- For an ℓ^{th}-level ciphertext ($\ell > 1$):
  (a) Parse C as $(C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$.
  (b) Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \text{H}_2(C_{\ell,3}C_{\ell,1} \parallel C_{\ell,2})P_1 + P_2)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$.
  (c) For $\forall k \in [2, \ell]$, $\mathcal{B}$ checks whether $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right)$ equals 1. Whenever one of them fails, return $\perp$ to $\mathcal{A}_{II}$.
  (d) Otherwise, $\mathcal{B}$ obtains the value $rk_{i\to j}$ by executing a Set-ReKey query. Then, compute $C^{(\ell+1)} = \text{ReEnc}(\text{par}, rk_{i\to j}, C)$, and returns $C^{(\ell+1)}$ to $\mathcal{A}_{II}$..

Decryption $\mathcal{O}_{dec}$: On input (id_i, C), $\mathcal{B}$ checks whether C is a *first-level* or an ℓ^{th}-level ($\ell > 1$) ciphertex. If not, $\mathcal{B}$ returns $\perp$. Otherwise, $\mathcal{B}$ looks up H_1^{list} to obtain the corresponding value of id_i as $\langle id_i, \alpha_i, Q_i, b_i, x_i, X_i, Y_i \rangle$. If $\alpha_i = 0$, $\mathcal{B}$ reports failure and aborts. Otherwise,

- For a *first-level* ciphertext:
  (a) Parse C as $(C_{1,1}, C_{1,2}, C_{1,3})$.
  (b) Check if $e(P, C_{1,3}) = e(C_{1,1}, \text{H}_2(C_{1,1} \parallel C_{1,2})P_1 + P_2)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$.
  (c) Otherwise, $\mathcal{B}$ computes $M = C_{1,2}/e(S_{id_i}, C_{1,1})$, where $S_{id_i} = x_i b_i P_0$ for id_i.
- For an ℓ^{th}-level ($\ell > 1$) ciphertext:
  (a) Parse C as $(C_{1,1}, \cdots, C_{\ell,1}, C_{\ell,2}, C_{\ell,3}, C_{\ell,4}, S_{\ell-1})$.
  (b) Check if $e(P, C_{\ell,4}) = e(C_{\ell,1}, \text{H}_2(C_{\ell,3}C_{\ell,1} \parallel C_{\ell,2})P_1 + P_2)$. If not, $\mathcal{B}$ returns $\perp$ to $\mathcal{A}_{II}$.
  (c) For $\forall k \in [2, \ell]$, $\mathcal{B}$ verifies that $\mathcal{V}\left(C_{k,3}, S_{k-1}, (C_{1,1}, \cdots, C_{k,1}, C_{k,3}, C_{k,4})\right) = 1$. Whenever one of them fails, return $\perp$ to $\mathcal{A}_{II}$.
  (d) Otherwise, $\mathcal{B}$ computes $K_{\ell-1} = C_{\ell,2}/e(S_{id_i}, C_{\ell,1})$, where $S_{id_i} = x_i b_i P_0$ for id_i.
  (e) For $j = \ell - 2$ down to 1, $\mathcal{B}$ computes $K_j = C_{j+1,2}/e(\text{H}_3(K_{j+1}), C_{j+1,1})$.
  (f) $\mathcal{B}$ computes and returns $M = C_{1,2}/e(\text{H}_3(K_1), C_{1,1})$ to $\mathcal{A}_{II}$.

3. **CHOICE AND CHALLENGE.** Once $\mathcal{A}_{II}$ decides that **PHASE 1** is over, it submits (id^*, M_0, M_1) on which it wishes to be challenged. The restrictions here are:
   (a) $\mathcal{A}_{II}$ has not previously issued a $\mathcal{O}_{sk}(id^*)$ query.
   (b) $\mathcal{A}_{II}$ has not previously issued any $\mathcal{O}_{rk}$ queries which will enable $\mathcal{A}_{II}$ to compute a re-encryption key from id^* to some entity for which $\mathcal{A}_{II}$ holds the private key.
   (c) M_0 and $M_1 \in \mathcal{M}$ are of the same length.

   $\mathcal{B}$ produces the challenge ciphertext as follows:
   (a) $\mathcal{B}$ looks up H_1^{list} to obtain the corresponding value of id^* as $\langle id^*, \alpha^*, Q^*, b^*, X^*, Y^* \rangle$.
   (b) $\mathcal{B}$ chooses $d \in \{0,1\}$ at random.
   (c) $\mathcal{B}$ computes and returns $C^* = (C_1^*, C_2^*, C_3^*)$ to $\mathcal{A}_{II}$ as the target ciphertext., where $C_1^* = cP$, $C_2^* = M_d \cdot T^{sx^*b^*}$, $C_3^* = \text{H}_2(C_1^* \parallel C_2^*)t_1(cP) + t_2(cP)$.
4. **PHASE 2.** In this phase, $\mathcal{B}$ responds $\mathcal{A}_{II}$'s queries as in **PHASE 1** with some restrictions. We also use a dynamic directed graph $G = (V, E)$ to describe the restrictions. G is formed almost in the same way as in proof of Lemma 1 except that:

(a) On $\mathcal{A}_{II}$'s an $\mathcal{O}_{psk}(id^*)$ query: Since $\mathcal{A}_{II}$ is a Type II adversary, it is not allowed to issue requests to replace public keys, $\mathcal{B}$ should always responds $\mathcal{A}_{II}$'s this query as in **PHASE 1**.

(b) On $\mathcal{A}_{II}$'s any query of the form $\mathcal{O}_{dec}(id_i, C)$, let α_i represent the corresponding value of id_i. If $\alpha_i = 0$, $\mathcal{B}$ reports failure and aborts the simulation. Otherwise, $\mathcal{B}$ responds $\mathcal{A}_{II}$ as described in the proof of Lemma 1.

5. **GUESS.** Eventually, $\mathcal{A}_{II}$ outputs its guess bit d' for d. If $\alpha^* = 1$, $\mathcal{B}$ aborts and outputs a random bit. Otherwise, if $d' = d$, $\mathcal{B}$ outputs 1, or 0 otherwise.

[Analysis]

Similar to proof of Lemma1, $\mathcal{B}$ advantage solving the DBDH problem is at least is at least $\varepsilon/e(q_{ex} + q_{rk} + q_{reen} + q_d + 1)$.

6. Performance Analysis

In this section, we evaluate the properties and performance of our CL-PRE scheme. We compare our CL-PRE with the existing CL-PRE [21-23]. The comparison includes computation costs, the ciphertext size and the computational assumption.

It is unreasonably to compare the computation cost and ciphertext size between a CPA secure scheme and a CCA secure scheme, so in the following, we only compare our scheme with [21].

We compare the computation costs of these two schemes in **Table 2**. Let P, E, T, H denote the operation of bilinear pairings, exponentiations, the multiplication operation, and the hash operation, respectively. From **Table 2**, we can see the difference of these two schemes is little.

Table 2. Computational Cost Comparsion between PRE schemes

	SJPR([21])	Our CL-PRE
PartialKey	1E+1T+1H	1E+1H
PublicKey	2E	2E
Enc	7E+1T+2H	1P+3E+1T+1H
ReKeyGen	5E+2T+3H	1P+4E+3T+3H
ReEnc	6P+1E+1T+2H	5P+2E+2T+2H
Dec	2P+4E+1T+3H	3P+1E+2T+1H

Table 3. Ciphertext Size and Assmption Comparsion between PRE schemes

	SJPR([21])	Our CL-PRE
Enc	$3\lvert\mathbb{G}_1\rvert + k$	$2\lvert\mathbb{G}_1\rvert + \lvert\mathbb{G}_2\rvert$
ReEnc	$3\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{G}_2\rvert + k + \lvert ID\rvert$	$4\lvert\mathbb{G}_1\rvert + 2\lvert\mathbb{G}_2\rvert + k + \lvert S\rvert$
Assumption	p-BDHI	BDH and DBDH

In **Table 3**, we compare the ciphertext size and the computational assumption. Let $|\mathbb{G}_1|$ and $|\mathbb{G}_2|$ be the length of the two bilinear groups, k the bit of the prime number q, q the order of the bilinear map groups $\mathbb{G}_1$ and $\mathbb{G}_2$, and $|S|$ denotes the length of the strong one-time signature in our CL-PRE scheme. From this table, we can see that the ciphertext size in our

scheme is a little shorter than that in [21]. In addition, we notice that the public parameter in [21] is longer than ours. Moreover, [21] includes five hash functions, while ours only needs three.

The computational assumption in [21] (p-BDHI) is less standard than [22-23] (DBDH in [22-23]) and ours (BDH and DBDH in ours).

We stress that the CL-PRE scheme in [21] includes the delegator's public key. Usually, a good PRE scheme is suggested not to demonstrate the delegator's information.

7. Conclusion and Future Works

In this paper, we construct the certificateless proxy re-encryption scheme with several practical and useful properties: (1) Unidirectionality; (2) Multi-use; (3) Non-interactivity; and (4)Non-transitivity. Our CL-PRE scheme solves the certificate management problem in PRE cryptosystems, as well as the key escrow problem in ID-PRE cryptosystems. This makes our CL-PRE more practical in real world. We prove that our CL-PRE scheme is IND-CCA2 secure under the BDH and DBDH assumptions in the random oracle model. However, we find that the ciphertext expansion remains a puzzle to be settled. Next, we are trying to find a more efficient CL-PRE scheme with less ciphertext expansion and more useful properties.

References

[1] Matt Blaze, Gerrit Bleumer, Martin Strauss, "Divertible protocols and atomic proxy cryptography," in *Proc. of EUROCRYPT 1998*, pp. 127-144, May 31- June 4, 1998. Article (CrossRef Link)

[2] Taher El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469-472, 1985. Article (CrossRef Link)

[3] Giuseppe Ateniese, Kevin Fu, Matthew Green, Susan Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," in *Proc. of NDSS 2005*, February 3-4, 2005. Article (CrossRef Link)

[4] Giuseppe Ateniese, Kevin Fu, Matthew Green, Susan Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," *ACM Transaction on Information and System Security*, vol. 9, no. 1, pp. 1-30, 2006. Article (CrossRef Link)

[5] Ran Canetti, Susan Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. of ACM Conference on Computer & Communication Security 2007*, pp. 185-194, October 29-November 2, 2007. Article (CrossRef Link)

[6] Giuseppe Ateniese, Karyn Benson and Susan Hohenberger, "Key-Private Proxy Re-encryption," in *Proc. of CT-RSA 2009*, pp. 279-294, April 20-24, 2009. Article (CrossRef Link)

[7] Beno î t Libert, Damien Vergnaud, "Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption," *IEEE Transactions on Information Theory*, vol. 57, no. 3, pp. 1786-1802, 2008. Article (CrossRef Link)

[8] Peter Gutmann, "PKI: It's Not Dead, Just Resting," *IEEE Computer*, vol. 35, no. 8, pp. 41-49, 2002. Article (CrossRef Link)

[9] Adi Shamir, "Identity-Based Cryptosystems and Signature Schemes," in *Proc. of CRYPTO 1984*, pp. 47-53, August 19-22, 1984. Article (CrossRef Link)

[10] Matthew Green, Giuseppe Ateniese, "Identity-Based Proxy Re-encryption," in *Proc. of ACNS 2007*, pp. 288-306, June 5-8, 2007. Article (CrossRef Link)

[11] S. S. Al-Riyami, K. G. Paterson, "Certificateless Public Key Cryptography," in *Proc. of ASIACRYPT 2003*, pp. 452-473, November 30 to December 4, 2003. Article (CrossRef Link)

[12] Dae Hyun Yum, Pil Joong Lee, "Generic Construction of Certificateless Encryption," in *Proc. of ICCSA(1) 2004*, pp. 802-811, May 14-17, 2004. Article (CrossRef Link)

[13] Joonsang Baek, Reihaneh Safavi-Naini, Willy Susilo, "Certificateless Public Key Encryption Without Pairing," in *Proc. of ACM Trans. Inf. Syst. Secur.*, pp. 134-148, September 20-23, 2005. Article (CrossRef Link)
[14] Benoît Libert, Jean-Jacques Quisquate, "On Constructing Certificateless Cryptosystems from Identity Based Encryption," in *Proc. of PKC 2006*, pp. 474-490, April 24-26, 2006. Article (CrossRef Link)
[15] Jong Hwan Park, Kyu Young Choi, Jung Yeon Hwang, Dong Hoon Lee, "Certificateless Public Key Encryption in the Selective-ID Security Model (Without Random Oracles)," in *Proc. of Pairing 2007*, 4575, pp. 60-82, July 2-4, 2007. Article (CrossRef Link)
[16] Yinxia Sun, Futai Zhang, Joonsang Baek, "Strongly Secure Certificateless Public Key Encryption Without Pairing," in *Proc. of CANS 2007*, pp. 194-208, December 8-10, 2007. Article (CrossRef Link)
[17] Alexander W. Dent, Benoît Libert, Kenneth G. Paterson, "Certificateless Encryption Schemes Strongly Secure in the Standard Model," in *Proc. of PKC 2008*, pp. 344-359, March 9-12, 2008. Article (CrossRef Link)
[18] Hua Guo, Xiyong Zhang, Yi Mu, Zhoujun Li, "An Efficient Certificateless Encryption Scheme in the Standard Model," in *Proc. of NSS 2009*, pp. 302-309, October 19-21, 2009. Article (CrossRef Link)
[19] Ario Fiore, Rosario Gennaro, Nigel P. Smart, "Constructing Certificateless Encryption and ID-Based Encryption from ID-Based Key Agreement," in *Proc. of Pairing 2010*, pp. 167-186, December 13-15, 2010. Article (CrossRef Link)
[20] S. Sree Vivek, S. Sharmila Deva Selvi, C. Pandu Rangan, "CCA Secure Certificateless Encryption Schemes based on RSA," in *Proc. of SECRYPT 2011*, pp. 208-217, July 18-21, 2011. Article (CrossRef Link)
[21] Chul Sur, Chae Duk Jung, Youngho Park, Kyung Hyune Rhee, "Chosen-Ciphertext Secure Certificateless Proxy Re-Encryption," in *Proc. of CMS 2010*, pp. 214-232, May 31 - June 2, 2010. Article (CrossRef Link)
[22] Xiaoxin Wu, Lei Xu, Xinwen Zhang, "Poster: a certificateless proxy re-encryption scheme for cloud-based data sharing," in *Proc. of CCS 2011*, pp. 869-872, October 17-21, 2011. Article (CrossRef Link)
[23] Lei Xu, Xiaoxin Wu, Xinwen Zhang, "CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud," in *Proc. of ASIACCS 2012*, pp. 87-88, May 2-4, 2012. Article (CrossRef Link)
[24] Kaitai Liang, Joseph K. Liu, Duncan S. Wong, Willy Susilo, "An Efficient Cloud-Based Revocable Identity-Based Proxy Re-encryption Scheme for Public Clouds Data Sharing," in *Proc. of ESORICS 2014*, pp. 257-272, September 7-11, 2014. Article (CrossRef Link)
[25] Peng Xu, Jun Xu, Wei Wang, Hai Jin, Willy Susilo, Deqing Zou, "Generally Hybrid Proxy Re-Encryption: {A} Secure Data Sharing among Cryptographic Clouds," in *Proc. of AsiaCCS 2016*, pp. 913-918, May 30-June 3, 2016. Article (CrossRef Link)
[26] Yang Yang, Maode Ma, "Conjunctive Keyword Search With Designated Tester and Timing Enabled Proxy Re-Encryption Function for E-Health Clouds," *IEEE transaction on Information Forensics and Security*, vol. 11, no. 4, pp. 746-759, 2016. Article (CrossRef Link)
[27] Ran Canetti, Susan Hohenberger, "Chosen-ciphertext secure proxy re-encryption," in *Proc. of ACM Conference on Computer & Communication Security 2007*, pp. 185-194, October 29-November 2, 2007. Article (CrossRef Link)
[28] Jee Hea An, Yevgeniy Dodis, Tal Rabin, "On the Security of Joint Signature and Encryption," in *Proc. of EUROCRYPT'02*, pp. 83-107, April 28- May 2, 2002. Article (CrossRef Link)
[29] Mihir Bellare, Anand Desai, David Pointcheval, Phillip Rogaway, "Relations Among Notions of Security for Public-Key Encryption Schemes," in *Proc. of CRYPTO 1998*, pp. 26-45, August 23-27, 1998. Article (CrossRef Link)
[30] Jean-Sébastien Coron, "On the Exact Security of Full Domain Hash," in *Proc. of CRYPTO 2000*, pp. 229-235, August 20-24, 2000. Article (CrossRef Link)

Ya Liu is currently a lecturer in Department of Computer Science and Engineering, University of Shanghai for Science and Technology. She was awarded her Ph.D. degree from Shanghai Jiao Tong University in 2013. Her research interests include applied cryptography, network security, cloud computing, the design and analysis of symmetric ciphers and computational number theory.

Hongbing Wang received the Ph.D. degree in computer science and technology from Shanghai Jiao Tong University, China, in 2013. Her current research interests include applied cryptography, network security, cloud computing, and RFID security, etc.

Chunlu Wang is an associate professor in the School of Computer Science, Beijing University of Posts and Telecommunications. Her current research is in computer networks, information security and intelligent transportation.