Non-Transferable Proxy Re-Encryption for Group Membership/ Non-Group Membership

Lwin San, Ei Mon Cho, Takeshi Koshiba
Saitama University, Japan
s15mm353@mail.saitama-u.ac.jp, s14dm054@mail.saitama-u.ac.jp
koshiba@mail.saitama-u.ac.jp

Abstract

In proxy re-encryption (PRE) scheme the message is sent by a delegator to a delegatee with a help of the trusted third party proxy without knowing the existing plaintext. PRE schemes are widely used in various applications. However, the standard PRE scheme has some proxy problems called private key generator (PKG) despotism problem. This means that PKG can make re-encryption key without getting permission from the delegator. And also, most of the PRE schemes have the key-escrow problem. If someone can attack PKG in PRE scheme, they can decrypt both the original ciphertext and the re-encrypted ciphertext which means the key-escrow problem. A solution for these two problems is to use non-transferable PRE scheme. Non-transferable PRE scheme solved the above PKG despotism problem and key-escrow problem. We would like to introduce our PRE scheme with a new approach. In our scheme, there are three sub-processes, which are based on non-transferable PRE scheme and group signature. Our scheme will provide the security for delegator i, delegatee j (who is in the same group with delegator i), and delegatee k (who is in a different group from delegator i).

1. Introduction

A cryptographic proxy re-encryption scheme was introduced in 1998 by Blaze, Bleumer, and Straus [1]. In their PRE scheme, the proxy with re-encryption keys can change a ciphertext for Alice (delegator) into another ciphertext of the same plaintext for Bob (delegatee). The proxy in their scheme cannot get any information about the plaintext or the private key. The PRE scheme can be classified into two parts, which are unidirectional PRE and bidirectional PRE schemes [7]. In the unidirectional scheme, a message can be re-encrypted from the proxy to Bob, but the reversible process cannot occur. And also, the proxy could delegate to Bob by combining his private key with Bob public key. In bidirectional scheme, the re-encryption key can be used to translate messages from the proxy to Bob, as well as from Bob to the proxy. In this scheme, both the proxy and Bob must combine their private keys to produce the re-encryption key. The above scheme got much interest from the cryptography community [1, 6, 7] and has many interesting and useful applications, such as:

• Email forwarding: The delegator wants to delegate his email to the delegatee. The proxy will send the re-encrypted email to the delegatee. The delegatee can access the email without knowing the delegator's decryprion key [9].

• Encrypted files distribution: The encrypted files are stored in a file server. The owner can only grant the access right of the files to the receivers; even the file server operator has no permission to access the data [6,10].

In this paper, we propose a new PRE scheme which is secure for group members or non-group members. In this scheme, we have three different decryption methods which are for delegator i, delegatee j, and delegatee k. We organize the paper as follows. In Section 2, we introduce preliminaries. In Section 3, we propose our scheme. We construct the proposed scheme in Section 4. We compare the previous works to our schemes in Section 5. In Section 6, we discuss about the security of our proposed scheme. Finally, we conclude in Section 7.

1.1. Related Works

The notion of "atomic proxy cryptography" was introduced by Blaze et al. [1] in 1998. This scheme can get secure and better way than usual. In this scheme, proxy decrypts a message by using Alice's secret key and then encrypts the result by using Bob's public key. In 2005, Ateniese et al. [6] was introduced the notion of "non-transferability". There have many attempts to prevent the transferability of PRE. Libert and Vergnaud [2] proposed “traceable proxy re-encryption” in 2008. In their scheme, the malicious proxy can reveal the re-encryption key to the third party can be found by the delegator. Therefore, this scheme still cannot solve the proxy colluding problem.

In 2011, Wang et al. [3] proposed “identity-based proxy re-encryption scheme” to solve the problem of proxy colluding. Furthermore, the proxy and the delegatees cannot transfer the decryption right to others without the permission of the public key generator. However, the colluding with PKG can decrypt both original ciphertext and the re-encrypted ciphertext. That means the transferable problem. To solved the above problem, Y. He et al. [5] proposed “non-transferable proxy re-encryption scheme” in 2012, and the proxy or delegatee cannot collude to transfer decryption right. They introduced two properties, namely “Non-key-escrow” and “Non-PKG-despotism”.

In 2009, Ma et al. [4] proposed “group-based proxy re-encryption scheme”. In their scheme, the proxy diverts the ciphertext for group A into group B, and vice versa. Any member from group A/B can independently decrypt the ciphertext for the group. However, Ma’s group re-encryption also have same proxy colluding problem.

2. Preliminaries

In this section, we will present some primitives that will be used in our scheme.

2.1. Bilinear Map

Let G and G_T be multiplicative cyclic groups of prime order p, and g be generator of G. We say that G_T has an admissible bilinear map $e : G \times G \rightarrow G_T$, if the following conditions hold [4, 5].

- $e(g^a, g^b) = e(g, g)^{a,b}$ for all a,b.
- $e(g, g) \neq 1$.
- There is efficient algorithm to compute $e(g^a, g^b)$ for all a,b and g.

2.2. Assumption

We use the definition of computational Diffie-Hellman Assumption and Truncated Decision Augmented Bilinear Diffie-Hellman Exponent Assumption by using the bilinear map.

Definition.1. (Computational Diffie-Hellman Assumption) [4] Given g^a and g^b for some $a,b \in Z_q^*$, compute $g^{ab} \in G_1$. A (τ, ε)-CDH attacker in G_1 is a probabilistic machine Ω with running time τ such that

$$Succ_{cdh}^{G_1}(\Omega) = \Pr[\Omega(g, g^a, g^b) = g^{ab}] \geq \varepsilon$$

where the probability is taken over the random values a and b. The CDH problem is (τ, ε) intractable if there is no (τ, ε)-attacker in G_1. The CDH assumption states that it is the case for all polynomial τ and any non-negligible ε [4].

Definition.2. (Truncated Decision Augmented Bilinear Diffie-Hellman Exponent Assumption) The security of our proposed scheme is based on a complexity assumption named Truncated q-ABDHE [5] which has been proposed by [13]. Let $e : (G \times G) \rightarrow G_T$ be a bilinear map, where G and G_T are cyclic groups of large prime order p. Given a vector of $q + 3$ elements:

$$\left(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, \dots, g^{(\alpha^q)}\right) \in G^{q+3}$$

and an element $Z \in G_T$ as input, output 0 if $Z = e\left(g^{\alpha^{q+2}}, g'\right)$ and output 1 otherwise. An algorithm B has advantage ε in solving the truncated q-ABDHE if:

$$\left| Pr\left[B\left(g', g'^{(\alpha^{q+2})}, g, g^{\alpha}, \dots, g^{(\alpha^q)}, e\left(g^{(\alpha^{q+1})}, g'\right)\right) = 0\right] \right.$$
$$\left. - Pr[B(g', g'^{(\alpha^{q+2})})] \right| \geq \varepsilon$$

where the probability is over the random choice of generators g, g' in G, the random choice of α in Z_p, the random choice of $Z \in G_T$, and the random bits is consumed by B.

3. Our Proposed PRE Scheme

Our proposed scheme consists of following thirteen sub-algorithms.

• Setup. From the input of a security parameter 1^k, the public parameters *mpk* and master secret key *msk* are generated.

• Key Generation.

- Set-Secret-Value: generates a secret value which is only known to the user himself.

- Partial-Private-Key: inputs a user’s identity *ID*, *msk*, and generates partial private key for the user.

- Set-Private-Key: inputs the partial private key and the secret value, outputs the private key for user.

- Set-Public-Key: inputs a user's identity *ID* and secret value, and generates public key (upk).

• Private Key Correctness Check: checks the correctness of the private key.

• Encryption: takes public key upk_i of delegator i and message m as input, outputs a ciphertext C_i encrypted under upk_i.

• Decryption (delegator i): takes private key usk_i of delegator i and ciphertext C_i as input, outputs message m.

• Re-Encryption Key Generation (delegate j): verifies the delegator *j's* signature, and public key. The re-encryption key generation algorithm outputs a re-encryption key rk_j and other relational values.

• Partial-Decryption-Key Generation (delegate j): checks the correctness of the re-encryption key, and generates a partial decryption key.

• Re-Encryption: takes re-encryption key rk_j and ciphertext C_i as input, outputs a re-encrypted ciphertext C_j under upk_j .

• Decryption (delegatee j): takes private key usk_j of delegatee j, partial decryption key and ciphertext C_i as input, outputs message m.

• Re-Encryption Key Generation (delegate k): verifies the delegator *i's* signature, and extracts delegatee *k's* *ID* from the signature. The re-encryption key generation algorithm outputs a re-encryption key $rk_{i\rightarrow j}$ and other relational values.

• Partial-Decryption-Key Generation (delegatee k): checks the correctness of the re-encryption key, and generates a partial decryption key.

• Re-Encryption: takes re-encryption key $rk_{i\rightarrow j}$ and ciphertext C_i as input, outputs a re-encrypted ciphertext C_j under *upk*.

• Decryption (delegatee k): takes private key usk_k of delegatee k, partial decryption key and ciphertext C_j as input, outputs message m.

4. Construction of the Scheme

In this section, we construct a new PRE scheme which is fully satisfying for both the group member and the non-group member. Our scheme is based on the Non-transferable scheme [5]. We add new primitive for the non-group membership which is the delegator and delegate do not exist in the group. The basic behind our scheme is PRE scheme which will allow re-encrypting at the proxy. And the delegate can decrypt the ciphertext using his private key.

Setup: Let G and G_T be groups of order p such that p is a n-bit prime, and let $e : G\times G \rightarrow G_T$ be the bilinear map. $H_I : \{0, 1\}^* \rightarrow Z_p$, $H' : G_T \rightarrow Z_p$ are secure hash functions. The PKG selects four random generators $h_1, h_2, h_3, g \in G$ and randomly chooses $\alpha \in Z_p$. It sets $g_1 = g^{\alpha}$. Define the message space $M \in G_T$. The public parameters *mpk* and master secret key *msk* are given by $mpk = (g, g_1, h_1, h_2, h_3, H_1, H, H', M)$, $msk = (\alpha)$.

Key Generation: This is a protocol through which a user U with an identity *ID* can securely get his partial private key from PKG. On input the public key/master secret key pair (*mpk*, *msk*) and an identity $ID_i \in \{0, 1\}^n$ of entity i, the PKG computes $id_i = H_I(ID_i)$. If $id_i = \alpha$, it aborts. Otherwise, the protocol proceeds as follow:

- Set-Secret-Value. Entity i selects $r_i \in Z_p$ at random. r_i is *i's* secret value.
- Partial-Private-Key-Extract.
  1. A sends $R = h_1^{r_i}$ to PKG, and gives PKG the following zero-knowledge proof of knowledge: $PK\{r_i: R = h_1^{r_i}\}$.
  2. PKG randomly selects $r'_i, r_{i,2}, r_{i,3} \in z_P$ and computes $h'_i = (Rg^{r'_i})^{\frac{1}{\alpha - id_i}}, h_{i,2} = (h_2 g^{-r_{i,2}})^{1/(\alpha - id_i)}$, $h_{i,3} = (h_2 g^{-r_{i,3}})^{1/(\alpha - id_i)}$ and sends *i's* partial private key $(r'_i, h'_i, r_{i,2}, h_{i,2}, r_{i,3}, h_{i,3})$ to i.
- Set-Private-Key. i compute $r_{i,1} = \frac{r'_i}{r_i}, h_{i,1} = (h'_i)^{\frac{1}{r_i}} = (h_1 g^{-r_{i,1}})^{1/(\alpha - id_i)}$. Then, *i's* private key can be denoted as $usk_i = (r_i, r_{i,1}, h_{i,1}, r_{i,2}, h_{i,2}, r_{i,3}, h_{i,3})$. Similarly, the delegate *j's* private key is denoted as $usk_i = (r_j, r_{j,1}, h_{j,1}, r_{j,2}, h_{j,2}, r_{j,3}, h_{j,3})$.
- Set- Public -Key. A publishes her public key $upk_i = (p_{i,1}, p_{i,2})$, where $p_{i,1} = g_1^{r_i}$, and $p_{i,2} = g^{r_i id_i}$. Anyone can verify the validity of upk_i by checking if the equalities $e(g^{id_i}, p_{i,1}) = e(g_1, p_{i,2})$, and $e(h_1^{r_i}, g_1) = e(h_1, p_{i,1})$ hold (i.e., $h_1^{r_i}$ can be obtained from PKG)

Private Key Correctness Check: On input (*mpk*, usk_{ID}) and checks whether $e\left(h_{i,t}, \frac{g_1}{g^{id_i}}\right) = e(h_t g^{-r_{i,1}}, g)$ for *t = 1, 2, 3.* If correct, output 1. Otherwise, output 0.

From now, the following steps vary depending on the type of delegatee.

4.1. PRE scheme for Owner (Delegator *i*)

If the delegatee *i* decrypts his own message, the encryption and the decryption steps done as normal public key cryptography schemes. See the details as following process.

Encryption: [**Enc** (*m*, upk_i) → Ciphertext C_i]

To encrypt a message $m \in G_T$ using public key, sender checks that whether the equalities $e(g^{id_i}, p_{i,1}) = e(g_1, p_{i,2})$ and $e(h_1^{r_i}, g_1) = e(h_1, p_{i,1})$ hold. If not, output ⊥ and abort encryption. Otherwise, sender generates a unique randomly-selected secret parameter $s \in Z_p$, and computes $id_i = H_I(ID_i)$. Finally, sender outputs the ciphertext *C* where: $C = (C_1, C_2, C_3, C_4, C_5, C_6)$

$= (p_{i,1}^{s} p_{i,2}^{-s}, e(g,g)^s, m.e(g,h_1)^{-s},\ e(g,g)^{H'(m)}, g^{s\beta+H'(m)}, e(g, h_3^{s\beta}))$. We set $\beta = H(C_1, C_2, C_3, C_4)$.

Decryption (delegator *i*): [**Dec** (usk_j, C_i) → Message *m*]

To decrypt a ciphertext $C = (C_1, C_2, C_3, C_4, C_5, C_6)$ using secret key usk_i, delegator *i* computes $\beta = H(C_1, C_2, C_3, C_4)$. and tests whether $e(C_5, g) = C_2^{\beta} C_4$ and $C_6 = e(C_1, h_{i,2} h_{i,3}^{\beta})^{1/r_i} \cdot C_2^{r_{i,2}+r_{i,3}\beta}$. If it is not equal, outputs will be⊥. Else then it computes $m = C_3 \cdot e(C_1, h_{i,1})^{1/r_i} \cdot C_2^{r_{i,1}}$. If $e(g,g)^{H'(m)} = C_4$ holds, then return *m*; otherwise return ⊥.

4.2 PRE scheme for Group member (Delegatee *j*)

If the delegator *i* and the delegatee *j* are in the same group, then the delegatee *j* can receive the message by his own signature (without any request the signature of delegator *i*). The encryption method of the sub process follows the encryption of owner/delegator as in section 4.1.

Re-Encryption–Key-Generation: [**ReKeyGen**(ID_j, upk_j) → rk_j]

1. Delegatee *j* belongs to the same group of delegator *i*. Delegatee *j* is only allowed to decrypt messages intended for delegator *i* during some specific time period *t*. To achieve this property, the delegator *i* generates a random value $a_t \in Z_p$ for each time period *t*, where $t \geq 1$. a_t will be invalid after the period *t*. Delegatee *j* sign the message and send ID_j and upk_j to the PKG secure channel. Delegator sign:
   - Choose $z \in Z_p$, and compute $U = g^z$.
   - Compute $V = H_I(ID_j, U)$.
   - Compute $W = g^{\alpha r_i + V}$.
   - The signature of *j* is $\sigma = (U,W)$.
2. PKG verifies the signature of delegatee *j* to identify whether delegatee *j* is is from the same group of delegator *i* or not. PKG Verify:
   - Compute $V = H_I(ID_j, U)$.
   - Accept the signature if
     $$e(h_1, W) = e(h_1^{r_i}, g^{\alpha}) e(h_1, g)^V.$$
3. If verification passes, PKG generates a unique randomly-selected secret parameter $y \in Z_p$, and computes re-encryption key $rk_{i\to j} = \left(\frac{\alpha - id_j}{\alpha - id_i} + a_t y\right) mod\ p,\ i_1 = (h_1^{r_i} g^{-r'_i})^y, j_1 = (h_1^{r_j} g^{-r'_j})^{\frac{a_t y}{\alpha - id_j}}, j_2 = h_1^{a_t y}$ and sends $rk_{i\to j}, i_1, j_1, j_2$ to delegator *i*.

Partial-Decryption-Key Generation: [**PartialDecKeyGen** (rk_j) → pdk_j]

4. Delegatee *j* sends h'_j to delegator *i* via a secure and authenticated channel.
5. Delegator *i* checks whether $e(h_1, j_1) = e(j_2, h' - j)$ to ensure j_1 is a valid value which will help delegatee for decryption later. If correct, output *1*, otherwise, output *0*.
6. Delegator *i* checks whether $h'^{(id_i - id_j)}_i \times i_1^{a_t} \times (h_1^{r_i} g^{-r'_i}) = (h_1^{r_i} g^{-r'_i})^{rk_{i\to j}}$ to ensure that $rk_{i\to j}$ is a re-encryption key generated properly for delegation from her to delegator *j*.
7. Delegator *i* sends the re-encryption key $rk_{i\to j}$ to Proxy via an authenticated channel.
8. Delegator *i* computes h'^{1/r_i}_j and j_1^{1/r_i} and sends them to delegator *j* as partial decryption key.

Re-Encryption: [**ReEnc**(rk_j, C_i, upk_j) → Ciphertext C_i]

Proxy computes $\beta = H(C_1, C_2, C_3, C_4)$. and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, outputs ⊥.Else computes $C' = C_1^{rk_{i\to j}} = g^{r_i s(\alpha - id_i)(\frac{\alpha - id_j}{\alpha - id_i} + a_t y)}$ and sends $C = (C', C_1, C_2, C_3, C_4, C_5)$ to delegate *j*.

Decryption (delegatee *j*): [**Dec**(usk_j, pdk_j, C_j) → Message *m*]

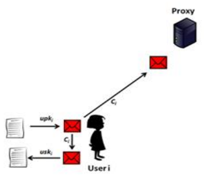

Fig. (a) Scheme for Delegator *i*

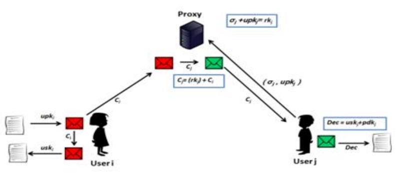

Fig. (b) Scheme for Delegatee *j*

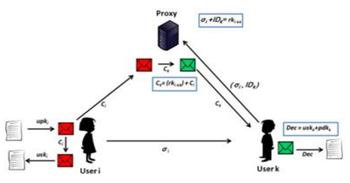

Fig. (c) Scheme for Delegatee *k*

Figure .1 Proxy Re encryption Scheme

Delegatee *j* computes $\beta = H(C_1, C_2, C_3, C_4))$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, outputs ⊥. Else delegatee *j* computes

$$C_3 \cdot \frac{e(C'_1, h'^{\left(\frac{1}{r_i}\right)\left(\frac{1}{r_j}\right)}_1) C_2^{r_{j,1}}}{e(C_1, j_1^{\left(\frac{1}{r_i}\right)\left(\frac{1}{r_j}\right)})}$$

$$= C_3 \cdot \frac{e\left(g^{r_i s(\alpha - id_i)\left(\frac{\alpha - id_j}{\alpha - id_i} + a_t y\right)}, (h_1 g^{-r_{j,1}})^{\frac{1}{(\alpha - id_j) r_i}}\right)(e(g,g)^s)^{r_{j,1}}}{e\left(g^{r_i s(\alpha - id_i)}, (h_1 g^{-r_{j,1}})^{\frac{a_t y}{(\alpha - id_j) r_i}}\right)}$$

$$= C_3 \cdot e\left(g^{s(\alpha - id_i)\left(\frac{\alpha - id_j}{\alpha - id_i}\right)}, (h_1 g^{-r_{j,1}})^{\frac{1}{(\alpha - id_j)}}\right) e(g,g)^{s * r_{j,1}}$$

$$= m \cdot e(g, h_1)^{-s} e(g^{s(\alpha - id_j)}, (h_1 g^{-r_{j,1}})^{\frac{1}{(\alpha - id_j)}}) \, e(g,g)^{s r_{j,1}}$$

$$= m.$$

4.3 PRE scheme for Non-group member (Delegatee *k*)

If the delegator *i* and the delegatee *k* are in the different group, then protocol follows as in Fig.1. (c) i.e, the delegator *i* has to send the signature to the delegatee *k* for receiving the message. The encryption method of the sub process follows the encryption of owner/delegator as in section 4.1.

Re-Encryption–Key-Generation: [ReKeyGen(*IDₖ*, *upkₖ*) → *rkₖ*]

1. Delegatee *k* belongs to the same group of delegator *i*. Delegatee *k* is only allowed to decrypt messages intended for delegator *i* during some specific time period *t*. To achieve this property, the delegator *i* generates a random value $a_t \in Z_p$ for each time period *t*, where $t \geq 1$. a_t will be invalid after the period *t*. Delegatee *k* signs the message and sends ID_k and upk_k to the PKG secure channel. Delegator sign:
   - Choose $z \in Z_p$, and compute $U = g^z$.
   - Compute $V = H_I(ID_k, U)$.
   - Compute $W = g^{\alpha r_i + V}$.
   - The signature of *k* is $\sigma = (U, W)$.
2. PKG verifies the signature of delegatee *k* to identify whether delegatee *k* is from the same group of delegator *i* or not. PKG Verify:
   - Compute $V = H_I(ID_k, U)$.
   - Accept the signature if $e(h_1, W) = e(h_1^{r_i}, g^{\alpha}) e(h_1, g)^V$.
3. If verification passes, PKG generates a unique randomly-selected secret parameter $y \in Z_p$, and computes re-encryption key $rk_{i \to k} = \left(\frac{\alpha - id_k}{\alpha - id_i} + a_t y\right) mod\ p,\ i_1 = (h_1^{r_i} g^{-r'_i})^y, k_1 = (h_1^{r_k} g^{-r'_k})^{\frac{a_t y}{\alpha - id_k}}, k_2 = h_1^{a_t y}$ and sends $rk_{i \to k}, i_1, k_1, k_2$ to delegator *i*.

Partial-Decryption-Key Generation: [PartialDecKeyGen (*rkₖ*) → *pdkₖ*]

4. Delegatee *k* sends h'_k to delegator *i* via a secure and authenticated channel.
5. Delegator *i* checks whether $e(h_1, k_1) = e(k_2, h' - k)$ to ensure k_1 is a valid value which will help delegatee for decryption later. If correct, output *1*, otherwise, output *0*.
6. Delegator *i* checks whether $h'^{(id_i - id_k)}_i \times i_1^{a_t} \times (h_1^{r_i} g^{-r'_i}) = (h_1^{r_i} g^{-r'_i})^{rk_{i \to k}}$ to ensure that $rk_{i \to k}$ is a re-encryption key generated properly for delegation from her to delegator *k* .
7. Delegator *i* sends the re-encryption key $rk_{i \to k}$ to Proxy via an authenticated channel.
8. Delegator *i* computes h'^{1/r_i}_k and k_1^{1/r_i} and sends them to delegator *k* as partial decryption key.

Re-Encryption: [ReEnc (*rkₖ*, *Cᵢ*, *upkₖ*) → Ciphertext *Cᵢ*]

Proxy computes $\beta = H(C_1, C_2, C_3, C_4)$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, outputs will be⊥. Else it computes $C' = C_1^{rk_{i \to k}} = g^{r_i s(\alpha - id_i)(\frac{\alpha - id_k}{\alpha - id_i} + a_t y)}$ and sends $C = (C', C_1, C_2, C_3, C_4, C_5)$ to delegate *j*.

Decryption (delegatee *k*): [Dec (*uskₖ*, *pdkₖ*, *Cₖ*) → Message *m*]

Delegatee k computes $\beta = H(C_1, C_2, C_3, C_4)$ and tests whether $e(C_5, g) = C_2^{\beta} C_4$. If it is not equal, outputs $\perp$. Else delegatee k computes

$$C_3 \cdot \frac{e(C'_1, {h'}_1^{\left(\frac{1}{r_i}\right)\left(\frac{1}{r_k}\right)}) C_2^{r_{k,1}}}{e(C_1, k_1^{\left(\frac{1}{r_i}\right)\left(\frac{1}{r_k}\right)})}$$

$$= C_3 \cdot \frac{e\left(g^{r_i s(\alpha - id_i)\left(\frac{\alpha - id_k}{\alpha - id_i} + a_t y\right)}, (h_1 g^{-r_{k,1}})^{\frac{1}{(\alpha - id_k) r_i}}\right)(e(g,g)^s)^{r_{k,1}}}{e\left(g^{r_i s(\alpha - id_i)}, (h_1 g^{-r_{k,1}})^{\frac{a_t y}{(\alpha - id_k) r_i}}\right)}$$

$$= C_3 \cdot e\left(g^{s(\alpha - id_i)\left(\frac{\alpha - id_k}{\alpha - id_i}\right)}, (h_1 g^{-r_{k,1}})^{\frac{1}{(\alpha - id_k)}}\right) e(g,g)^{s * r_{k,1}}$$

$$= m \cdot e(g, h_1)^{-s} e(g^{s(\alpha - id_k)}, (h_1 g^{-r_{k,1}})^{\frac{1}{(\alpha - id_k)}})\; e(g,g)^{s r_{k,1}}$$

$$= m.$$

5. Comparison of previous works

The main advantage of our scheme is non-transferable property for group member or non-group members. To compare our scheme and existing, we analyze by using the following properties.

• Unidirectional: "Delegator *A* to delegatee *B*" does not allow reverse "*B* to *A*".

• Bidirectional: The re-encryption scheme is reversible. The re-encryption key can be used to translate the message from proxy to delegatee, as well as from delegatee to proxy.

• Proxy-invisible: Although the delegator needs to know the existence of proxy for re-encryption step, delegatee does not need to know that. We have to determine the proxy is invisible for delegator or invisible for delegatee.

• Collusion-resistance: The delegatee and proxy's collusion can receive the output from delegator. But they cannot recover the secrete key of the delegator.

• Non-Transferable: Even though proxy and delegatee *B* devise re-encryption for another person, the re-encryption key of $rk_{A \to C}$ is impossible to produce without the permission.

• Non-Transitive: Proxy cannot generate the re-encryption key $rk_{A \to C}$ based on re-encryption key of *A* to *B* ($rk_{A \to B}$) and re-encryption key of *B* to *C* ($rk_{B \to C}$). The result of the comparison is as shown in Table 1.

• Certificateless: The certificateless cryptosystem does not need certificate in public key cryptosystem and it solves the key escrow problem.

Table 1. Comparison of Previous Works

Property	HCLHY[5]	MA [4]	Our Scheme
Unidirectional	Yes	Yes	Yes
Bidirectional	No	Yes	No
Proxy-Invisible	Yes	No	Yes
Collusion-resistance	Yes	No	Yes
Non- Transferable	Yes	No	Yes
Non-Transitive	Yes	Yes	Yes
Certificate less	No	No	Yes

6. Security

Firstly, we consider the CPA-security (Chosen-Plaintext Attack).

Theorem 1. If PKG is a secure pseudo-random generator, then our PRE scheme is CPA secure. **Proof:** The theorem is obtained by following the security proof of Guo's non-transferable PRE scheme [13].

Theorem 2. If PRG is a secure pseudo-random generator, then our PRE scheme is non-transferable against one-pair-of-keys collusion attacks.

Proof: The proposed construction allows one-pair-of-keys collusion attacks, that $|\boldsymbol{C} \cap \boldsymbol{D}| = 1$, where C denotes the set of corrupted users and D denotes the set of delegatees whose corresponding re-encryption keys are corrupted. Let $\{j^*\} = \boldsymbol{C} \cap \boldsymbol{D}$. For any polynomial time algorithm A which outputs an ϵ - useful decryption box $L_{i*,\, \epsilon}$ as a subroutine to invert any given encrypted ciphertext $C* \leftarrow Enc_2\,(pk_{j*}, m)$ of user $j*$ with probability ϵ, where $m \leftarrow M$.

7. Conclusion

In this paper, we solve the proxy colluding problem for group or non-group communication. We extend the notion of non-transferable PRE scheme which is used for one or more group communication. In our scheme, the proxy and the delegatees cannot collude because they are unable to generate re-encryption key for redelegating decryption right without the original delegator's help. To the members of different two groups, they can decrypt the ciphertext with the help of the proxy. In this paper, we proposed three sub schemes. Those are same delegator-delegatee, different group membership, and same grup membership. Apart from Y. He et al. [5], we used certificateless for same group membership. Our scheme allows every type of delegator-delegatee whether same group or different. In future work, we plan to proof the detailed security of the proxy re-encryption scheme.

References

[1]. M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography”, Advances in *EUROCRY PT 0*98, *LNCS* 1403: 127- 144.

[2]. B. Libert and D. Vergnaud, “Tracing malicious proxies in proxy re-encryption”, *Pairing – Based Cryptography − Pairing*2008, pages 332-353. Springer, 2008.

[3]. L. Wang, L. Wang, M. Mambo, and E. Okamoto, “New identity-based proxy re-encryption schemes to prevent collusion attacks”, *Pairing*2010, pages 327-346. Springer, 2010.

[4]. C. Ma, and J. Ao, “Group-Based Proxy Re-encryption Scheme”, *ICIC (1)*, pages 1025-1034. 2009.

[5]. Y. He, T. W. Chim, L. C. K. Hui, and S. Yiu, “Non-Transferable Proxy Re-Encryption Scheme”, *NTMS*, pages 1-4. 2012

[6].G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with application to secure distributed storage”, *DSS*, pages 29-43, February 2005.

[7]. A. Ivan and Y. Dodis, “Proxy cryptography revisited”, *NDSS*, February 2003.

[8]. H. Guo, Z. Zhang, J. Xu, “Non-Transferable Proxy Re-Encryption”, *IACRCryptology ePrintArchive*2015: 1216 (2015).

[9]. R. Canetti, and S. Hohenberger, “Chosen-ciphertext secure proxy re-encryption”, *ACM* conference on Computer and Communication Security, pages 185-1946. 2007.

[10]. G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with applications to secure distributed storage”, *ACM* Transactions on Information and System Security (*TISSEC*), 9(1): 1-30 (2006).

[11]. G. Taban, A. A. Cardenas, and V. D. Gligor, “Towards a secure and interoperable drm architecture”, *ACM* workshop on digital right management, pages 69-78. 2006.

[12]. L. Xu, X. Wu, and X. Zhang, “A certificateless proxy re-encryption scheme for secure data sharing with public cloud”, *ASIA CCS*, pages 87 – 88. 2012.

[13]. C. Gentry, “Practical identity-based encryption without random oracles”, *EUROCRYPT*, pages 445–464, May 2006.