A No-Pairing Proxy Re-Encryption Scheme for Data Sharing in Untrusted Cloud

Xin Qian[1,2(✉)], Zhen Yang[1,2], Shihui Wang[3], and Yongfeng Huang[1,2]

[1] Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
qianxl6@mails.tsinghua.edu.cn
[2] Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China
[3] School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China

Abstract. To protect data sharing from leakage in untrusted cloud, proxy re-encryption is commonly exploited for access control. However, most proposed schemes require bilinear pair to attain secure data sharing, which gives too heavy burden on the data owner. In this paper, we propose a novel lightweight proxy re-encryption scheme without pairing. Our scheme builds the pre-encryption algorithm based on computational Diffie–Hellman problem, and designs a certificateless protocol based on a semi-trusted Key Generation Center. Theoretical analysis proves that our scheme is Chosen-Ciphertext-Attack secure. The performance is also shown to achieve less computation burden for the data owner compared with the state-of-the-art.

Keywords: Proxy re-encryption · No pairing · Certificateless · Cloud data sharing

1 Introduction

Booming cloud computing has been developing very fast for many years, which brings convenience for billions of users all over the world. However, widely used cloud service has also caught the attention of attackers which have conducted many cloud security accidents, especially cloud data leakage. Accumulating data leakage in cloud environment makes cloud service untrusted for users.

To protect cloud data from leakage, encryption access control mechanisms has been introduced for secure data sharing [16], including proxy re-encryption.

The notion of proxy re-encryption (PRE) was first introduced by Blaze et al. [1] in 1998, and they promoted the first bidirectional PRE scheme based on a simple modification of the El Gamal encryption scheme [13]. Proxy re-encryption (PRE) is a cryptographic scheme which efficiently solve the problem of delegation of decryption

Supported by the National Key Research and Development Program of China (No. 2016YFB0801301, No. 2016YFB0800402); the National Natural Science Foundation of China (No. U1405254, No. U1536207).

X. Sun et al. (Eds.): ICAIS 2019, LNCS 11632, pp. 85–96, 2019.
https://doi.org/10.1007/978-3-030-24274-9_8

rights. It allows a semi-trusted proxy with re-encryption keys, generated by the delegator, to transform a ciphertext under a given public key of the delegator into a ciphertext of the same message under a different public key the delegatee, while learning nothing about the encrypted message. PRE schemes have many practical applications due to its transformation property, such as cloud storage [2], distributed file systems, encrypted email forwarding [1], outsourced filtering of encrypted spam and so on.

Most of the PRE schemes in practical applications are constructed based on either traditional Public Key Infrastructure (PKI) or Identity-Based Encryption (IBE) setting. In the PKI setting, the authenticity of a user's public key is assured by Digital Certificates, which is digitally signed and issued by a trusted third party called the Certification Authority (CA). It is the management of certificates, which is a costly and a cumbersome process including the revocation, storage, distribution of certificates and so on, that inherently makes PRE schemes based on PKI setting inefficient. In the IBE setting, the secret keys of all the users are generated by a third party called the Private Key Generator (PKG), therefore brings the key-escrow problem.

To solve both certificate management problem in the PKI setting and key-escrow problem in the IBE setting, certificateless public key encryption (CLPKE) was first introduced by Al-Riyami and Paterson [4] in 2003. CLPKE combines the advantages of PKI and of IBE, while does not suffer from the aforementioned problems, therefore indicates a new direction for the construction of the PRE schemes. The notion of certificateless proxy re-encryption (CLPRE) was introduced by Sur et al. [5] in 2010.

We propose a lightweight proxy certificateless proxy re-encryption(CLPRE) scheme for data sharing in untrusted cloud. Our scheme builds the pre-encryption algorithm based on computational Diffie–Hellman problem, and designs a certificateless protocol based on a semi-trusted Key Generation Center. Compared with the proxy re-encryption schemes state-of-the-art, our CLPRE scheme can be more lightweight for data owner and achieve same security against Chosen Ciphertext Attack (CCA).

2 Related Work

Since the first bidirectional PRE scheme was promoted by Blaze et al. [1] in 1998, there have been lots of works and promotions about PRE scheme. Ateniese et al. [3] proposed a unidirectional PRE schemes in 2005, which is the first unidirectional PRE scheme based on bilinear pairings and achieves CPA-secure (chosen-plaintext attack, CPA).

Canetti et al. [6] and Libert et al. [7] proposed the first bidirectional multi-hop and the first unidirectional single hop scheme both achieve RCCA-secure (replayable chosen ciphertext attack) in the standard model. For greater security, the construction of CCA-secure (chosen-ciphertext attack, CCA) PRE scheme has become an significant issue. In 2008, Deng et al. [8] proposed a bidirectional CCA secure PRE scheme without the bilinear pairings. Later, Hanaoka et al. [9] proposed a generic construction of CCA secure PRE scheme in the standard model.

All the PRE schemes are constructed based on either traditional Public Key Infrastructure (PKI) or Identity-Based Encryption (IBE) setting. In order to avoid both certificate management problem in the PKI setting and key-escrow problem in the IBE

setting, certificateless public key encryption (CLPKE) first introduced by Al-Riyami and Paterson [4] in 2003 is considered in the construction of the PRE schemes. In 2010, Sur et al. [5] first introduced the notion of CLPRE and proposed a CCA secure CLPRE scheme in the random oracle model, which was shown to be vulnerable to chosen ciphertext attack by Zheng et al. [10]. In 2013, a CLPRE scheme using bilinear pairings proposed by Guo et al. [11] satisfies RCCA-security in the random oracle model. A lot of works have been proposed during these years [17]. Unfortunately, construction of CLPRE schemes has so far depended on the costly bilinear pairings.

In 2005, Baek et al. [12] proposed an efficient certificateless public key encryption (CLPKE) scheme that does not rely on the bilinear parings. In 2014, Yang et al. [14] addressed a CLPRE scheme without bilinear pairing, which claimed to be CCA-secure in the random oracle model, yet was shown to be vulnerable to chain collusion attack in by Srinivasan et al. [15] in their work proposed in 2015. In that work, they proposed the first CCA-secure unidirectional certificateless PRE scheme without bilinear pairing under the Computational Diffie-Hellman (CDH) assumption in the random oracle model.

3 Model

In order to describe our lightweight proxy re-encryption scheme for data sharing in untrusted cloud more figurative and in details, Fig. 1 gives an overview of entities and their activities in this framework.

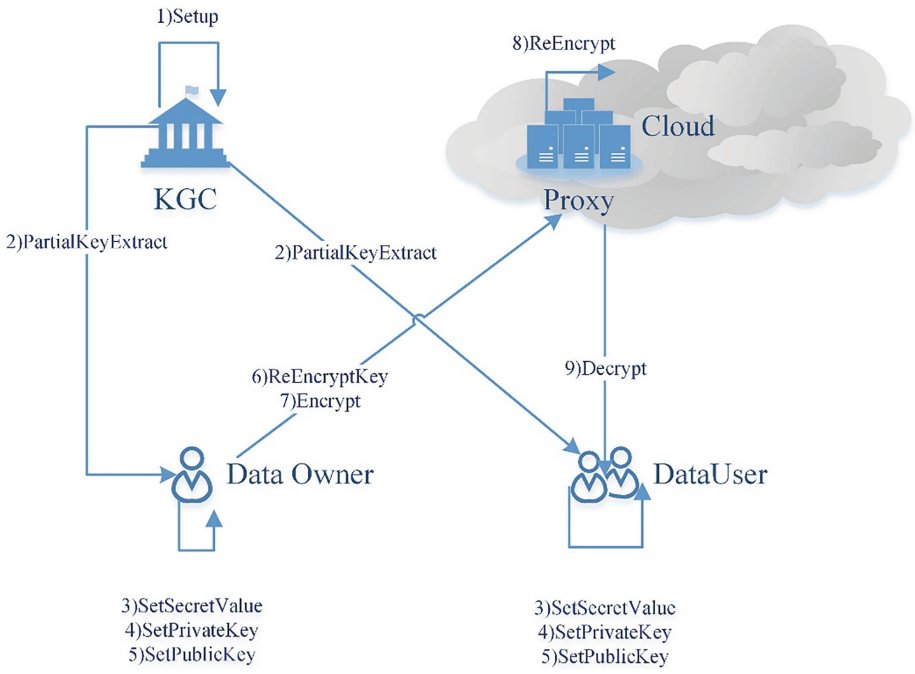

Fig. 1. Model architecture of our lightweight proxy re-encryption scheme.

Data Owner: Owner encrypts his data and uploads the ciphertext to the Cloud. Owner decides which users are authorized to share data from cloud by generating re-encryption keys for those authorized to transform the ciphertext.

Data User: Authorized users can share data from cloud by download the transformed ciphertext from cloud and decrypt by their own.

Proxy: Proxy is a semi-trusted third party in untrusted cloud. It re-encrypt the ciphertext in cloud under the private key of data owner into a ciphertext of the same message under given private keys of authorized users, while learning nothing about the encrypted data.

KGC: Key Generation Center is a semi-trusted fourth party, which means it is honest but curious. KGC initialize the whole system, and generate partial keys for everyone in the system. Users can generate his own public key and private key from partial keys generated by KGC and secret value chosen by himself. At the same time, KGC does not have any information about the secret value generated by the user and hence cannot decrypt any ciphertext.

4 CLPRE Scheme

The detailed construction of our CLPRE scheme are as follows. User i and user j represent the data owner and the authorized user respectively.

(1) **Setup** (1^{κ})**:** In the setup phase, on input a security parameter 1^{κ}, the Probabilistic Polynomial Time (PPT) algorithm run by the Key Generation Center (KGC) outputs the public parameters *params* and master secret key *msk*. The algorithm works as below:
Generate a k-bit prime q and a group $\mathbb{G}$ of order q. Pick a random generator $g \in \mathbb{G}$, and then randomly pick $s \in \mathbb{Z}_q^*$, compute $h = g^s$;
Choose cryptographic hash functions $H_1 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3 : \mathbb{G} \rightarrow \{0,1\}^{l+l_0}$, $H_4 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$, $H_5 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$.
The public parameters are $params = \{q, l, l_0, H_1, H_2, H_3, H_4, H_5\}$, where l, l_0 denote the bit-length of a message and a randomness respectively, and master secret key is $msk = s$, which is stored secretly. The message space is $\mathcal{M} = \{0,1\}^l$.

(2) **PartialKeyExtract** $(params, msk, ID_i)$**:** On input public parameters *params*, master secret key *msk* and the user i's identity ID_i, this algorithm run by KGC outputs the partial public key ppk_i and the partial secret key psk_i.
Pick a random $\alpha_i \in \mathbb{Z}_q^*$, and compute $a_i = g^{\alpha_i}$, $x_i = \alpha_i + sH_1(ID_i, a_i)$, return $(ppk_i, psk_i) = (a_i, x_i)$.

(3) **SetSecretValue** $(params, ID_i)$**:** On input public parameters *params*, and user i's identity ID_i, this algorithm run by user i outputs user i's secret value v_i.
Pick a random $z_i \in \mathbb{Z}_q^*$, return $v_i = z_i$.

(4) **SetPrivateKey** $(params, psk_i, v_i)$**:** On input public parameters *params*, user i's partial secret key psk_i, user i's secret value v_i, this algorithm run by user i outputs user i's secret key sk_i.
Return $sk_i = (x_i, z_i)$.

(5) **SetPublicKey** $(params, ppk_i, v_i)$**:** On input public parameters *params*, user i's partial public key ppk_i, user i's secret value v_i, this algorithm run by user i outputs user i's public key pk_i.
Compute $u_i = g^{z_i}$, return $pk_i = (a_i, u_i)$.

(6) **ReEncryptKey** $\left(params, ID_i, (pk_i, sk_i), ID_j, pk_j\right)$**:** On input public parameters *params*, user i's identity ID_i, user i's cryptographic key pair (pk_i, sk_i), user j's identity ID_j and public key pk_j, this algorithm run by user i outputs a re-encryption key $rk_{i\to j}$ from user i to user j.

Parse pk_i as (a_i, u_i), sk_i as (x_i, z_i) and pk_j as $\left(a_j, u_j\right)$, compute $V_j = a_j h^{H_1\left(ID_j, a_j\right)}$, and $W_{ij} = H_5\left(t_j^{z_i} \parallel u_j^{x_i} \parallel ID_i \parallel pk_i \parallel ID_j \parallel pk_j\right)$. Then compute $B_i = (x_i H_4(u_i) + z_i)$, return $rk_{i\to j} = B_i W_{ij}$.

(7) **Encrypt** $(params, m, pk_i)$**:** On input public parameters *params*, data $m \in \mathcal{M}\left(\mathcal{M} = \{0,1\}^l\right)$, and user i's public key pk_i, this algorithm run by user i outputs a second level ciphertext c_i.

Parse pk_i as (a_i, u_i), pick a randomness $\sigma \in \{0,1\}^{l_0}$, compute $V_i = a_i h^{H_1(ID_i, a_i)}$, $r = H_2(m \parallel \sigma \parallel ID_i \parallel u_i)$, then compute $c_1 = g^r$, $c_2 = (m \parallel \sigma) \oplus H_3\left(A_i^r\right)$, in which $A_i = V_i^{H_4(u_i)} \cdot u_i$. Return $c_i = (c_1, c_2)$.

(8) **ReEncrypt** $\left(params, c_i, rk_{i\to j}\right)$**:** On inputs public parameters *params*, a second level ciphertext c_i, a re-encryption key $rk_{i\to j}$, this algorithm run by proxy outputs a first level ciphertext c_j.

Parse $c_i = (c_1, c_2)$, compute $c_1' = c_1^{rk_{i\to j}}$, $c_2' = c_2$, in which $rk_{i\to j} = B_i W_{ij}$, $B_i = (x_i H_4(u_i) + z_i)$. Return $c_j = \left(c_1', c_2'\right)$.

(9) **Decrypt** $(params, pk_i, c_i)$**:** On input public parameters *params*, user i's public key pk_i, and a ciphertext c_i, this algorithm run by user i outputs either a plaintext $m \in \mathcal{M}$ or an error symbol $\perp$.

i. **Decrypt2** $(params, pk_i, c_i)$**:** The algorithm is run by the data owner user i.
Parse pk_i as (a_i, u_i), sk_i as (x_i, z_i), $c_i = (c_1, c_2)$, compute $(m \parallel \sigma) = c_2 \oplus H_3\left(c_1^{(x_i H_4(u_i) + z_i)}\right)$, then compute $r = H_2(m \parallel \sigma \parallel ID_i \parallel u_i)$.
Check if the equation holds: $g^r = c_1$.
Return m if it holds: $(m \parallel \sigma) = c_2 \oplus H_3\left(c_1^{(x_i H_4(u_i) + z_i)}\right)$.
Otherwise return $\perp$, which implies the ciphertext c_i is wrongful.

ii. **Decrypt1** $(params, pk_i, c_i)$**:** The algorithm is run by the authorized user j.
Parse pk_i as (a_i, u_i), sk_i as (x_i, z_i), $c_i = \left(c_1', c_2'\right)$, compute $V_j = a_j h^{H_1\left(ID_j, a_j\right)}$, $W_{ji} = H_5\left(V_j^{z_i} \parallel u_j^{x_i} \parallel ID_i \parallel pk_i \parallel ID_j \parallel pk_j\right)$ and $(m \parallel \sigma) = c_2' \oplus H_3\left(c_1'^{1/W_{ji}}\right)$, $r = H_2\left(m \parallel \sigma \parallel ID_j \parallel u_j\right)$.
Check if the equation holds: $\left(V_j^{H_4\left(u_j\right)} \cdot u_j\right)^{rW_{ji}} = c_1'$.
Return m if it holds: $(m \parallel \sigma) = c_2' \oplus H_3\left(c_1'^{1/W_{ji}}\right)$.
Otherwise return $\perp$, which implies the ciphertext c_i is wrongful.

5 Correctness

5.1 Correctness of PartialKey

As we can see from the **PartialKeyExtract** algorithm, $a_i = g^{\alpha_i}$, $x_i = \alpha_i + sH_1(ID_i, a_i)$, the equation $g^{psk_i} = g^{x_i} = g^{\alpha_i + sH_1(ID_i, a_i)} = a_i h^{H_1(ID_i, a_i)} = ppk_i h^{H_1(ID_i, ppk_i)}$ will hold, if the partial cryptographic key pair $(ppk_i, psk_i) = (a_i, x_i)$ is properly generated by KGC.

5.2 Correctness of Ciphertext2

As we can see from the **Encryt** algorithm, $c_1 = g^r$, $c_2 = (m \parallel \sigma) \oplus H_3\left(A_i^r\right)$, the equation $g^r = g^{H_2(m\|\sigma\|ID_i\|u_i)} = g^{H_2\left(c_2 \oplus H_3\left(c_1^{(x_i H_4(u_i) + z_i)}\right)\|ID_i\|u_i\right)} = c_1$ will hold, if $c_i = (c_1, c_2)$ is a correctly second level ciphertext generated by user i.

5.3 Correctness of Ciphertext1

As we can see from the **ReEncryt** algorithm, $\mathrm{c}_1^{'} = \mathrm{c}_1^{rk_{i\to j}} = (g^r)^{rk_{i\to j}}$, in which $rk_{i\to j} = B_i W_{ij}$, $\mathrm{c}_2^{'} = c_2 = (m \parallel \sigma) \oplus H_3\left(A_j^r\right)$, the equation $c_1^{'} = g^{rB_j W_{ji}} = A_j^{rW_{ji}} = \left(V_j^{H_4(u_j)} \cdot u_j\right)^{H_2\left(m\|\sigma\|ID_j\|u_i\right)W_{ji}} = \left(V_j^{H_4(u_j)} \cdot u_j\right)^{H_2\left(c_2^{'} \oplus H_3\left(c_1^{'1/W_{ji}}\right)\|ID_j\|u_i\right)W_{ji}}$ will hold, if $c_i = \left(\mathrm{c}_1^{'}, \mathrm{c}_2^{'}\right)$ is a correctly first level ciphertext generated by the proxy.

5.4 Correctness of Decrypt2

As mentioned above, $c_1 = g^r$, $c_2 = (m \parallel \sigma) \oplus H_3\left(A_i^r\right)$, hence $(m \parallel \sigma) = c_2 \oplus H_3\left(A_i^r\right) = c_2 \oplus H_3\left(g^{B_i r}\right) = c_2 \oplus H_3\left(c_1^{B_i}\right) = c_2 \oplus H_3\left(c_1^{(x_i H_4(u_i) + z_i)}\right)$, in which $A_i = V_i^{H_4(u_i)} \cdot u_i$, $B_i = (x_i H_4(u_i) + z_i)$.

5.5 Correctness of Decrypt1

As mentioned above, $\mathrm{c}_1^{'} = \mathrm{c}_1^{rk_{i\to j}}$, $c_2^{'} = c_2 = (m \parallel \sigma) \oplus H_3\left(A_j^r\right)$, hence $(m \parallel \sigma) = c_2^{'} \oplus H_3\left(A_j^r\right) = c_2^{'} \oplus H_3(g^{B_j r}) = c_2^{'} \oplus H_3\left(c_1^{B_j}\right) = c_2^{'} \oplus H_3\left(c_1^{'1/W_{ji}}\right)$.

5.6 Correctness of the Whole Scheme

For all $m \in \mathcal{M}$ and all users' cryptographic key pair (pk_i, sk_i), $\left(pk_j, sk_j\right)$, these algorithms should satisfy the following conditions of correctness:

(a) $\mathrm{Decrypt}_2(params, sk_i, \mathrm{Encrypt}(params, ID_i, pk_i, m)) = m$
(b) $\mathrm{Decrypt}_1\left(params, sk_j, \mathrm{ReEncrypt}\left(params, rk_{i\to j}, c_i\right)\right) = m$
where $c_i = \mathrm{Encrypt}(params, ID_i, pk_i, m)$, $rk_{i\to j} = \mathrm{ReEncryptKey}(params, ID_i, (pk_i, sk_i), ID_j, pk_j)$.

6 Security Proof

Two types of adversaries, Type I adversary $\mathcal{A}_{\mathbf{I}}$ and Type II adversary $\mathcal{A}_{\mathbf{II}}$ are considered for a CL-PRE. $\mathcal{A}_{\mathbf{I}}$ models an attacker from the outside (i.e. anyone except the KGC) without access to the master secret key msk but may replace public keys of entities (i.e. user i or user j) with values of its choice. $\mathcal{A}_{\mathbf{II}}$ models an honest-but-curious KGC who has access to the master secret key msk, but is not allowed to replace public keys of entities. In our model, Type I adversary $\mathcal{A}_{\mathbf{I}}$ is not considered since

Here we focus on the Type II adversary $\mathcal{A}_{\mathbf{II}}$ only and prove the chosen ciphertext security of first level ciphertext.

Definition (CLPRE-CCA Security). We say a CLPRE scheme is CLPRE-CCA secure if the scheme is 1st-IND-CLPRE-CCA secure and 2nd-IND-CLPRE-CCA secure.

Theorem 1 (1st-IND-CLPRE-CCA security). The proposed CLPRE scheme is 1st-IND-CLPRE-CCA secure against Type-II adversary $\mathcal{A}_{\mathbf{II}}$ in the random oracle model, if the CDH assumption holds in $\mathbb{G}$.

Lemma 1. Assume that H_1, H_2, H_3, H_4, H_5 are random oracles, if there exists a 1st-IND-CLPRE-CPA Type I adversary $\mathcal{A}_{\mathbf{II}}$ against the proposed CLPRE scheme with advantage ϵ when running in time t, making at most q_{pk} public key request queries, at most q_{pak} partial key extract queries, at most q_{sk} private key extract queries, at most q_{pkr} public key replacement queries, at most q_{rk} re-encryption key extract queries, at most q_{re} re-encryption queries, q_{dec_2} decrypt2 queries, q_{dec_1} decrypt1 queries, and q_{H_i} random oracle queries to $H_i (1 \le i \le 5)$. Then, for any $0 < v < \epsilon$, there exists an algorithm $\mathcal{C}$ to solve the (t', ϵ')-CDH problem in $\mathbb{G}$ with

$$\begin{aligned} t' \le t + (q_H + 2q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{pk} + q_{pak} + q_{sk} + q_{rk} + q_{re} \\ + q_{dec_2} + q_{dec_1})\,\mathcal{O}(1) + (2q_{pk} + q_{pak} + 2q_{sk} + 5q_{rk} + 6q_{re} \\ + 2q_{dec_2} + 5q_{dec_1})t_e \end{aligned}$$

$$\epsilon' \ge \frac{1}{q_{H_3}} \left(\frac{2(\epsilon - v)}{e\left(1 + q_{pak} + q_{rk}\right)} - \tau \right)$$

where e is the base of the natural logarithm, we denote the time taken for exponentiation operation in group $\mathbb{G}$ as t_e, and τ denotes the advantage that $\mathcal{A}_{\mathbf{II}}$ can distinguish the incorrectly-formed re-encryption keys in our simulation from all correctly-formed re-encryption keys in a "real world" interaction.

Theorem 2 (2nd-IND-CLPRE-CCA security). The proposed CLPRE scheme is 2nd-IND-CLPRE-CCA secure against Type-II adversary $\mathcal{A}_{\mathbf{II}}$ in the random oracle model, if the CDH assumption holds in $\mathbb{G}$.

Lemma 2. Assume that H_1, H_2, H_3, H_4, H_5 are random oracles, if there exists a 2nd-IND-CLPRE-CPA Type I adversary $\mathcal{A}_{\mathbf{II}}$ against the proposed CLPRE scheme with advantage ϵ when running in time t, making at most q_{pk} public key request queries, at most q_{pak} partial key extract queries, at most q_{sk} private key extract queries, at most q_{pkr} public key replacement queries, at most q_{rk} re-encryption key extract queries, at most q_{re} re-encryption queries, q_{dec_2} decrypt2 queries, q_{dec_1} decrypt1 queries, and q_{H_i} random oracle queries to $H_i (1 \le i \le 5)$. Then, for any $0 < v < \epsilon$, there exists an algorithm C to solve the (t', ϵ')-CDH problem in $\mathbb{G}$ with

$$\begin{aligned} t' \le t + (q_H + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{pk} + q_{pak} + q_{sk} + q_{rk} + q_{re} \\ + q_{dec_2} + q_{dec_1})\,\mathcal{O}(1) + (2q_{pk} + q_{pak} + 2q_{sk} + 5q_{rk} + 6q_{re} \\ + 2q_{dec_2} + 5q_{dec_1})t_e \end{aligned}$$

$$\epsilon' \ge \frac{1}{q_{H_3}} \left(\frac{2(\epsilon - v)}{e\left(1 + q_{pak} + q_{rk}\right)} - \tau \right)$$

where e is the base of the natural logarithm, we denote the time taken for exponentiation operation in group $\mathbb{G}$ as t_e, and τ denotes the advantage that $\mathcal{A}_{\mathbf{II}}$ can distinguish the incorrectly-formed re-encryption keys in our simulation from all correctly-formed re-encryption keys in a "real world" interaction.

Proof. We show how to construct an algorithm $\mathcal{C}$ which can solve the (t', ϵ')-CDH problem in group $\mathbb{G}$.

Suppose $\mathcal{C}$ is given a CDH challenge tuple $(g, g^a, g^b) \in \mathbb{G}^3$ with random unknown $a, b \in \mathbb{Z}_q^*$ as input. The goal of $\mathcal{C}$ is to compute the g^{ab}. $\mathcal{C}$ act as challenger and play the 2nd-IND-CLPRE-CPA "Game II" with adversary $\mathcal{A}_{\mathbf{II}}$ as follows.

"Game II": This is a game between $\mathcal{A}_{\mathbf{II}}$ and the challenger $\mathcal{C}$.

$\mathcal{A}_{\mathbf{II}}$ has access to the master secret key msk, but is not allowed to replace public keys of entities.

Setup. $\mathcal{C}$ takes a security parameter 1^κ, runs Setup(1^κ) algorithm to generate the system parameter params $= \{q, l, l_0, H_1, H_2, H_3, H_4, H_5\}$, and a master secret key msk $= s$. $\mathcal{C}$ gives params to $\mathcal{A}_{\mathbf{II}}$ while keeping msk secret.

Random Oracle Queries. H_1, H_2, H_3, H_4, H_5 are random oracles controlled by $\mathcal{C}$, who maintains six hash lists Hlist, H_1list, H_2list, H_3list, H_4list, H_5list. Whenever $\mathcal{A}_{\mathbf{II}}$ request access to any hash function, $\mathcal{C}$ responds as follows:

H queries: On receiving a query $(<Q>, \alpha)$, $\mathcal{C}$ searches Hlist and returns α as answer if found. Otherwise, chooses $\alpha \in_R \mathbb{Z}_q^*$ and returns α. $\mathcal{C}$ adds $(<Q>, \alpha)$ to the Hlist.

H_1 queries: On receiving a query $(<$ID,a$>, X)$, $\mathcal{C}$ searches H_1list and returns X as answer if found. Otherwise, chooses $X \in_R \mathbb{Z}_q^*$ and returns X. $\mathcal{C}$ adds $(<$ID,a$>, X)$ to the H_1list.

H_2 queries: On receiving a query $(<m, \sigma, ID, u>, R)$, $\mathcal{C}$ searches H_2list and returns R as answer if found. Otherwise, chooses $R \in_R \mathbb{Z}_q^*$ and returns R. $\mathcal{C}$ adds $(<m, \sigma, ID, u>, R)$ to the H_2list.

H_3 queries: On receiving a query $(<t, u>, A)$, $\mathcal{C}$ searches H_3list and returns A as answer if found. Otherwise, chooses $A \in_R \mathbb{Z}_q^*$ and returns A. $\mathcal{C}$ adds $(<t, u>, A)$ to the H_3list.

H_4 queries: On receiving a query $(<u>, B)$, $\mathcal{C}$ searches H_4list and returns B as answer if found. Otherwise, chooses $B \in_R \mathbb{Z}_q^*$ and returns B. $\mathcal{C}$ adds $(<u>, B)$ to the H_4list.

H_5 queries: On receiving a query $\left(<k_1, k_2, ID_i, pk_i, ID_j, pk_j>, W\right)$, $\mathcal{C}$ searches H_5list and returns W as answer if found. Otherwise, chooses $W \in_R \mathbb{Z}_q^*$ and returns W. $\mathcal{C}$ adds $\left(<k_1, k_2, ID_i, pk_i, ID_j, pk_j>, W\right)$ to the H_5list.

Phase1. $\mathcal{A}_{\mathbf{II}}$ issues any one of the following queries adaptively.

Partial Key Compute. $\mathcal{A}_{\mathbf{II}}$ computes the partial private key pair $\left(ppk_{ID_i}, psk_{ID_i}\right)$ for any ID_i of its choice, by computing compute $a_i = g^{\alpha_i}$, $x_i = \alpha_i + sH_1(ID_i, a_i)$, $(ppk_i, psk_i)=(a_i, x_i)$. $\mathcal{C}$ maintains a list of partial keys computed by $\mathcal{A}_{\mathbf{II}}$ in a partial key list $\left(ID_i, ppk_{ID_i}, psk_{ID_i}\right)$.

Public key request queries. On input ID by $\mathcal{A}_{\mathbf{II}}$, the challenger $\mathcal{C}$ searches whether there exists a tuple $(ID, pk_{ID}, coin) \in$ pklist. If not, $\mathcal{C}$ runs algorithm **SetPublicKey** to generate the public key pk_{ID} for entity ID, then adds the tuple $(ID, pk_{ID}, coin)$ to the pklist and return pk_{ID} to $\mathcal{A}_{\mathbf{II}}$. Otherwise, $\mathcal{C}$ returns pk_{ID} to $\mathcal{A}_{\mathbf{II}}$. The value of coin is decided the challenger $\mathcal{C}$'s strategy.

Private key extract queries. On input identity ID by $\mathcal{A}_{\mathbf{II}}$, the challenger $\mathcal{C}$ searches whether there exists a tuple $(ID, pk_{ID}, coin) \in$ pklist. If coin $\neq \perp$, $\mathcal{C}$ runs algorithm **SetPrivateKey** to generate the private key sk_{ID} for entity ID. If coin $= \perp$, $\mathcal{C}$ returns "Reject".

Re-encryption key extract queries. On input $\left(ID_i, ID_j\right)$ by $\mathcal{A}_{\mathbf{II}}$, the challenger $\mathcal{C}$ searches whether there exists a tuple $\left(ID_i, pk_{ID_i}, coin_i\right) \in$ pklist. If $coin_i \neq \perp$, $\mathcal{C}$ responds by running algorithm **ReEncryptKey** to generate the re-encryption key $rk_{ID_i \to ID_j}$ for entity ID_i, ID_j. Otherwise, $\mathcal{C}$ returns "Reject".

Re-encryption queries. On input $\left(ID_i, ID_j C_{IDi}\right)$ by $\mathcal{A}_{\mathbf{II}}$, the challenger $\mathcal{C}$ searches whether there exists a tuple $\left(ID_i, pk_{ID_i}, coin_i\right) \in$ pklist. If $coin_i \neq \perp$, $\mathcal{C}$ responds by running algorithm **ReEncrypt** to convert the second level ciphertext c_{ID_i} into the first level ciphertext c_{ID_j}. Otherwise, returns "Reject".

Decryption queries for second level ciphertext. On input $(\mathrm{ID}, \mathrm{c})$ by $\mathcal{A}_{\mathbf{II}}$, if c is a second level ciphertext, $\mathcal{C}$ responds by running algorithm $\mathbf{Decrypt_2}$ using the related private key to decrypt the C and returns the result to $\mathcal{A}_{\mathbf{II}}$. Otherwise, returns "Reject".

Decryption queries for first level ciphertext. On input $(\mathrm{ID}, \mathrm{c})$ by $\mathcal{A}_{\mathbf{II}}$, if c is a first level ciphertext, $\mathcal{C}$ responds by running algorithm $\mathbf{Decrypt_1}$ using the related private key to decrypt the c and returns the result to $\mathcal{A}_{\mathbf{II}}$. Otherwise, returns "Reject".

Challenge. Once the adversary $\mathcal{A}_{\mathbf{II}}$ decides that Phase 1 is over, it outputs the challenge identity ID_*, and two equal length plaintexts $\mathrm{m}_0, \mathrm{m}_1 \in \mathcal{M}$. Moreover, $\mathcal{A}_{\mathbf{II}}$ is restricted to choose a challenge identity ID_* that trivial decryption is not possible. $\mathcal{C}$ searches a tuple $\left(\mathrm{ID}_*, \mathrm{pk}_{\mathrm{ID}_*}, \mathrm{coin}_*\right) \in$ pklist, then picks a random bit $\delta \in \{0, 1\}$, and computes the challenge ciphertext c_* = Encrypt $\left(\mathrm{params}, \mathrm{ID}_*, \mathrm{pk}_{\mathrm{ID}_*}, \mathrm{m}_\delta\right)$, and returns c_* to $\mathcal{A}_{\mathbf{II}}$.

Phase2. The adversary $\mathcal{A}_{\mathbf{II}}$ continues to query any of the above mentioned oracles with the restrictions defined in the IND-CLPRE-CCA "Game II".

Guess. Finally, the adversary $\mathcal{A}_{\mathbf{II}}$ outputs a guess $\delta' \in \{0, 1\}$ and wins the game if $\delta' = \delta$.

We define the advantage of $\mathcal{A}_{\mathbf{II}}$ in "Game II" as $\mathrm{Adv}^{\mathrm{IND-CLPRE-CCA}}_{\mathrm{GameII}, \mathcal{A}_{\mathrm{II}}}(\mathrm{k}) = \left|\Pr\left[\mathrm{X}' = \mathrm{X}\right] - \frac{1}{2}\right|$.

A CLPRE scheme is said to be (t, ϵ)-2nd-IND-CLPRE-CCA secure if for any t-time 2nd-IND-CLPRE-CCA Type-II adversary $\mathcal{A}_{\mathbf{II}}$ we have $\mathrm{Adv}^{\mathrm{IND-CLPRE-CCA}}_{\mathrm{GameII}, \mathcal{A}_{\mathrm{II}}}(\mathrm{k}) = |\Pr[\mathrm{X}' = \mathrm{X}]| < \epsilon$. We simply say that a CLPRE scheme is 2nd-IND-CLPRE-CCA secure if t is polynomial with respect to security parameter k and ϵ is negligible.

7 Performance Evaluation

We make comparison between our scheme and Sur et al.'s scheme, Yang et al.'s scheme and Srinivasan et al.'s scheme, in terms of computational cost and ciphertext size. The number of "bignum" operations that CLPRE schemes need to perform are considered, other operations like addition or multiplication in group, XOR operation and conventional hash function evaluation is omitted, since the computation of these operations is efficient and far less than that of exponentiations or pairings. two modular exponentiations and the three modular exponentiations can be computed at a cost of about 1.17 and 1.25 exponentiations respectively, using simultaneous multiple exponentiation algorithm mentioned in [23]. The Map-To-Point hash function is special and far more expensive than conventional hash operation so it cannot be omitted.

From the Tables, we can find that all of our five algorithm listed in the table are much more computation efficient than Sur et al. [5]'s scheme and Yang et al. [14]'s scheme. Our encrypt algorithm and ciphertext size of the first and second level ciphertext are more efficient than those in Srinivasan et al. [15]'s scheme (Table 1).

Table 1. Comparison of CLPRE schemes' performance.

Schemes	Sur [5]	Yang [14]	Srinivasan [15]	Our CLPRE scheme
Encrypt	$5.08t_e$	$3.25t_e$	$3t_e$	$2.25t_e$
SetReEncKey	$4.08t_e$	$2.17t_e$	$2t_e$	$2.17t_e$
ReEncrypt	$6t_p$	$2.17t_e$	t_e	t_e
Decrypt_2	$2t_p+3.08t_e$	$3.17t_e$	$2t_e$	$2t_e$
Decrypt_1	t_p+4t_e	$4.25t_e$	$4t_e$	$4.25t_e$
$\|c_i\|$	$3\|G_1\|+\|m\|+\|\sigma\|$	$2\|G\|+\|\mathbb{Z}_q^*\|+\|m\|+\|\sigma\|$	$2\|G\|+\|\mathbb{Z}_q^*\|+\|m\|+\|\sigma\|$	$\|G\|+\|m\|+\|\sigma\|$
$\text{rk}_{i\to j}$	$3\|G_1\|$	$\|\mathbb{Z}_q^*\|$	$\|\mathbb{Z}_q^*\|$	$\|\mathbb{Z}_q^*\|$
$\|c_j\|$	$\|G_1\|+2\|G_2\|+\|m\|+\|\sigma\|$	$\|G\|+\|m\|+\|\sigma\|$	$3\|G\|+\|m\|+\|\sigma\|$	$\|G\|+\|m\|+\|\sigma\|$
Pairing-Free	×	✓	✓	✓
Map-To-Point-Free	×	✓	✓	✓
Assumption	p-BDHI	CDH	M-CDH	CDH

8 Conclusion

We provide a lightweight proxy certificateless proxy re-encryption (CLPRE) scheme without pairing. We also prove security of our CLPRE scheme against Type II adversary $\mathcal{A}_{\mathbf{II}}$ under the CDH assumption in the random oracle model, and make comparison with representative CLPRE scheme. The results show that our scheme is much more computational and communicational efficient than Sur et al.'s scheme and Yang et al.'s scheme, and outperform Srinivasan et al.'s scheme in terms of space efficiency.

References

1. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054122
2. Xu, L., Wu, X., Zhang, X.: CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, pp. 1–10 (2012)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. (TISSEC) **9**(1), 1–30 (2006)
4. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40061-5_29

5. Sur, C., Jung, C.D., Park, Y., Rhee, K.H.: Chosen-ciphertext secure certificateless proxy re-encryption. In: De Decker, B., Schaumüller-Bichl, I. (eds.) CMS 2010. LNCS, vol. 6109, pp. 214–232. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13241-4_20
6. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of ACM CCS 2007, pp. 185–194 (2007)
7. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy reencryption. IEEE Trans. Inf. Theory **57**(3), 1786–1802 (2011)
8. Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption without pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89641-8_1
9. Hanaoka, G., et al.: Generic construction of chosen ciphertext secure proxy re-encryption. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 349–364. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_22
10. Zheng, Y., Tang, S., Guan, C., Chen, M.-R.: Cryptanalysis of a certificateless proxy re-encryption scheme. In: 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp. 307–312. IEEE (2013)
11. Guo, H., Zhang, Z., Zhang, J., Chen, C.: Towards a secure certificateless proxy re-encryption scheme. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 330–346. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41227-1_19
12. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005). https://doi.org/10.1007/11556992_10
13. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_2
14. Yang, K., Xu, J., Zhang, Z.: Certificateless proxy re-encryption without pairings. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 67–88. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-12160-4_5
15. Srinivasan, A., Rangan, C.P.: Certificateless proxy re-encryption without pairing: revisited. In: Proceedings of the 3rd International Workshop on Security in Cloud Computing, pp. 41–52. ACM (2015)
16. Liu, Y., Peng, H., Wang, J.: Verifiable diversity ranking search over encrypted outsourced data. CMC Comput. Mater. Contin. **55**(1), 037–057 (2018)
17. Tang, Y., Lian, H., Zhao, Z., Yan, X.: A proxy re-encryption with keyword search scheme in cloud computing. CMC Comput. Mater. Contin. **56**(2), 339–352 (2018)