A Fine-grained Conditional Proxy Broadcast Re-Encryption Policy for File Sharing System

Yongjun Ren^{1,2}, Yepeng Liu^{1,2}, Chengshan Qian^{1,3}

{renyj100@126.com [1], wind.bell@nuist.edu.cn [2], qianchengshan@163.com [3] }

School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, China [1]
Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology (CICAEET), Nanjing University of Information Science & Technology, Nanjing, China [2]
Binjiang College, Nanjing University of Information Science & Technology, Wuxi, China [3]

Abstract. With the proliferation of electronic devices, cloud storage has become an integral part of synchronizing files between devices. However, data stored on the cloud is vulnerable to unauthorized access during the sharing process. The traditional conditional proxy broadcast re-encryption solution is suitable for the above application scenarios but does not support flexible control of conditions. This article describes the concept of fine-grained conditional proxy broadcast re-encryption (FGC-BPRE). The scheme utilizes an access tree to generate a re-encryption key. If the ciphertext keyword satisfies the conditions in the access tree, the proxy can convert the user's ciphertext into a new ciphertext for a group of users. In addition, this paper also constructs a fine-grained conditional proxy broadcast re-encryption scheme and verifies the security of selective ciphertext attacks without oracles.

Keywords: Conditional proxy re-encryption, Proxy broadcast re-encryption, Cloud storage secure.

1 Introduction

As digital devices enter our lives, people are getting used to using cloud services to synchronize private files on different devices. Cloud service providers usually provide necessary security for private files. But this can cause problems, for example, Alice encrypted files cannot be directly shared with Bob for decryption. If Alice passes the private key directly to the agent for decryption, the agent will have access to all Alice's files. This is unacceptable to Alice and can significantly increase the risk of file compromise or unauthorized access, and encryption will lose its meaning. From another perspective, all encryption and decryption operations are performed by the agent, which makes it difficult to improve execution efficiency.

The proxy re-encryption (PRE) proposed by Blaze works well for the above situation [1]. PRE-scheme allows the agent to encrypt the user's ciphertext twice. The re-encrypted ciphertext can be decrypted directly by Bob, while the agent cannot obtain any valuable information. In the actual environment, PRE has a wide range of applications, such as cloud data sharing system [2], distributed file system [3], network backup system [4] and so on. However, in the original PRE scheme, the proxy can arbitrarily convert its encrypted file without Alice's consent, and

MOBIMEDIA 2019, June 29-30, Weihai, People's Republic of China
Copyright © 2019 EAI
DOI 10.4108/eai.29-6-2019.2282387

the conversion permissions are not well controlled. To solve this problem, Weng et al. proposed the concept of conditional proxy re-encryption (C-PRE) [5]. Under C-PRE, the agent can re-encrypt its ciphertext only after matching the conditions set by Alice. C-PRE has been put into practical use, such as social network data sharing systems [6]. In an actual application environment, Alice may need to share files to multiple people, at this time, the C-PRE scheme needs to perform multiple re-encryption operations, which takes up a lot of time and storage resources. The conditional proxy broadcast re-encryption proposed by Chu et al. can solve such a problem [7]. In this scheme, the proxy only needs to re-encrypt Alice's ciphertext once, and a group of users can decrypt the obtained ciphertext. The previous C-PRE and CPBRE schemes can not provide flexible control over proxy transition conditions. A proxy broadcast re-encryption scheme supporting variable condition numbers, arbitrary condition combinations, and partial condition matching has yet to be proposed.

Taking the file-sharing system as an example, Alice wants to hand over the working documents to the grassroots engineers in the R&D department in Hong Kong or Guangdong. She needs to describe the forwarding conditions $\mathcal{T}$ = ("Research and Development Department" $\wedge$"Grassroots Engineer" $\wedge$ ("Hongkong" $\vee$ "Guangdong")). In this article, we introduce the concept of fine-grained conditional proxy broadcast re-encryption. In the FGC-PBRE scheme, ciphertext is encrypted with a set of keywords W, and a re-encryption key is generated using the access tree $\mathcal{T}$. The agent can transform Alice's ciphertext into a group of users if and only if W satisfies $\mathcal{T}$.

1.1 Related Work

Blaze et al. introduced the concept of proxy re-encryption in Eu-rocrypt98 [1]. He proposed a two-way PRE scheme that chooses plaintext security. In this scheme, the proxy can convert Alice's ciphertext to Bob's ciphertext, or Bob's ciphertext to Alice's ciphertext. Subsequently, Ateniese et al. proposed a one-way proxy re-encryption based on bilinear pairing in ACM CCS 2005 and proved the CPA security of the scheme [8]. Weng et al. proposed a one-way proxy re-encryption scheme using the adaptive model, which is the first CCA-secured one-way proxy re-encryption scheme [9]. Deng et al. proposed a multi-directional proxy re-encryption scheme without bilinear pairing [10]. Chu et al. proposed a scheme for conditional proxy re-encryption [11]. The conditional proxy re-encryption implements the permission control of the encrypted ciphertext, and only the ciphertext is satisfying the specified condition set by Alice can be re-encrypted by the proxy. In ProvSec 2009, Fang et al. proposed an anonymous conditional proxy re-encryption scheme that supports conditional fault tolerance [12]. Luo et al. proposed the concept of ciphertext policy attribute proxy re-encryption [13].

1.2 Roadmap

Section two of this paper provides some related theorems. Section three illustrates the construction of the FGC-BPRE scheme. Section four demonstrates the scheme is CCA-safe in stand model. Finally, the text concludes Section five.

2 Preliminaries

2.1 Bilinear mapping

Let G and G_T be two multiplicative cyclic groups. G and G_T have the same prime order p. g is a generator of group G. A bilinear map $e: G \times G \rightarrow G_T$ satisfying the following conditions:

1. $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \overset{R}{\leftarrow} Z_p^*$ and $g_1, g_2 \in G$.
2. $e(g, g) \neq 1$.
3. $e(g_1, g_2)$ can be computed in polynomial time for all $g_1, g_2 \in G$.

2.2 The n-BDHE assumption

Z_p denotes the set $\{0,1, \dots, p-1\}$ and Z_p^* denotes the set $\{1,2, \dots, p-1\}$. Set prime p. Let $e: G \times G \rightarrow G_T$ be a bilinear map. Given $2n+2$ elements:

$$\left(h, g, g^{\alpha}, g^{\alpha^2}, \dots, g^{\alpha^n}, g^{\alpha^{n+2}}, \dots, g^{\alpha^{2n}}, T\right) \in G^{2n+1} \times G_T$$

The adversary needs to decide if $T \overset{?}{=} e(g, h)^{\alpha^{n+1}}$.

Use g_i to indicate g^{α^i}. The davantage of an adversary $\mathcal{A}$ is:

$$Adv_{G,\mathcal{A}}^{n-BDHE} = \left| \begin{matrix} \Pr\left[\mathcal{A}\left(h, g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, e(g_{n+1}, h)\right)\right] = 1 \\ - \Pr[\mathcal{A}(h, g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, T)] = 1 \end{matrix} \right|$$

where $g, h \in G, \alpha \in Z_p^*$ and $T \in G_T$ are randomly chosen. The n-BDHE assumption holds, if $Adv_{G,\mathcal{A}}^{n-BDHE}$ is negligible for all probability polynomial time adversary $\mathcal{A}$.

2.3 Model and security notion

This section describes the definition of fine-grained conditional proxy broadcast re-encryption and their security models. The scheme uses an access tree to describe an access policy, where each internal node is a threshold gate and the leaves are associated with a keyword. Using this approach, we can render trees with "AND" and "OR" gates by using n-out-of-n and 1-out-of n threshold gates, respectively. When the ciphertext satisfies the condition of the tree node, the system will be able to re-encrypt the ciphertext using the re-encryption key that provides the user.

Definition 1 (Access Tree). Let $\mathcal{T}$ represent the access tree. The non-leaf nodes of the tree represent threshold gates, and the child nodes represent thresholds. Let num_x be the number of children of node x, k_x is its threshold and get $0 < k_x \leq num_x$. When $k_x = num_x$, it is an AND gate. When $k_x = 1$, the threshold is an OR gate. Each leaf node x of the tree is described by a conditional keyword and a threshold $kx = 1$. Let the depth of the root node of the access tree be 0. Let $\overline{LN_{\mathcal{T}}}$ be the set of all non-leaf nodes, and $LN_{\mathcal{T}}$ is the set of all leaf nodes. The tree $p(x)$ represents the parent of node x. The function $keyword(x)$ is defined only when $x \in LN_T$, which represents the condition key associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines the ordering between the child nodes of each node, that is, the child nodes of the node are numbered from 1 to num. The function $index(x)$ returns the index associated with node x, each index value representing a unique node in the access tree.

Definition 2 (Satisfying an Access Tree). Let $\mathcal{T}$ be the access tree with root x, and $\mathcal{T}_x$ is the subtree rooted at node x. Then $\mathcal{T}$ and $\mathcal{T}_x$ are the same. $\mathcal{T}_x(W) = 1$ indicates that a set of conditions W satisfies the access tree $\mathcal{T}_x$. If $x \in LNT$,then $\mathcal{T}_x(W) \; = \; 1$ if and only if the

$keyword(x) \in W$. If $x \in \overline{LN_{\mathcal{T}}}$, evaluate $\mathcal{T}_z(W)$ for all children z of node x, $\mathcal{T}_x(W)$ return 1 if and only if at least k_x children return 1.

Definition 3 (FGC-PBRE). The final grain conditional proxy broadcast re-encryption scheme consists the following algorithms.

- $Setup(\lambda, n)$: Input the security parameter λ, maximum the number of users n, output the public key PK and the master secret key msk.
- $Extract(PK, msk, i)$: Input the public key PK, the master key msk, user $i \in \{1,2,\dots,n\}$, output user i's secret key sk_i.
- $Encrypt(PK, S, m, W)$: Input the public key PK, a user set $S \subseteq \{1,2,\dots,n\}$, message m and a descriptive keyword set W, output the ciphertext C for user set S with the condition set W.
- $RKGen(PK, sk_i, S', \mathcal{T})$: Input the public key PK, user i's secret key sk_i, a user set $S' \subseteq \{1,2,\dots,n\}$ and an access tree $\mathcal{T}$, output the re-encryption key $rk_{i \to S', \mathcal{T}}$.
- $ReEnc\left(PK, rk_{i \to S', \mathcal{T}}, i, S, S', C\right)$: Input the public key PK, re-encryption key $rk_{i \to S', W'}$, user i, two user sets S, S', an original ciphertext C. If $\mathcal{T}(W) = 1$, output the re-encrypted ciphertext C_R, or an error symbol ⊥.
- $Decrypt_O(PK, sk_i, i, S, C)$: $Decrypt_O$ is used to decrypt the ciphertext. Input the public key PK, user i's private key sk_i, user i, a user set S and ciphertext C for user set S. Output the plaintext m, or an error symbol ⊥.
- $Decrypt_R(PK, sk_j, i, j, S, S', C_R)$: $Decrypt_R$ is used to decrypt the re-encrypted ciphertext. Input the public key PK, user j's private key sk_j, two users i, j, two user sets S, S', and re-encrypted ciphertext C_R. Output the plaintext m, or an error symbol ⊥.

Consistency. The affirmative determination of the FGC-PBRE scheme is defined as given two users $i \in S, j \in S'$, two user set S, S' , condition sets W, W' , $C = Encrypt(PK, S, m, W)$, $rk_{i \to S', \mathcal{T}} = RKGen(PK, sk_i, S', \mathcal{T})$ and $C_R = ReEnc\left(PK, rk_{i \to S', \mathcal{T}}, i, S, S', C\right)$:

$$\Pr[Decrypt_O(PK, sk_i, i, S, C) = m] = 1, \text{ if } i \in S;$$

$$\Pr\left[Decrypt_R\left(PK, sk_j, i, j, S, S', C_R\right) = m\right] = 1, \text{ if } j \in S', \mathcal{T}(W) = 1.$$

Definition 4 (IND-CCA game). Let $\mathcal{A}$ be an adversary and $\mathcal{C}$ be a challenger. Considering the two security games.

Game 1. IND-O-CCA game proves the security of the original ciphertexts.

1. Init. The adversary $\mathcal{A}$ chooses a target user set $S^* \subseteq \{1,2,\dots,n\}$ and condition set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
2. Setup. The challenger $\mathcal{C}$ runs $Setup(n)$ to obtain public key PK and master key msk, and give PK to $\mathcal{A}$.
3. Query Phase I. The adversary $\mathcal{A}$ submits the following queries:

- $Extract(i)$: The challenger runs $sk_i = KeyGen(PK, msk, i)$, returns sk_i to $\mathcal{A}$.
- $RKGen(i, S', \mathcal{T})$: The challenger runs $rk_{i \to S', \mathcal{T}} = RKGen(PK, sk_i, S', \mathcal{T})$, where $sk_i = KeyGen(PK, msk, i)$, returns $rk_{i \to S', \mathcal{T}}$ to $\mathcal{A}$.
- $ReEnc(i, S, S', C)$: The challenger runs $ReEnc(PK, rk_{i \to S', \mathcal{T}}, i, S, S', C)$, where $rk_{i \to S', \mathcal{T}} = RKGen(PK, sk_i, S', \mathcal{T})$, $sk_i = KeyGen(PK, msk, i)$. Returns the result to $\mathcal{A}$.

- $Decrypt_O(i,S,C)$: The challenger runs $Decrypt_O(PK,sk_j,i,S,C)$, where $sk_i = KeyGen(PK,msk,i)$, returns the result to $\mathcal{A}$.
- $Decrypt_R(i,j,S,S',C_R)$: the challenger runs $Decrypt_R(PK,sk_j,i,j,S,S',C_R)$, where $sk_j = KeyGen(PK,msk,j)$, returns the result to $\mathcal{A}$.

During Query Phase I, $\mathcal{A}$ is limited to:
- $\mathcal{A}$ is restricted to make $Extract(i)$ for any $i \in S^*$;
- $\mathcal{A}$ s restricted to make $RKGen(i,S',W')$ and $Extract(j)$, where $i \in S^*$, $j \in S'$ and $\mathcal{T} = 1$.

4. Challenge. After Query Phase I is executed, $\mathcal{A}$ outputs two equal length messages m_0, m_1. Challenger $\mathcal{C}$ randomly chooses a bit $b \in \{0,1\}$ and sets the challenge ciphertext $C^* = Encrypt(PK,m_b,S^*,W^*)$. Afterwards, Challenger $\mathcal{C}$ return C^* to adversary $\mathcal{A}$.
5. Query phase II. The steps in Query Phase I and Query Phase II are the same. However, the restrictions in query phase II have changed:
- $\mathcal{A}$ is restricted to make $Extract(i)$ for any $i \in S^*$;
- $\mathcal{A}$ is restricted to make $RKGen(i,S',\mathcal{T})$ and $Extract(j)$, where $i \in S^*$, $j \in S'$ and $\mathcal{T}(W^*) = 1$;
- $\mathcal{A}$ is restricted to make $ReEnc(i,S^*,S',C^*)$ and $Extract(j)$, if $i \in S^*, j \in S'$;
- $\mathcal{A}$ is restricted to make $Decrypt_O(i,S^*,C^*)$ for any $i \in S^*$;
- $\mathcal{A}$ is restricted to make $Decrypt_R(i,j,S^*,S',C_R)$, where $i \in S^*$, $j \in S'$ and $C_R = ReEnc(i,S^*,S',C^*)$.

6. Guess. $\mathcal{A}$ outputs the guess b'. If $b = b'$, the adversary $\mathcal{A}$ wins.

The above adversary $\mathcal{A}$ is an IND-O-CCA adversary. Its advantage is defined as:

$$Adv_{\mathcal{A},n}^{Game_1} = |\Pr[b' = b] - \frac{1}{2}|$$

Game 2. IND-Re-CCA game considers the indistinguishability of the re-encrypted ciphertext. Game 2 considers the security of the re-encrypted ciphertext. A complementary definition of security provides a guarantee that an adversary cannot distinguish re-encrypted ciphertexts. For a one-time use scenario, an attacker can access all re-encryption keys, so re-encrypting oracle becomes useless. And Decrypt2 oracle has become unnecessary.

1. Init. The adversary $\mathcal{A}$ chooses a target user set $S^* \subseteq \{1,2,\dots,n\}$ and condition set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
2. Setup. The challenger $\mathcal{C}$ runs $Setup(n)$ to calculate public key PK and master key msk, and outputs PK to $\mathcal{A}$.
3. Query Phase I. The adversary $\mathcal{A}$ makes the following queries:

- $Extract(i)$: the challenger calculates $sk_i = KeyGen(PK,msk,i)$, returns sk_i to $\mathcal{A}$. $\mathcal{A}$ cannot make $Extract(i)$, where $i \in S^*$;
- $RKGen(i,S',\mathcal{T})$: the challenger runs $rk_{i\to S',\mathcal{T}} = RKGen(PK,sk_i,S',\mathcal{T})$, where $sk_i = KeyGen(PK,msk,i)$, returns $rk_{i\to S',\mathcal{T}}$ to $\mathcal{A}$.
- $DecryptR(i,j,S,S',C_R)$: the challenger runs $DecryptR(PK,sk_j,i,j,S,S',C_R)$, where $sk_j = KeyGen(PK,msk,j)$, returns the result to $\mathcal{A}$.

4. Challenge. Once Query Phase I is over, adversary $\mathcal{A}$ outputs an equal length message (m_0, m_1). Challenger $\mathcal{C}$ randomly chooses a bit $b \in \{0,1\}$ and sets the challenge ciphertext

to be $C^* = ReEnc(PK, rk_{i\to S^*,\mathcal{T}}, i, S, S^*, C)$, where $i \in S, i \notin S^*$ and $C = Encrypt(PK, m_b, S, W^*), \mathcal{T} = 1$. Finally, return C^* to the adversary $\mathcal{A}$.

5. Query Phase II. $\mathcal{A}$ continues making queries in Query Phase I with the following restrictions:

- $\mathcal{A}$ cannot make $Extract(i)$ for any $i \in S^*$;
- $\mathcal{A}$ cannot make $DecryptR(i, j, S, S^*, C^*)$, if $i \in S$ and $j \in S^*$.

6. Guess. $\mathcal{A}$ outputs the guess b'. If $b' = b$, the adversary $\mathcal{A}$ wins.

Referring to the above adversary $\mathcal{A}$ as an IND-Re-CCA adversary. Its advantage is defined as:

$$Adv_{\mathcal{A},n}^{Game_2} = |\Pr[b' = b] - \frac{1}{2}|$$

If for all PPT adversary $\mathcal{A}, Adv_{\mathcal{A},n}^{Game_1}$ and $Adv_{\mathcal{A},n}^{Game_2}$ are negligible, fine-grained conditional proxy broadcast re-encryption scheme is IND-sSet-CCA secure.

3 Proposed FGC-PBRE Scheme

3.1 FGC-PBRE Overview

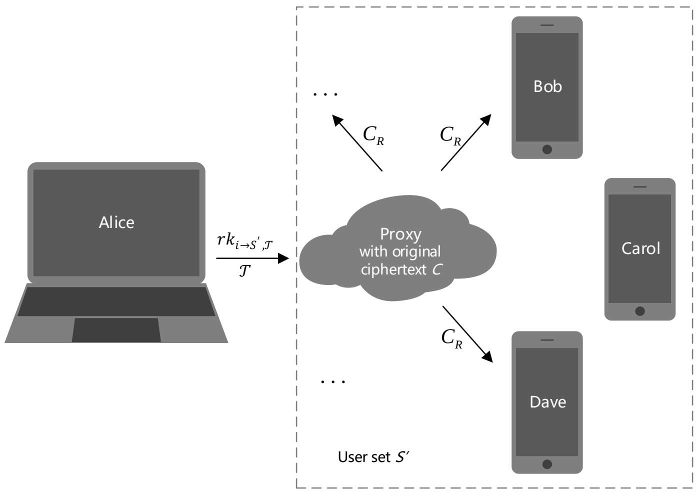

Fig. 1. FGC-BPRE framework

The framework of fine-grained conditional proxy broadcast re-encryption is shown as **Figure 1**. Alice wants to share the encrypted file that has been saved on the server to someone else. At this point, she needs to generate the conversion key $rk_{i\to S',\mathcal{T}}$ and send it to the server together with multiple condition tree $\mathcal{T}$. At this point, the server will separately verify whether the users Bob, Carol, Dave, etc. in the group S' satisfy the forwarding conditions set by Alice.

If Bob, Dave meets the condition, the server will forward the re-encrypted ciphertext that Bob and Dave can decrypt directly. The proxy will not send the re-encrypted ciphertext to users who do not meet the criteria, such as Carol. Multiple conditions provide good control over the receiving permissions of individual users in a user group. In this process, the agent cannot get any content from Alice's ciphertext.

3.2 FGC-PBRE Construction

Define the Lagrange coefficient $\Delta_{\omega,F(x)}$, where $\omega \in Z_p$ and a set F of elements in Z_p:

$$\Delta_{\omega,F}(x) = \prod_{i \in F, i \neq \omega} \frac{x-i}{\omega - i}$$

FGC-PBRE consists of the following algorithms:

- $Setup(\lambda, n)$: Construct a bilinear map parameter (p, g, G, G_T, e) and message $M = \{0,1\}^k$. Randomly choose $\alpha, \gamma \in Z_p$, $Z \in G$ and $g_i = g^{\alpha^i}$ for $i = 1,2,\dots,n, n+2,\dots,2n$. Set $H_\alpha: Z_p^* \to G$, $H_\beta: \{0,1\}^k \to Z_p^*$ as collusion resistant hash functions. Compute $v = g^\gamma$. Output the public key PK and the master secret key msk as:

$$PK = \left(g, g_1, \dots, g_n, g_{n+2}, g_{2n}, v, Z, H_\alpha, H_\beta\right), \qquad msk = \gamma$$

- $KeyGen(PK, msk, i)$: User i's private key is:

$$sk_i = g_i^\gamma$$

- $Encrypt(PK, S, m, W)$: $Encrypt$ is used to encrypt a message $m \in M$ for user set $S \subseteq \{1,2,\dots,n\}$ with the condition set W. Picks a random $\sigma \in G_T$, $t \in Z_p^*$. Output the ciphertext $C = (C_1, C_2, C_3, C_4, C_5, S)$.

$$C_1 = \sigma \cdot e(g_1, g_n)^t, C_2 = g^t, C_3 = \left(v \cdot \prod_{j \in S} g_{n+1-j}\right)^t$$

$$C_4 = \left(H_{\alpha(\omega)}\right)^t, C_5 = [PRF(\sigma, C_2)^{K-k} || ([PRF(\sigma, C_2)]_k \oplus m)$$

$$\mathcal{G}(\lambda) \to (svk, ssk), S = \mathcal{S}\left(ssk, (C_2, C_4, C_5)\right)$$

- $RKGen(PK, sk_i, S', \mathcal{T})$: Input $sk_i = g_i^\gamma$, $S' \in \{1,2,\dots,n\}$ and $\mathcal{T}$. Randomly select $\sigma \in \{0,1\}^k$ and a polynomial q_x for each non-leaf node x in the tree $\mathcal{T}$. Starting from the root node r, the program $RKG\left(\mathcal{T}, H_\beta(\sigma)\right)$ is selected from top to bottom to select the polynomial. The program $RKG\left(\mathcal{T}, H_\beta(\sigma)\right)$ is defined as follows: For each node x in the tree, the degree of the polynomial q_x is set to $d_x = k_x - 1$. Set the degree of root $q_r(0) = H_\beta(\sigma)$. For any other node x, set $q_x(0) = q_p(x)\ \big(\text{index(x)}\big)$ and randomly select d_x to fully define q_x. Once the polynomial is determined, for each leaf node x, set $\omega = keyword(x)$.
  Therefore, the re-encryption key for the agent is calculated as follows:
  Select a random value $r_x \overset{R}{\leftarrow} Z_p^*$, computes:

$$A_x = sk_i \cdot Z^{q_x(0)} \cdot H_\alpha(\omega)^{r_x}; B_x = g^{r_x}; x \in LN_{\mathcal{T}}$$

Chooses random value $t' \in Z_p^*$, $r' \in G_T$, $R \in \{0,1\}^k$ and sets:

$$rk_1 = R' \cdot e(g_1, g_n)^{t'}, rk_2 = g^{t'}, rk_3 = \left(v \cdot \prod_{j \in S'} g_{n+1-j}\right)^{t'}$$

$$rk_4 = [PRF(\sigma', rk_2)^{K-k} || ([PRF(\sigma', rk_2)]_k \oplus R)$$

$$\mathcal{G}(\lambda) \to (svk', ssk'), S' = \mathcal{S}(ssk', (rk_2, rk_4))^{t'}$$

Output the re-encryption key:

$$rk_{i \to S', \mathcal{T}'} = (\mathcal{T}, A_x, B_x, rk_1, rk_2, rk_3, rk_4, S').$$

- $ReEnc(PK, rk_{i \to S', \mathcal{T}'}, i, S, S', C)$: Input a re-encryption key $rk_{i \to, S', W'}$ and a ciphertext C. Check whether the following equalities hold:

$$e\left(C_2, v \cdot \prod_{j \in S} g_{n+1-j}\right) \stackrel{?}{=} e(g, C_3) \tag{1}$$

$$\mathcal{V}\left(svk, S, (C_2, C_4, C_5)\right) \stackrel{?}{=} 1 \tag{2}$$

$$e\left(C_2, H_\alpha(\mathcal{T})\right) \stackrel{?}{=} e(g, C_4) \tag{3}$$

If one of the above equations does not holds, output $\perp$. Otherwise, define a recursive algorithm $NodeReEnc(C, rk_{i \to S', \mathcal{T}'}, x)$ that takes as input the original ciphertext C, the re-encryption key$rk_{i \to S', \mathcal{T}'}$ and the note x in the tree.

1. For leaf node x, if $\omega \in W$, let $\omega = keyword(X)$, then

$$NodeReEnc\left(C, rk_{i \to S', \mathcal{T}'}, x\right) = \frac{e(C_2, A_x)}{e(B_x, C_4)} = \frac{e\left(g^t, sk_i \cdot Z^{q_x(0)} \cdot H_\alpha(\omega)^{r_x}\right)}{e(g^{r_\omega}, H_\alpha(\omega)^t)}$$
$$= e(sk_i, g^t) \cdot e(Z, g)^{t \cdot q_x(0)}$$

   Otherwise, output $\perp$.

2. If x is a non-leaf node, recursively call $NodeReEnc(C, rk_{i \to j, \mathcal{T}'}, z)$ on all child nodes z of x and store the output as $\mathcal{T}_z$. Let F_x be an arbitrary k_x-sized set of child notes z, makes the $\mathcal{T}_z \neq \perp$. If there is no such set, the node is not satisfied and the function $NodeReEnc(C, rk_{i \to S', \mathcal{T}'}, x)$ returns $\perp$. Otherwise, let $F_x' = \{index(z) : z \in S\}$ and calculate:

$$T_x = \prod_{z \in F_x, i = index(z)} (T_z)^{\Delta_{i, F_x'}(0)}$$
$$= \prod_{z \in F_x, i = index(z)} \left(e(sk_i, g^t) \cdot e(Z, g)^{t \cdot q_x(0)}\right)^{\Delta_{i, F_x'}(0)}$$
$$= e(sk_i, g^t) \cdot \prod_{z \in F_x, i = index(z)} \left(e(Z, g)^{t \cdot q_{p(z)} index(z)}\right)^{\Delta_{i, F_x'}(0)}$$

$$= e(sk_i, g^t) \cdot \prod_{z \in F_x, i = index(z)} \left(e(Z, g)^{t \cdot q_x(i)}\right)^{\Delta_{i, F'_x}(0)}$$
$$= e(sk_i, g^t) \cdot e(Z, g)^{sq_x(0)}$$

In the end, computes:

$$\widetilde{C_1} = C_1 \cdot \frac{e\left(T_r \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_2\right)}{e(g_i, C_3)}$$

The re-encrypted ciphertext is $C_R = \left(svk, \widetilde{C_1}, C_2, C_4, C_5, S, rk_1, rk_2, rk_3, rk_4, S'\right)$.

- $DecryptO(PK, sk_i, i, S, C)$: Input a private key sk_i and an original ciphertext $C = (C_1, C_2, C_3, C_4, C_5, S)$, proceeds as follows:

1. Checks whether the equations (1) ~ (3) hold. If one of the above equations does not holds, output $\perp$ the abort.
2. Computes $\sigma = C_1 \cdot e\left(sk_i, \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_2\right) / e(g_i, C_3)$. If $PRF[\sigma, C_2]^{K-k} = [C_5]^{K-k}$ hold, returns $m = PRF[\sigma, C_2]_k \oplus [C_5]_k$. Else return $\perp$ and abort.

- $DecryptR(PK, sk_j, i, j, S, S', C_R)$: Input a private key sk_j and a re-encrypted ciphertext C_R, proceeds as follows:

1. Checks the equations:

$$e\left(rk_2, v \cdot \prod_{j \in S} g_{n+1-j}\right) \overset{?}{=} e(g, rk_3) \qquad (4)$$

$$\mathcal{V}\left(svk', S', (rk_2, rk_4)\right) \overset{?}{=} 1 \qquad (5)$$

If one of these equations does not holds, output $\perp$ and abort.

2. Calculate $\sigma' = rk_1 \cdot e\left(sk_j \cdot \prod_{l \in S', l \neq j} g_{n+1-l+j}, rk_2\right) / e\left(g_j, rk_3\right)$, if $PRF[\sigma', rk_2]^{K-k} = [rk_4]^{K-k}$, output $R = PRF[\sigma', rk_2]_k \oplus [rk_4]_k$. If not, returns $\perp$ and abort.
3. Calculate $\sigma = \widetilde{C_1} / e\left(C_2, Z^{H_\beta(R)}\right)$. If $PRF[\sigma, C_2]^{K-k} = [E]^{K-k}$, output $m = PRF[\sigma, C_2]_k \oplus [C_5]_k$. Else, returns $\perp$ and abort.

Consistency.

1. If $C = (svk, C_1, C_2, C_3, C_4, C_5, S)$ is an original ciphertext, then:

$$C_1 \cdot \frac{e\left(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, C_2\right)}{e(g_i, C_3)}$$
$$= \sigma \cdot e(g_1, g_n)^t \cdot \frac{e\left(g_i^\gamma \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t\right)}{e\left(g_i, g^\gamma \cdot \prod_{j \in S} g_{n+1-j}\right)^t}$$
$$= \sigma \cdot e(g_1, g_n)^t \cdot \frac{e\left(g^t, \prod_{j \in S, j \neq i} g_{n+1-j+i}\right)}{e\left(g^t, \prod_{j \in S} g_{n+1-j+i}\right)}$$
$$= \sigma \cdot \frac{e(g_1, g_n)^t}{e(g^t, g_{n+1})}$$
$$= \sigma$$

2. If $C_R = (svk, \widetilde{C_1}, C_2, C_4, S, svk', rk_1, rk_2, rk_3, rk_4, rk_5, S')$ is a re-encrypted ciphertext, then:

$$\widetilde{C_1} = C_1 \cdot \mathcal{T}_r \cdot \frac{e\left(\prod_{j\in S, j\neq i} g_{n+1-j+i}, C_2\right)}{e(g_i, C_3)}$$
$$= \sigma \cdot e(g_1, g_n)^t \cdot e(sk_i, g^t) \cdot e(Z, g)^{sq_x(0)} \cdot \frac{e\left(\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t\right)}{e\left(g_i, v \cdot \prod_{j\in S} g_{n+1-j}\right)^t}$$
$$= \sigma \cdot e(g^t, Z)^{H_\beta(R)}$$

It is finally possible to calculate:

$$\frac{\widetilde{C_1}}{e\left(C_2, Z^{H_\beta(R)}\right)} = \sigma$$

4 Proof of security

This section will demonstrate the IND-CCA security of the scheme. The analysis of the security game is as follows.

Theorem 1. If H_α, H_β are target collision resistant hash functions, then the FGC-BPRE scheme is IND-CCA secure without random oracles under the Decisional n-BDHE assumption.

Lemma 1. If there exists an IND-O-CCA adversary $\mathcal{A}$ that can break the FGC-BPRE scheme, may wish to construct a simulator $\mathcal{B}$ that can solve Decisional n-BDHE assumption.

Proof. Given a Decisional n-BDHE instance $(h, g, g_1, \dots g_n, g_{n+2}, \dots, g_{2n}, T)$, $\mathcal{B}$ has decide whether $T = e(g_{n+1}, h)$ or T is a random element in G_T.

$\mathcal{B}$ maintains the following initial empty table:

- Key^{List}: records the tuples (β, i, sk_i), which are the information of secret keys;
- $ReKey^{List}$: stores the information generated by $RKGen(sk_i, S', W')$ in tuple $(\beta_1, i, S', W', rk_{i\to S',\mathcal{T}}, \sigma, R, flag_1)$. $flag_1 = 1$ indicates that the re-encryption key is a valid key, and $flag_1 = 0$ indicates that the re-encryption key is a random value.

1. **Init.** The adversary $\mathcal{A}$ selects a challenge user set $S^* \subseteq \{1,2,\dots,n\}$ and condition set $W^* = \{\omega_1^*, \omega_2^*, \dots, \omega_n^*\}$.
2. **Setup.** The simulator $\mathcal{B}$ pick up a random value $\mu \in Z_p^*$, $Z \in G$ and sets:

$$v = g^\mu \cdot \left(\prod_{j\in S^*} g_{n+1-j}\right)^{-1} \triangleq g^\gamma$$

$\mathcal{B}$ sets the public key as $PK = (g, g_1, \dots, g_n, g_{n+2}, \dots, g_{2n}, v, Z, H_\alpha, H_\beta)$ and $sk = \gamma$, giving PK to $\mathcal{A}$.

3. **Query Phase I.** $\mathcal{B}$ responds to the following questions raised by $\mathcal{A}$:

- $Extract(i)$: $\mathcal{B}$ first confirm $i \notin S^*$. Then, $\mathcal{B}$ searches Key^{List}, if (β, i, sk_i) exists in Key^{List}, returns sk_i to $\mathcal{A}$. Otherwise, $\mathcal{B}$ generates a biased coin β , where $\Pr[\beta = 1] = \delta$.

- If $\beta = 0$, $\mathcal{B}$ outputs a random bit and aborts.
- If $\beta = 1$, $\mathcal{B}$ computes $sk_i = g_i^{\mu} \cdot \left(\prod_{j\in S^*} g_{n+1-j+i}\right)^{-1}$. Then

$$\begin{aligned} sk_i &= g_i^{\mu} \cdot \left(\prod_{j\in S^*} g_{n+1-j}\right)^{-1} \\ &= \left(g^{\mu} \cdot \left(\prod_{j\in S^*} g_{n+1-j}\right)^{-1}\right)^{\alpha^i} \\ &= v^{\alpha^i} \\ &= g_i^{\gamma} \end{aligned}$$

- $RKGen(i, S', \mathcal{T})$: Set $i \in S^*$, $j \in S^*$, and $\mathcal{T}(W^*) = 1$. $\mathcal{B}$ checks that there is no tuple $(*, j, sk_j)$ in Key^{List}, where $*$ is a wildcard. If such an entry exists, $\mathcal{B}$ aborts. Else if there is a tuple $(*, i, S', \mathcal{T}, rk_{i\to S',\mathcal{T}}, \sigma, R, *)$ in $ReKey^{List}$, $\mathcal{B}$ returns $rk_{i\to S',\mathcal{T}}$ Else, $\mathcal{B}$ proceeds as follows:

- If $(1, i, sk_i)$ exists in Key^{List}, $\mathcal{B}$ uses sk_i to generate the re-encryption key $rk_{i\to S',\mathcal{T}}$ visa algorithm $RKGen$ as in the real scheme. Returns the re-encryption key to $\mathcal{A}$ and adds $(*, i, S', \mathcal{T}, rk_{i\to S',\mathcal{T}}, \sigma, R, 1)$ to $ReKey^{List}$, where r', R are randomly chosen in the $RKGen$ algorithm.

- Otherwise, $\mathcal{B}$ flips a biased coin $\mathcal{B}$. If $\beta = 1$, $\mathcal{B}$ queries the $Extract(i)$ to get sk_i, and then generates $rk_{i\to S',\mathcal{T}}$ visa algorithm $RKGen$, returning the re-encryption key $rk_{i\to S',\mathcal{T}}$ to $\mathcal{A}$, then adds $(1, i, sk_i)$ and $(*, i, S', \mathcal{T}, rk_{i\to S',\mathcal{T}}, \sigma, R, 1)$ to Key^{List} and $ReKey^{List}$. If $\beta = 0$, $\mathcal{B}$ sets $\{(A_\omega = \rho_\omega), (B_\omega = \rho'_\omega); \omega \in keyword(\omega)\}$ for randomly chosen $\rho_\omega, \rho'_\omega \in G$. Then $\mathcal{B}$ constructs rk_1, rk_2, rk_3, rk_4 and pick up σ', R. In the end, $\mathcal{B}$ forwards the re-encryption key to $\mathcal{A}$ and adds $(*, i, S', \mathcal{T}, rk_{i\to S',\mathcal{T}}, \sigma, R, 0)$ to $ReKey^{List}$.
- $ReEnc(i, S, S', C)$: $\mathcal{B}$ performs the following operations:

- If $(*, i, S', \mathcal{T}, rk_{i\to S',\mathcal{T}}, \sigma, R, *)$ exists in $ReKey^{List}$, $C = Encrypt(PKm, S, m, W), \mathcal{T}(W) = 1$, $\mathcal{B}$ uses $rk_{i\to S',\mathcal{T}}$ to generate C_R visa algorithm $ReEnc$. Then, $\mathcal{B}$ returns C_R to $\mathcal{A}$ and adds $(i, S, S', C, C_R, *)$ to $ReEnc^{List}$.
- Otherwise, $\mathcal{B}$ issues $RKGen(i, S')$ query to obtain re-encryption key $rk_{i\to S',\mathcal{T}}$. Then $\mathcal{B}$ generates C_R and adds $(i, S, S', C, C_R, *)$ to the $ReEnc^{List}$.

- $Decrypt_O(i, S, C)$: $\mathcal{B}$ verifies whether (1)-(3) is established. If the equations do not hold, output $\perp$. Otherwise proceeds:

- If $(1, i, sk_i)$ exists in Key^{List}, using sk_i to recover m.
- Otherwise, $\mathcal{B}$ issues $Extract(i)$ query to obtain sk_i and uses sk_i to recover m.

- $Decrypt_R(i, j, S, S', C_R)$: $\mathcal{B}$ verifies whether (4)-(5) is established. If the above formula is not true, output $\perp$ and abort. Otherwise, $\mathcal{B}$ proceeds:

- If $(1, j, sk_j)$ exists in Key^{List}, using sk_j to recover m.
- Otherwise, $\mathcal{B}$ runs $Extract(j)$ to obtain sk_j and uses sk_j to recover m.

4. **Challenge.** When $\mathcal{A}$ decides Query Phase I is over, it outputs two equal length message m_0, m_1. $\mathcal{B}$ randomly chooses $b \in \{0,1\}, r^* \in G_T$. Let $h = g^{t^*}$for some randomly chosen t^*. $\mathcal{B}$ computes:

$$
\begin{aligned}
C_1^* &= \sigma^* \cdot T \\
C_2^* &= h = g^{t^*} \\
C_3^* &= h^{\mu} = g^{\mu t^*} \\
&= \left(g^{\mu} \cdot \left(\prod_{j \in S^*} g_{n+1-j} \right)^{-1} \left(\prod_{j \in S^*} g_{n+1-j} \right) \right)^{t^*} \\
&= \left(v \cdot \prod_{j \in S^*} g_{n+1-j} \right)^{t^*} \\
C_4^* &= H_{\alpha}(\omega)^{t^*}, \omega \in W^* \\
C_5^* &= [PRF(\sigma^*, C_2^*)]^{K-k} || ([PRF(\sigma^*, C_2^*)]_k \oplus m_b) \\
\mathcal{G}(\lambda) &= (ssk^*, svk^*) \\
S^* &= \mathcal{S}\big(svk^*, (C_2^*, C_4^*, C_5^*)\big)
\end{aligned}
$$

If $T = e(g_{n+1}, h)$, $C_1^* = \sigma^* \cdot T = \sigma^* \cdot e(g, g_{n+1})^{t^*}$. We can draw C_3^* is a valid challenge ciphertext. If T is a random value in G_T, C_3^* is independent of b in the adversary's view.

5. **Query Phase II.** $\mathcal{A}$ continues making queries as in Query Phase I with the restrictions described in the IND-O-CCA game.
6. **Guess.** $\mathcal{A}$ outputs the guess b', if $b' = b$, then output 1 meaning $T = e(g_{n+1}, h)$; else output 0 meaning T is a random value in G_T.

The above steps complete the proof of Lemma 1.

Lemma 2. If there exists an IND-Re-CCA adversary $\mathcal{A}$ that can break the FGC-BPRE scheme, a simulator $\mathcal{B}$ that can solve Decisional n-BDHE assumption.
Proof. In addition to the challenge phase, the proof of Lemma 2 is like Lemma 1. The challenger constructs the challenge re-encrypted ciphertext in the following manner. $\mathcal{B}$ randomly chooses $b \in \{0,1\}, \sigma^* \in G_T$. Let $h = g^{t^*}$ for some randomly chosen t^*. $\mathcal{B}$ computes:

$$
\begin{aligned}
\widetilde{C_1}^* &= \sigma^* \cdot e(h, Z)^{H_{\beta}(\sigma)} \\
C_2^* &= h = g^{t^*} \\
C_5^* &= [PRF(\sigma^*, C_2^*)]^{K-k} || ([PRF(\sigma^*, C_2^*)]_k \oplus m_b)
\end{aligned}
$$

$\mathcal{B}$ constructs $rk_1^*, rk_2^*, rk_3^*, rk_4^*, rk_5^*$ in the same way as in Game 1.

5 Conclusion

This article describes the concept of fine-grained conditional proxy broadcast re-encryption. If the ciphertext keyword satisfies the conditions in the access tree, then the scheme allows the proxy to convert the user's ciphertext into a new set of users' ciphertext. In addition, this paper also constructs a fine-grained conditional proxy broadcast re-encryption scheme and verifies the security of selective ciphertext attacks under the standard model. Fine-grained conditional proxy broadcast re-encryption itself is built on attribute-based encryption and can be applied to, file sharing systems, email systems, and so on.

Acknowledgments. This work is supported by the NSFC (No. 61772280, 61702236, and 61672290), the Changzhou Sci&Tech Program (No.CJ20179027), and the PAPD fund from NUIST. Prof. Chengshan Qian is the corresponding author.

References

[1] Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. International Conference on the Theory and Applications of Cryptographic Techniques. pp. 127-144. (1998).

[2] Mishra, B., Jena, D.: CCA Secure Proxy Re-Encryption Scheme for Secure Sharing of Files through Cloud Storage. 2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT). pp. 1-6. (2018).

[3] Borcea, C., Polyakov, Y., Rohloff, K., Ryan, G.J.F.G.C.S.: PICADOR: End-to-end encrypted Publish–Subscribe information distribution with proxy re-encryption. Future Generation Computer Systems. pp. 177-191 (2017).

[4] Ge, C., Susilo, W., Fang, L., Wang, J., Shi, Y.J.D., Codes, Cryptography: A CCA-secure key-policy attribute-based proxy re-encryption in the adaptive corruption model for dropbox data sharing system. Designs, Codes and Cryptography. pp. 2587-2603 (2018).

[5] Weng, J., Chen, M., Yang, Y., Deng, R., Chen, K., Bao, F.J.S.C.I.S.: CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. Science China Information Sciences. pp. 593-606 (2010).

[6] Huang, Q., Yang, Y., Fu, J.: PRECISE: Identity-based private data sharing with conditional proxy re-encryption in online social networks. Future Generation Computer Systems. pp. 1523-1533 (2018).

[7] Chu, C.-K., Weng, J., Chow, S.S., Zhou, J., Deng, R.H.: Conditional proxy broadcast re-encryption. Australasian Conference on Information Security and Privacy. pp. 327-342. (2009).

[8] Ateniese, G., Fu, K., Green, M., Hohenberger, S.J.A.T.o.I., Security, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Transactions on Information and System Security (TISSEC). pp. 1-30 (2006).

[9] Weng, J., Chen, M., Yang, Y., Deng, R., Chen, K., Bao, F.J.S.C.I.S.: CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. Science China Information Sciences. pp. 593-606 (2010).

[10] Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption without pairings. International Conference on Cryptology and Network Security. pp. 1-17. (2008).

[11] Weng, J., Deng, R.H., Ding, X., Chu, C.-K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. pp. 322-332. (2009).

[12] Fang, L., Susilo, W., Wang, J.: Anonymous conditional proxy re-encryption without random oracle. International Conference on Provable Security. pp. 47-60. (2009).

[13] Luo, S., Hu, J., Chen, Z.: Ciphertext policy attribute-based proxy re-encryption. International Conference on Information and Communications Security. pp. 401-415. (2010).

[14] Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. Proceedings of the 13th ACM conference on Computer and communications security. pp. 89-98. (2006).

[15] Weng, J., Chen, M., Yang, Y., Deng, R., Chen, K., Bao, F.: CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. Science China Information Sciences. pp.593-606. (2010)