Singapore Management University
Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems School of Information Systems

7-2009

Conditional Proxy Broadcast Re-Encryption

Cheng-Kang CHU
Singapore Management University

Jian Weng
Singapore Management University

Sherman Chow

Jianying Zhou
Singapore Management University

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

DOI: https://doi.org/10.1007/978-3-642-02620-1_23

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

Part of the Information Security Commons

Citation

CHU, Cheng-Kang; Weng, Jian; Chow, Sherman; Zhou, Jianying; and DENG, Robert H.. Conditional Proxy Broadcast Re-Encryption. (2009). *Information Security and Privacy: 14th Australasian Conference, ACISP 2009 Brisbane, Australia, July 1-3, 2009: Proceedings*. 5594, 327-242. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/288

This Conference Proceeding Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Conditional Proxy Broadcast Re-Encryption*

Cheng-Kang Chu[1], Jian Weng[1,2],
Sherman S.M. Chow[3], Jianying Zhou[4], and Robert H. Deng[1]

[1] School of Information Systems
Singapore Management University, Singapore
{ckchu,jianweng,robertdeng}@smu.edu.sg
[2] Department of Computer Science
Jinan University, Guangzhou 510632, P.R. China
[3] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[4] Institute for Infocomm Research, Singapore
jyzhou@i2r.a-star.edu.sg

Abstract. A proxy re-encryption (PRE) scheme supports the delegation of decryption rights via a proxy, who makes the ciphertexts decryptable by the delegatee. PRE is useful in various applications such as encrypted email forwarding. In this paper, we introduce a more generalized notion of conditional proxy broadcast re-encryption (CPBRE). A CPBRE scheme allows Alice to generate a re-encryption key for some condition specified during the encryption, such that the re-encryption power of the proxy is restricted to that condition only. This enables a more fine-grained delegation of decryption right. Moreover, Alice can delegate decryption rights to a set of users at a time. That is, Alice's ciphertexts can be re-broadcasted. This saves a lot of computation and communication cost. We propose a basic CPBRE scheme secure against chosen-plaintext attacks, and its extension which is secure against replayable chosen-ciphertext attacks (RCCA). Both schemes are unidirectional and proved secure in the standard model. Finally, we show that it is easy to get a unidirectional RCCA-secure identity-based proxy re-encryption from our RCCA-secure CPBRE construction.

Keywords: proxy re-encryption, conditional proxy re-encryption, broadcast encryption, hierarchical identity-coupling broadcast encryption.

1 Introduction

Proxy re-encryption (PRE) schemes enable (by delegating a transformation-key to) a semi-trusted proxy to transform Alice's ciphertext into one encrypting the same message which is decryptable by Bob, without allowing the proxy any ability to perform tasks outside of the delegation. PRE found applications [3,8] in

* Funded by A*STAR project SEDS-0721330047.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 327–342, 2009.
© Springer-Verlag Berlin Heidelberg 2009

digital rights management, distributed file storage systems, and email forwarding. For example, users can assign their email server as the proxy such that it can re-encrypt the emails for different users without knowing the email content.

Although PRE is useful in many applications, we found that sometimes we need more than the basic. In corporate email forwarding, Alice may ask the proxy to re-encrypt her emails to her colleague Bob when she is on leave. However, this is not enough in the following scenarios:

1. Alice does not want Bob to read all her private emails.
2. For some business emails, Alice has to forward them to more than one colleague other than Bob, in an extreme case, the whole staff of the company.

Using a traditional PRE, a proxy is too powerful as it has the ability to re-encrypt *all* Alice's emails to Bob once the re-encryption key is given. For more than one delegatees, Alice needs to generate a re-encryption key for *each* staff member, and the proxy also needs to re-encrypt emails for *each* of them.

We believe there is a better way to handle these situations. We envision a more generalized notion of *Conditional Proxy Broadcast Re-Encryption* (CP-BRE). Alice can specify a condition to generate a *conditional* re-encryption key, such that the re-encryption power of the proxy is restricted to that condition only. Moreover, Alice can delegate the decryption rights to a set of users at a time, which means Alice's ciphertexts can be *re-broadcasted*. In this paper, we formalize this notion, propose a basic CPBRE scheme secure against chosen-plaintext attacks (CPA), and an extension that is secure against replayable chosen-ciphertext attacks (RCCA) [9]. RCCA is a weaker variant of chosen-ciphertext attack (CCA) in which a harmless mauling of the challenge ciphertext is tolerated. Both schemes are unidirectional and secure in the standard model.

The new CPBRE is much more flexible. Back to our email-forwarding example, Alice can use the keywords "business", "private" and "golf" as the conditions, to allow forwarding of her encrypted emails to her colleagues, family members, and golf club members respectively. For each group, Alice only requires to produce one re-encryption key and the proxy only requires to transform a single ciphertext. This saves a lot of computation and communication cost.

Finally, being a generalization of PRE, it is easy to use our RCCA-secure CP-BRE construction to build a RCCA-secure unidirectional identity-based proxy re-encryption (IB-PRE), which is the first of its kind.

1.1 Related Works

Following Blaze, Bleumer and Strauss's seminal work [3] which presented a bidirectional CPA-secure PRE scheme, many PRE schemes have been proposed. Ateniese *et al.* [1] presented a CPA-secure unidirectional PRE. Canetti and Hohenberger [8] presented a CCA-secure bidirectional PRE. Later, Libert and Vergnaud [14] presented a RCCA-secure unidirectional PRE. These PRE schemes [1,8,14] rely on the somewhat costly bilinear pairings. Without pairings, Deng *et al.* [11] proposed a CCA-secure bidirectional PRE. Subsequently, Weng *et al.* [19] and Shao and Cao [16] presented CCA-secure unidirectional PRE.

Proxy re-encryption has also been studied in identity-based encryption (IBE) settings. Based on Boneh-Boyen IBE [4], Boneh, Goh and Matsuo [6] described a hybrid proxy re-encryption system. Green and Ateniese [12] presented a CPA-secure IB-PRE and a CCA-secure IB-PRE, which are proven in the random oracle model and only support single-use (i.e., the ciphertext can only be re-encrypted once). Matsuo [15] also proposed a CPA-secure IB-PRE scheme. Later, Chu and Tzeng [10] tried to propose a CCA-secure multi-use IB-PRE scheme without random oracles. However, as Shao and Cao [16] stated, [12,10] are unable to resist a "chain collusion attack" (described later). Up to now, there is still no CCA-secure (or RCCA-secure) IB-PRE scheme in the standard model.

Tang [17] introduced the primitive of type-based proxy re-encryption, which allows the proxy to re-encrypt a specific type of delegator's ciphertexts. Independently, Weng *et al.* [18] introduced a similar primitive named "conditional proxy re-encryption", in which the proxy can re-encrypt a ciphertext under a specific condition iff he has the re-encryption key with respect to this condition. However, both of these schemes are proved in the random oracle model. Finally, Libert and Vergnaud [13] introduced the notion of traceable proxy re-encryption, where malicious proxies leaking their re-encryption keys can be identified.

2 Definition

We briefly describe the assumptions and underlying encryption schemes that will be used in our constructions, then the definition of CPBRE will be given.

2.1 Pairing and Related Computational Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicatively) cyclic groups of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is a map with the following properties:

- Bilinear: for all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Non-degenerate: for some $g \in \mathbb{G}$, $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the bilinear map are efficiently computable.

Our schemes are based on the Decisional Bilinear Diffie-Hellman Exponent (BDHE) problem in $(\mathbb{G}, \mathbb{G}_T)$ [5] – given $2n+1$ elements

$$(\tilde{g}, g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}) \in \mathbb{G}^{2n+1},$$

and an element $R \in \mathbb{G}_T$, decide if $R = e(g, \tilde{g})^{\alpha^{n+1}}$.

In the rest of this paper, we will use g_i to denote the term g^{α^i}.

Definition 1. *The Decisional n-BDHE assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$ *if no polynomial time algorithm* $\mathcal{A}$ *has non-negligible advantage in solving the Decisional n-BDHE problem in* $(\mathbb{G}, \mathbb{G}_T)$*, where the advantage of* $\mathcal{A}$ *is* ε *if*

$$\left| \begin{array}{r} \Pr[\mathcal{A}(\tilde{g}, g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, e(g_{n+1}, \tilde{g})) = 1] \\ - \Pr[\mathcal{A}(\tilde{g}, g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, R) = 1] \end{array} \right| \geq \varepsilon.$$

2.2 Hierarchical Identity-Coupling Broadcast Encryption

Attrapadung, Furukawa and Imai [2] proposed a notion of Hierarchical Identity-Coupling Broadcast Encryption (HICBE). We review its model and security definition here. For an identity $ID = \{id_1, id_2, \ldots, id_l\}$, we denote ID_j as $\{id_1, id_2, \ldots, id_j\}$ for $1 \leq j \leq l$. An HICBE consists of the following algorithms.

- **Setup**(n): on input the maximum number of users, output the public key PK and master secret key MK.
- **KeyGen**(PK, MK, i): on input the public key PK, the master secret key MK and an index $i \in \{1, 2, \ldots, n\}$, output the root secret key sk_i of user i.
- **Derive**$(PK, sk_{i,ID_{l-1}}, i, ID)$: on input the public key PK, the secret key $sk_{i,ID_{l-1}}$ of user i coupling with the $(l-1)$-level identity ID_{l-1}, an index $i \in \{1, 2, \ldots, n\}$ and an l-level identity ID, output the secret key $sk_{i,ID}$.
- **Encrypt**(PK, S, ID, m): on input the public key PK, an index set $S \subseteq \{1, 2, \ldots, n\}$, an identity ID and a message m, output the ciphertext C.
- **Decrypt**$(PK, sk_{i,ID}, i, S, ID, C)$: on input the public key PK, the secret key $sk_{i,ID}$, an index i, a set S, an identity ID and a ciphertext C, output the plaintext m.

The selective identity-and-set security of HICBE is defined by the following game between an adversary $\mathcal{A}$ and a challenger. Both are given n as input.

1. Init. $\mathcal{A}$ picks a set $S^* \subseteq \{1, 2, \ldots, n\}$ and an identity ID^* to be attacked.
2. Setup. Perform **Setup**(n) to get (PK, MK) and give PK to $\mathcal{A}$.
3. Query phase 1. $\mathcal{A}$ can issue the following queries:
   - Extract(i, ID): if $i \notin S^*$ or ID is not a prefix of ID^* or ID^* itself, return $sk_{i,ID} \leftarrow$ **Derive**(PK, sk_i, i, ID) where $sk_i \leftarrow$ **KeyGen**(PK, MK, i); otherwise, return '$\perp$'.
   - Decrypt(i, S, ID, C): return $m \leftarrow$ **Decrypt**$(PK, sk_{i,ID}, i, S, ID, C)$, where $sk_{i,ID} \leftarrow$ **Derive**(PK, sk_i, i, ID), $sk_i \leftarrow$ **KeyGen**(PK, MK, i).
4. Challenge. $\mathcal{A}$ presents (m_0, m_1). Return $C^* \leftarrow$ **Encrypt**(PK, S^*, ID^*, m_b) to $\mathcal{A}$, where $b \in_R \{0, 1\}$.
5. Query phase 2. $\mathcal{A}$ continues making queries as in Query phase 1 except that $\mathcal{A}$ cannot issue the decryption query on (i, S^*, ID, C^*) such that $i \in S^*$ and $(ID = ID^*$ or ID is a prefix of $ID^*)$.
6. Guess. $\mathcal{A}$ outputs the guess $b' \in \{0, 1\}$.

We say $\mathcal{A}$ wins the game if $b' = b$. The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - \frac{1}{2}|$. An HICBE scheme is IND-sID-sSet-CCA-secure if for any probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in this game is negligible. The CPA security is defined in the same way as CCA security except that $\mathcal{A}$ is not allowed to issue the decryption queries.

Attrapadung, Furukawa and Imai provided two constructions based on BGW [5] broadcast encryption. In this paper, we use the HICBE based on BB-IBE [4] to build our CPBRE constructions. The security of this HICBE can be asserted by the following theorem, details can be found in [7] and the full version of [2].

Theorem 1. *Suppose the Decisional n-BDHE assumption holds. The (BGW+BB) HICBE scheme for n users is IND-sID-sSet-CCA-secure.*

2.3 Conditional Proxy Broadcast Re-Encryption

We define our new notion of conditional proxy broadcast re-encryption as follows.

Definition 2. *A conditional proxy broadcast re-encryption scheme consists of the following algorithms:*

- ***Setup***(n)*: used for the generation of the system public key and master secret key of n users. On input the maximum number of users, output the public key PK and master secret key MK.*
- ***KeyGen***(PK, MK, i)*: used for the generation of user i's secret key. On input the public key PK, the master secret key MK and an index $i \in \{1, 2, \ldots, n\}$, output the secret key sk_i.*
- ***Encrypt***(PK, S, w, m)*: used for the generation of a regular ciphertext of m for the set S under condition w. On input the public key PK, an index set $S \subseteq \{1, 2, \ldots, n\}$, a condition w and a message m, output the ciphertext C.*
- ***RKGen***(PK, sk_i, S', w)*: used for the generation of a re-encryption key from i to S' under condition w. On input the public key PK, the secret key sk_i, an index set S' and a condition w, output the re-encryption key $d_{i \to S'|w}$.*
- ***ReEnc***$(PK, d_{i \to S'|w}, i, S, S', w, C)$*: used for the generation of a re-encrypted ciphertext from C. On input the public key PK, the re-encryption key $d_{i \to S'|w}$, the original recipient i, the original set S, the new set S', the condition w and a ciphertext C, output the re-encrypted ciphertext C_R or '$\perp$'.*
- ***Decrypt-I***(PK, sk_i, i, S, w, C)*: used for the decryption of the regular ciphertext C. On input the public key PK, the secret key sk_i, an index i, a set S, a condition w and a ciphertext C, output the plaintext m or '$\perp$'.*
- ***Decrypt-II***$(PK, sk_{i'}, i, i', S, S', w, C_R)$*: used for the decryption of the re-encrypted ciphertext C_R. On input the public key PK, the delegatee's secret key $sk_{i'}$, two indices i and i', two sets S and S', a condition w and a re-encrypted ciphertext C_R, output the plaintext m or '$\perp$'.*

Correctness. *For any integer n, any sets S and S', any indices $i \in S$ and $i' \in S'$, any condition w and any message m,*

$$\Pr\begin{bmatrix} \textbf{\textit{Decrypt-I}}(PK, sk_i, i, S, w, C) = m : (PK, MK) \leftarrow \textbf{\textit{Setup}}(n), \\ sk_i \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i), C \leftarrow \textbf{\textit{Encrypt}}(PK, S, w, m) \end{bmatrix} = 1,$$

$$\Pr\begin{bmatrix} \textbf{\textit{Decrypt-II}}(PK, sk_{i'}, i, i', S, S', w, C_R) = m : \\ (PK, MK) \leftarrow \textbf{\textit{Setup}}(n), sk_i \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i), \\ sk_{i'} \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i'), d_{i \leftarrow S'|w} \leftarrow \textbf{\textit{RKGen}}(PK, sk_i, S', w), \\ C \leftarrow \textbf{\textit{Encrypt}}(PK, S, w, m), C_R \leftarrow \textbf{\textit{ReEnc}}(PK, d_{i \leftarrow S'|w}, i, S, S', w, C) \end{bmatrix} = 1.$$

Now, we proceed to define the security model for CPBRE. Here we consider the security in the replayable CCA sense [9,8,14]. For traditional public key cryptosystems, in such a relaxed security notion, an adversary who can simply modify a given ciphertext into another encryption of the same plaintext is not deemed successful [14]. To define the RCCA security for CPBRE systems, we here disallow the adversary to ask for a decryption of any re-randomized version of the -encrypted ciphertext re-encrypted from the challenge ciphertext.

Definition 3. *The chosen-ciphertext security of a CPBRE scheme against a static adversary is defined by the following game between an adversary $\mathcal{A}$ and a challenger. Both the challenger and $\mathcal{A}$ are given n as input.*

1. *Init. $\mathcal{A}$ chooses a set $S^* \subseteq \{1, 2, \ldots, n\}$ and a condition w^* that it wants to attack.*
2. *Setup. Perform **Setup**(n) to get (PK, MK) and give PK to $\mathcal{A}$.*
3. *Query phase 1. We define the following oracles.*
   - *(a)* $\textsc{Extract}(i)$*: return $sk_i \leftarrow$ **KeyGen**(PK, MK, i).*
   - *(b)* $\text{RK}\textsc{Extract}(i, S', w)$*: return $d_{i \to S'|w} \leftarrow$ **RKGen**(PK, sk_i, S', w), where $sk_i \leftarrow$ **KeyGen**(PK, MK, i).*
   - *(c)* $\textsc{ReEncrypt}(i, S, S', w, C)$*: return $C_R \leftarrow$ **ReEnc**$(PK, d_{i \to S'|w}, C)$, where $d_{i \to S'|w} \leftarrow$ **RKGen**(PK, sk_i, S', w) and $sk_i \leftarrow$ **KeyGen**(PK, MK, i).*
   - *(d)* $\textsc{Decrypt}\text{-I}(i, S, w, C)$*: return $m \leftarrow$ **Decrypt-I**(PK, sk_i, i, S, w, C), where $sk_i \leftarrow$ **KeyGen**(PK, MK, i).*
   - *(e)* $\textsc{Decrypt}\text{-II}(i, i', S, S', w, C_R)$*: compute $sk_{i'} \leftarrow$ **KeyGen**(PK, MK, i') and return $m \leftarrow$ **Decrypt-II**$(PK, sk_{i'}, i, i', S, S', w, C_R)$.*

   $\mathcal{A}$ can issue these queries except
   - $\textsc{Extract}(i)$ *for any $i \in S^*$ and*
   - *both* $\text{RK}\textsc{Extract}(i, S', w^*)$ *and* $\textsc{Extract}(i')$ *for any S', $i \in S^*$ and $i' \in S'$.*
4. *Challenge. $\mathcal{A}$ presents (m_0, m_1). Return $C^* =$ **Encrypt**(PK, S^*, w^*, m_b) to $\mathcal{A}$, where $b \in_R \{0, 1\}$.*
5. *Query phase 2. $\mathcal{A}$ continues making queries as in the Query phase 1, except for the following queries*
   - $\textsc{Extract}(i)$ *for any $i \in S^*$;*
   - $\text{RK}\textsc{Extract}(i, S', w^*)$ *and* $\textsc{Extract}(i')$ *for any S', $i \in S^*$ and $i' \in S'$;*
   - $\textsc{Decrypt}\text{-I}(i, S^*, w^*, C^*)$ *for any $i \in S^*$;*
   - $\textsc{ReEncrypt}(i, S', w^*, C^*)$ *and* $\textsc{Extract}(i')$ *for any S', $i \in S^*$ and $i' \in S'$;*
   - $\textsc{Decrypt}\text{-II}(i, i', S^*, S', w^*, C_R^*)$ *for any S' and C_R^*, where $i \in S^*$, $i' \in S'$ and*

     $$\textbf{\textit{Decrypt-II}}(PK, sk_{i'}, i, i', S^*, S', w^*, C_R^*) \in \{m_0, m_1\}.$$
6. *Guess. $\mathcal{A}$ outputs the guess $b' \in \{0, 1\}$. We say $\mathcal{A}$ wins the game if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - \frac{1}{2}|$. A CPBRE scheme is IND-sCond-sSet-RCCA-secure if for any probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in this game is negligible. The CPA security is defined in the same way as RCCA security except that $\mathcal{A}$ is not allowed to query $\textsc{ReEncrypt}$, $\textsc{Decrypt}\text{-I}$ *and* $\textsc{Decrypt}\text{-II}$ *oracles, similar to the definition in [12].*

3 CPA-Secure CPBRE

We start by presenting a CPA version of our final scheme. Without loss of generality, we assume a condition is always specified for every encryption.

3.1 Construction

The basic CPBRE scheme is described as follows.

- **Setup**(n). Pick a prime p and generates groups $\mathbb{G}, \mathbb{G}_T$, bilinear map e and a generator g as defined in Section 2.1. Randomly choose $\alpha, \gamma \in_R \mathbb{Z}_p$ and compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$. Define the function $F(x) = g_1^x h$, where $h \in_R \mathbb{G}$. Let $H' : \mathbb{G}_T \to \mathbb{G}$ be a target collision resistant (TCR) hash. Compute $v = g^\gamma$. Output the public/secret key:

$$PK = (v, g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, F, H'), \quad MK = \gamma.$$

- **KeyGen**(PK, MK, i). The private key for i is defined as

$$sk_i = g_i^\gamma.$$

- **Encrypt**(PK, S, w, m). For the set $S \subseteq \{1, 2, \ldots, n\}$ and the condition $w \in \mathbb{Z}_p$, pick $t \in_R \mathbb{Z}_p$, the ciphertext for message $m \in \mathbb{G}_T$ is output as

$$C = (c_1, c_2, c_3, c_4) = (m \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F(w)^t).$$

- **RKGen**(PK, sk_i, S', w). Randomly choose $s \in \mathbb{Z}_p$, output

$$d_{i \to S'|w} = (sk_i \cdot F(w)^s, C'), \text{ where } C' \leftarrow \textbf{Encrypt'}(PK, S', g^s),$$

where the algorithm **Encrypt'**(PK, S, m) for $S \subseteq \{1, 2, \ldots, n\}$ and $m \in \mathbb{G}$ picks $t \in_R \mathbb{Z}_p$, $\sigma \in_R \mathbb{G}_T$ and outputs the ciphertext as

$$C' = (c'_0, c'_1, c'_2, c'_3) = (m \cdot H'(\sigma), \sigma \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t).$$

- **ReEnc**$(PK, d_{i \to S'|w}, i, S, S', w, C)$. Let $C = (c_1, c_2, c_3, c_4)$ and $d_{i \to S'|w} = (d, C')$. Compute

$$\tilde{c}_1 = c_1 \cdot e(d \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3) \quad \text{and} \quad \tilde{c}_2 = c_4.$$

The re-encrypted ciphertext is output as

$$C_R = (\tilde{c}_1, \tilde{c}_2, C').$$

- **Decrypt-I**(PK, sk_i, i, S, w, C). Let $C = (c_1, c_2, c_3, c_4)$. Output

$$m = c_1 \cdot e(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3).$$

- **Decrypt-II**$(PK, sk_{i'}, i, i', S, S', w, C_R)$. Let $C_R = (\tilde{c}_1, \tilde{c}_2, C')$ and $C' = (c'_0, c'_1, c'_2, c'_3)$. Recover

$$g^s \leftarrow c'_0/H'(c'_1 \cdot \frac{e(sk_{i'} \cdot \prod_{j \in S', j \neq i'} g_{n+1-j+i'}, c'_2)}{e(g_{i'}, c'_3)})$$

and output

$$m = \tilde{c}_1/e(g^s, \tilde{c}_2).$$

Discussion. For the regular encryption, the scheme is just the same as the underlying BGW+BB HICBE scheme, hence we enjoy a constant-size ciphertext. For the re-encryption key under condition w, we encrypt g^s, one of the two elements of $sk_{i,w}$, under the key of the recipient set. Again, its size is independent of the number of delegatees. The same hold trues for re-encrypted ciphertext. The regular decryption procedure is proceeded in **ReEnc**, except for the cancellation of the term $e(g^s, f(w)^t)$. In **Decrypt-II**, the recipient can decrypt C' to get g^s first, and then cancel $e(g^s, f(w)^t)$ to get m.

Correctness. For any integer n, any sets S and S', any indices $i \in S$ and $i' \in S'$, any condition w and any message m, we can see that

- for **Decrypt-I**,

$$\begin{aligned}
&c_1 \cdot e(sk_i \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3)\\
&= c_1 \cdot e(g_i^\gamma \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t)/e(g_i, (v \cdot \prod_{j\in S} g_{n+1-j})^t)\\
&= c_1 \cdot e(\textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t)/e(g_i, \prod_{j\in S} g_{n+1-j}^t)\\
&= c_1/e(g, g_{n+1}^t) = m \cdot e(g_1, g_n)^t/e(g, g_{n+1}^t) = m;
\end{aligned}$$

- for **Decrypt-II**,

$$\begin{aligned}
&\tilde{c}_1/e(g^s, \tilde{c}_2)\\
&= (c_1 \cdot e(d \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))/e(g^s, c_4)\\
&= (c_1 \cdot e(sk_i \cdot F(w)^s \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))/e(g^s, c_4)\\
&= (c_1 \cdot e(sk_i \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))e(F(w)^s, c_2)/e(g^s, c_4)\\
&= m \cdot e(F(w)^s, g^t)/e(g^s, F(w)^t) \qquad \text{(via } \textbf{Decrypt-I}\text{)}\\
&= m.
\end{aligned}$$

3.2 Security

Theorem 2. *Suppose the decisional n-BDHE assumption holds and H' is a TCR hash function, the basic CPBRE scheme for n users described above is IND-sCond-sSet-CPA-secure.*

Proof. Suppose there is an adversary $\mathcal{A}$ breaking our basic CPBRE scheme with non-negligible advantage. Initially, $\mathcal{A}$ outputs a selected set S^* and a selected condition w^*. Then we construct another algorithm $\mathcal{B}$ breaking the underlying HICBE scheme as follows.

Given the public key PK of HICBE, $\mathcal{B}$ simulates the security game of basic CPBRE. Initially, $\mathcal{B}$ prepares two tables:

- EX with an index list: a track of Extract queries.
- RK with columns (i, S', w, d): d is the re-encryption key from index i to the set S' under the condition w.

Moreover, we use $*$ to denote the wildcard symbol.

1. Init. $\mathcal{B}$ outputs S^* and w^* as the target set and target identity of HICBE.
2. Setup. Choose a TCR hash function H'. $\mathcal{B}$ sends PK along with H' to $\mathcal{A}$.

3. Query phase 1. $\mathcal{B}$ answers the following queries issued by $\mathcal{A}$:
   (a) Extract(i): if $i \in S^*$, or $(j, S', w^*, *)$ exists in the RK table, where $j \in S^*, i \in S'$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ forwards the query to the key extraction oracle of HICBE, and responds the received sk_i. $\mathcal{B}$ records i in the EX table.
   (b) RKExtract(i, S', w): if there is a tuple $(i, S', w, d_{i \to S'|w})$ in the RK table, $\mathcal{B}$ responds $d_{i \to S'|w}$ to $\mathcal{A}$. Otherwise, we have the following cases:
      - $i \notin S^*$ or $w \neq w^*$: $\mathcal{B}$ queries i's secret key under identity w from the challenger of HICBE, and responds the re-encryption key as the real scheme (except that g^s is given by the challenger). $\mathcal{B}$ records the tuple $(i, S', w, d_{i \to S'|w})$ in the RK table.
      - $i \in S^*, w = w^*$ and $S' \cap EX \neq \emptyset$: $\mathcal{B}$ responds '$\perp$'.
      - $i \in S^*$ and $w = w^*$, but $S' \cap EX = \emptyset$: $\mathcal{B}$ picks $d, d' \in_R \mathbb{G}$ and responds a random re-encryption key $d_{i \to S'|w} = (d, \textbf{Encrypt'}(PK, S', d'))$. $\mathcal{B}$ records the tuple $(i, S', w, d_{i \to S'|w})$ in the RK table.
4. Challenge. $\mathcal{A}$ sends (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ forwards it to the challenger of HICBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ responds C^* to $\mathcal{A}$.
5. Query phase 2. $\mathcal{B}$ responds $\mathcal{A}$'s queries as in phase 1.
6. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

We can see that $\mathcal{B}$ successfully simulates $\mathcal{A}$'s view in the attack except for a case of re-encryption key queries (when $i \in S^*$ and $w = w^*$). For a randomly chosen key $d_{i \to S'|w} = (d, C')$, there must be a value $s' \in \mathbb{Z}_p$ such that $d = sk_i \cdot F(w)^{s'}$. Therefore the problem is equivalent to the indistinguishability of C' and the encryption of some value $g^{s'}$. This is implied by the CPA security of HICBE and the TCR hash function. So, $\mathcal{B}$ has non-negligible advantage in breaking the HICBE scheme. By Theorem 1, $\mathcal{B}$ has non-negligible advantage in breaking the decisional n-BDHE assumption. □

Note that our constructions can withstand the below attack mentioned in [16].

Chain collusion attack. Suppose Alice is the attack target in a proxy re-encryption scheme. Although the adversary $\mathcal{A}$ cannot get the re-encryption key from Alice to Bob and Bob's private key at the same time, $\mathcal{A}$ can get the re-encryption key from Bob to Carol and Carol's private key instead. If Bob's private key can be derived with these two keys, then Alice's private key can be derived as well by just asking for the re-encryption key from Alice to Bob.

In our constructions, we assume a condition w for each encryption. It means all re-encryption keys given to the proxy are coupled with a condition. The collusion of the proxy and the delegatee can recover the decryption key for some condition only. However, the ciphertext C' in the re-encryption key can be decrypted by the root private key only. So the above attack cannot be applied.

4 RCCA-Secure CPBRE

The RCCA-secure CPBRE scheme follows the structure of the basic CPBRE. However, it is challenging to design a RCCA-secure CPBRE scheme because it involves two kinds of secret keys (regular decryption keys and re-encryption keys) and two kinds of ciphertexts (regular ciphertexts and re-encrypted ciphertexts). Again, we assume a condition is always specified for every encryption.

4.1 Construction

- **Setup**(n). Pick a prime p and generates groups $\mathbb{G}, \mathbb{G}_T$, bilinear map e and a generator g. Randomly choose $\alpha, \gamma \in_R \mathbb{Z}_p$ and compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \ldots, n, n{+}2, \ldots, 2n$. Define the functions $bit(i, x)$ be the i-th bit of a bit-string x, $F_1(x) = g_1^x h_1$ and $F_2(x) = u' \prod u_i{}^{bit(i,x)}$ where $h_1, u, u_1, \cdots, u_\eta \in_R$ $\mathbb{G}$ Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H' : \mathbb{G}_T \rightarrow \mathbb{G}$ be two TCR hash functions. Compute $v = g^\gamma$. Output the public key and the secret key:
$$PK = (v, g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, F_1, F_2, H, H'), \quad MK = \gamma.$$
- **KeyGen**(PK, MK, i). The private key for i is defined as
$$sk_i = g_i^\gamma.$$
- **Encrypt**(PK, S, w, m). For the set $S \subseteq \{1, 2, \ldots, n\}$ and the condition $w \in \mathbb{Z}_p$, the ciphertext for message $m \in \mathbb{G}_T$ can be output as
$$C = (c_1, c_2, c_3, c_4, c_5) = (m \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F_1(w)^t, F_2(h)^t),$$
where $t \in_R \mathbb{Z}_p$ and $h = H(c_1, c_2, c_3, c_4)$.
- **RKGen**(PK, sk_i, S', w). Pick $s \in_R \mathbb{Z}_p$. Output the re-encryption key as
$$d_{i \rightarrow S'|w} = (sk_i \cdot F_1(w)^s, C'), \text{ where } C' \leftarrow \textbf{Encrypt'}(PK, S', g^s).$$
The algorithm **Encrypt'**(PK, S, m) for $S \subseteq \{1, 2, \ldots, n\}$ and $m \in \mathbb{G}$ outputs
$$C' = (c'_0, c'_1, c'_2, c'_3, c'_4) = (m \cdot H'(\sigma), \sigma \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F_2(h')^t),$$
where $t \in_R \mathbb{Z}_p$, $\sigma \in_R \mathbb{G}_T$ and $h' = H(c'_0, c'_1, c'_2, c'_3)$.
- **ReEnc**$(PK, d_{i \rightarrow S'|w}, i, S, S', w, C)$. Let $C = (c_1, c_2, c_3, c_4, c_5)$ and $d_{i \rightarrow S'|w} = (d, C')$. Compute $h = H(c_1, c_2, c_3, c_4)$. Check that
$$e(c_2, v \cdot \textstyle\prod_{j \in S} g_{n+1-j}) \stackrel{?}{=} e(g, c_3), e(c_2, F_2(h)) \stackrel{?}{=} e(g, c_5), e(c_2, F_1(w)) \stackrel{?}{=} e(g, c_4).$$
If any of the equations does not hold or $i \notin S$, output '$\perp$'. Otherwise, compute
$$d'_1 = d \cdot F_2(h)^{s'}, \quad \text{and} \quad d'_2 = g^{s'}.$$
Output the re-encrypted ciphertext
$$C_R = (C, C', d'_1, d'_2).$$

- **Decrypt-I**(PK, sk_i, i, S, w, C). Let $C = (c_1, c_2, c_3, c_4, c_5)$. Compute $h = H(c_1, c_2, c_3, c_4)$ and check that

  $$e(c_2, v \cdot \prod_{j\in S} g_{n+1-j}) \overset{?}{=} e(g, c_3), e(c_2, F_2(h)) \overset{?}{=} e(g, c_5), e(c_2, F_1(w)) \overset{?}{=} e(g, c_4).$$

  If any of the equations does not hold or $i \notin S$, output '$\perp$'. Otherwise, output

  $$m = c_1 \cdot \frac{e(sk_i \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3)}.$$

- **Decrypt-II**$(PK, sk_{i'}, i, i', S, S', w, C_R)$. Let $C_R = (C, C', d'_1, d'_2)$ where $C = (c_1, c_2, c_3, c_4, c_5)$ and $C' = (c'_0, c'_1, c'_2, c'_3, c'_4)$. Compute $h = H(c_1, c_2, c_3, c_4)$ and $h' = H(c'_0, c'_1, c'_2, c'_3)$. Check that

  $$\begin{gathered} e(c_2, v \cdot \prod_{j\in S} g_{n+1-j}) \overset{?}{=} e(g, c_3), \quad e(c'_2, v \cdot \prod_{j\in S'} g_{n+1-j}) \overset{?}{=} e(g, c'_3) \\ e(c_2, F_1(w)) \overset{?}{=} e(g, c_4), \quad e(c_2, F_2(h)) \overset{?}{=} e(g, c_5), \\ e(c'_2, F_2(h')) \overset{?}{=} e(g, c'_4) \quad \text{and} \quad i \in S. \end{gathered} \tag{1}$$

  If any of the equations does not hold, output '$\perp$'. Otherwise, perform

  $$g^s = c'_0 / H'(c'_1 \cdot \frac{e(sk_{i'} \cdot \prod_{j\in S', j\neq i'} g_{n+1-j+i'}, c'_2)}{e(g_{i'}, c'_3)}).$$

  and check that

  $$e(d'_1, g) \overset{?}{=} e(g_i, v) e(F_1(w), g^s) e(F_2(h), d'_2). \tag{2}$$

  If the equation does not hold, output '$\perp$'. Otherwise output

  $$m = c_1 \cdot \frac{e(d'_1 \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3) e(g^s, c_4) e(d'_2, c_5)}.$$

Correctness. The decryption of regular ciphertexts is just the same as our basic CPBRE. For the re-encrypted ciphertexts, we first check the decryption of C':

$$\begin{aligned} & c'_0 / H'(c'_1 \cdot \tfrac{e(sk_{i'} \cdot \prod_{j\in S', j\neq i'} g_{n+1-j+i'}, c'_2)}{e(g_{i'}, c'_3)}) \\ = & c'_0 / H'(c'_1 \cdot e(g_{i'}^{\gamma} \cdot \prod_{j\in S', j\neq i'} g_{n+1-j+i'}, g^t) / e(g_{i'}, (v \cdot \prod_{j\in S'} g_{n+1-j})^t)) \\ = & c'_0 / H'(c'_1 \cdot e(\prod_{j\in S', j\neq i'} g_{n+1-j+i'}, g^t) / e(g_{i'}, \prod_{j\in S'} g_{n+1-j}^t)) \\ = & c'_0 / H'(c'_1 / e(g, g_{n+1}^t)) = g^s \cdot H'(\sigma) / H'(\sigma \cdot e(g_1, g_n)^t / e(g, g_{n+1}^t)) = g^s. \end{aligned}$$

Once g^s is correctly computed, we can compute the message:

$$\begin{aligned} & c_1 \cdot \tfrac{e(d'_1 \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)}{(e(g_i, c_3) e(g^s, c_4) e(d'_2, c_5))} \\ = & c_1 \cdot \tfrac{e(g_i^{\gamma} F_1(w)^s F_2(h)^{s'} \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t)}{(e(g_i, c_3) e(g^s, F_1(w)^t) e(g^{s'}, F_2(h)^t))} \\ = & c_1 \cdot e(g_i^{\gamma} \cdot \prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t) / e(g_i, (v \cdot \prod_{j\in S} g_{n+1-j})^t) \\ = & c_1 \cdot e(\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t) / e(g_i, \prod_{j\in S} g_{n+1-j}^t) \\ = & c_1 / e(g, g_{n+1}^t) = m \cdot e(g_1, g_n)^t / e(g, g_{n+1}^t) = m; \end{aligned}$$

Moreover, it is easy to verify the checking equations are correct. Therefore the correctness of this construction holds.

4.2 Security

In this scheme, we get the RCCA security from the conversion technique of [7]. The scheme involves one more identity level for hash values. This ensures C and C' will not be modified. We did not check the integrity of the re-encrypted ciphertext as a whole. but we use Equation 1 and Equation 2 to check the validity of C, C', d'_1, d'_2 and their relationships to i, S, w. In Lemma 1, we show that if all of these equations hold, C will be decrypted to the original message by d'_1, d'_2 and g^s encrypted in C'. Therefore, the challenger is able to reject any re-randomization of the re-encrypted ciphertexts or any re-encryption of the challenge ciphertext.

Lemma 1. *If a re-encrypted ciphertext $C_R = (C, C', d'_1, d'_2)$ passes Equation 1 and 2, then C_R will be decrypted to the same message as the decryption of C.*

Proof. By Equation 1, we know that C is indeed an encryption under the set S coupling with identity w, C' is an encryption under the set S', and $i \in S$. Then Equation 2 implies that C_R will be decrypted to the same message as the decryption of C, since

$$\begin{aligned}
& c_1 \cdot \frac{e(d'_1 \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3) e(g^s, c_4) e(d'_2, c_5)} \\
&= c_1 \cdot \frac{e(d'_1 \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} = c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(d'_1, g)^t}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} \\
&= c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(g_i, v)^t e(F_1(w), g^s)^t e(F_2(h), d'_2)^t}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} \quad \text{(by Equation 2)} \\
&= c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(g_i, v)^t}{e(g_i, c_3)} = c_1 \cdot \frac{e(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3)}.
\end{aligned}$$

□

Theorem 3. *Suppose the decisional n-BDHE assumption holds and H, H' are two TCR hash functions, the CPBRE scheme for n users described above is IND-sCond-sSet-RCCA-secure.*

Proof. Suppose there is an adversary $\mathcal{A}$ breaking our CPBRE scheme with non-negligible advantage. Initially, $\mathcal{A}$ outputs a selected set S^* and a selected condition w^*. Then we construct another algorithm $\mathcal{B}$ breaking the underlying CCA-secure HICBE scheme (CCA-HICBE) as follows.

Given the public key PK of CCA-HICBE, $\mathcal{B}$ simulates the RCCA game of CPBRE. Initially, $\mathcal{B}$ prepares the following tables:

- EX with (i): a track of Extract queries.
- RK with columns $(i, S', w, d, C', g^s, b_r, b_q)$: the records of re-encryption key (d, C') returned by $\mathcal{B}$, where $C' \leftarrow$ **Encrypt'**(PK, S', g^s); b_r and b_q indicate whether the key is randomly chosen or output by RKExtract oracle.
- RE with column (S'): a track of ReEncrypt(i, S, S', w^*, C^*) queries, where $i \in S^*$ and C^* is the challenge ciphertext.

We use $*$ to denote the wildcard symbol. Note that the intermediate ciphertext C' with a different form cannot be issued to Decrypt oracle.

1. Init. $\mathcal{B}$ outputs S^* and w^* as the target set and identity of CCA-HICBE.
2. Setup. Pick a TCR hash function H'. $\mathcal{B}$ sends PK along with H' to $\mathcal{A}$.
3. Query phase 1. $\mathcal{B}$ answers the following queries issued by $\mathcal{A}$:
   (a) $\text{Extract}(i)$: if $i \in S^*$, or $(j, S', w^*, *, *, *, *, 1)$ exists in the RK table, where $j \in S^*, i \in S'$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ forwards the query to the key extraction oracle of CCA-HICBE, and responds the received sk_i. $\mathcal{B}$ records i in the EX table.
   (b) $\text{RKExtract}(i, S', w)$: if there is a tuple $(i, S', w, d, C', g^s, *, *, 1)$ in the RK table, $\mathcal{B}$ responds (d, C') to $\mathcal{A}$. Otherwise, $\mathcal{B}$ answers the re-encryption key for the following cases:
      - $i \notin S^*$ or $w \neq w^*$: $\mathcal{B}$ queries i's secret key under identity w from the challenger of CCA-HICBE, and computes and responds the re-encryption key (d, C') as the real scheme (except that g^s is given by the challenger). $\mathcal{B}$ records $(j, S', w^*, d, C', g^s, 0, 1)$ on RK.
      - $i \in S^*, w = w^*$ and $S' \cap EX \neq \emptyset$: $\mathcal{B}$ responds '$\perp$'.
      - $i \in S^*$ and $w = w^*$, but $S' \cap EX = \emptyset$: if the RK table contains $(i, S', w, d, C', g^s, 1, 0)$, $\mathcal{B}$ responds (d, C') to $\mathcal{A}$ and sets b_q of this tuple to 1. Otherwise, $\mathcal{B}$ responds a random re-encryption key (d, C'), where d is randomly chosen from $\mathbb{G}$ and $C' \leftarrow \textbf{Encrypt'}(PK, S', g^s)$ for some random $s \in \mathbb{Z}_p$. $\mathcal{B}$ records the tuple $(i, S', w, d, C', g^s, 1, 1)$ in the RK table. Note that in this case, any $\text{Extract}(i')$ query for $i' \in S'$ is forbidden, so the re-encryption key cannot be verified.
   (c) $\text{ReEncrypt}(i, S, S', w, C)$: $\mathcal{B}$ proceeds depending on the following cases:
      - $i \notin S^*$ or $w \neq w^*$: if there is no tuple $(i, S', w, *, *, *, *, *)$ on RK, $\mathcal{B}$ performs $\text{RKExtract}(i, S', w)$ to get the re-encryption key (d, C') and records $(i, S', w^*, d, C', g^s, 0, 0)$ on RK. Then $\mathcal{B}$ re-encrypts C using the re-encryption key on RK as in the real scheme.
      - $i \in S^*, w = w^*$: if $(i, S', w, *, *, *, 1, 1)$ exists on RK, $\mathcal{B}$ re-encrypts C using the re-encryption key on RK as in the real scheme. Otherwise, $\mathcal{B}$ issues the key extraction query on $(i, (w, h))$ to the challenger of CCA-HICBE where $h = H(c_1, c_2, c_3, c_4)$, and gets back the private key $(d_1, d_2, d_3) = (sk_i F_1(w)^s F_2(h)^{s'}, g^s, g^{s'})$. Then $\mathcal{B}$ computes $C' = \textbf{Encrypt'}(PK, S', d_2)$ and responds (C, C', d_1, d_3). $\mathcal{B}$ records $(i, S', w^*, d, C', d_2, 1, 0)$ on RK, where $d \in_R \mathbb{G}$.
   (d) $\text{Decrypt-I}(i, S, w, C)$: $\mathcal{B}$ forwards (i, S, w, C) to the decryption oracle of CCA-HICBE and responds the result to $\mathcal{A}$.
   (e) $\text{Decrypt-II}(i, i', S, S', w, C_R)$: let $C_R = (C, C', d'_1, d'_2)$. $\mathcal{B}$ issues C' to the decryption oracle of CCA-HICBE scheme to obtain g^s first[1]. If the result is '$\perp$', $\mathcal{B}$ responds '$\perp$' as well. For (i, S', w), check whether there

[1] We cannot issue C' to the decryption oracle of CCA-HICBE directly. However, we can modify the CCA-HICBE scheme (removing the ID part and adding the TCR part) to get a new scheme which encrypts messages as in **Encrypt'**. The resulting scheme is like another way of making the BGW [5] scheme CCA-secure using the conversion technique of [7]. It is easy to show that the obtained scheme is IND-sSet-CCA secure assuming the decisional n-BDHE assumption and a TCR hash function.

exists a tuple $(i, S', w, \hat{d}, \hat{C}, g^s, 1, 1)$ on RK. If there does not exist such tuple, $\mathcal{B}$ decrypts the ciphertext as in the real scheme. Otherwise, check

$$e(d'_1, g) \overset{?}{=} e(\hat{d}, g)e(F_2(h), d'_2) \quad \text{and} \quad C' \overset{?}{=} \hat{C}.$$

If both equations hold, C_R is re-encrypted by the random re-encryption key given by $\mathcal{B}$. So $\mathcal{B}$ issues a query (i, S, w, C) to the decryption oracle of CCA-HICBE and responds the result to $\mathcal{A}$. Otherwise, $\mathcal{B}$ decrypts the ciphertext as in the real scheme.

4. Challenge. $\mathcal{A}$ sends (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ forwards it to the challenger of CCA-HICBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ responds C^* to $\mathcal{A}$.
5. Query phase 2. $\mathcal{A}$ continues making the following queries as in phase 1, except for the restrictions described in the definition.
   (a) Extract(i): If $i \in S'$ for some S' on RE, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1.
   (b) RKExtract(i, S', w): $\mathcal{B}$ responds as in Query phase 1.
   (c) ReEncrypt(i, S, S', w, C): If $C = C^*, S = S^*, w = w^*$, $i \in S$ and $S' \cap EX \neq \emptyset$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1. $\mathcal{B}$ records S' on RE if it did not respond '$\perp$' and $C = C^*$.
   (d) Decrypt-I(i, S, w, C): If if $C = C^*, S = S^*, w = w^*$ and $i \in S$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1.
   (e) Decrypt-II(i, i', S, S', w, C_R): let $C_R = (C, C', d'_1, d'_2)$. $\mathcal{B}$ responds the queries as in Query phase 1 except for the case $C = C^*, S = S^*, w = w^*$ and $i \in S$. For the latter case, $\mathcal{B}$ simply responds with '$\perp$'. We explain this by considering the following two cases for $i \in S$:
      - C_R does not pass Equation 1 or 2: $\mathcal{B}$ should return '$\perp$'.
      - C_R passes Equation 1 and 2: by Lemma 1, C_R must be decrypted into m_b. According to our security definition, this query is forbidden.
6. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

$\mathcal{A}$ is successfully simulated except for randomly chosen re-encryption keys. For a randomly chosen re-encryption key (d, C'), there must be a value $s' \in \mathbb{Z}_p$ such that $d = sk_i \cdot F(w)^{s'}$. Therefore the distinguishability of these keys is equivalent to the distinguishability of C' and the encryption of some value $g^{s'}$. By the CCA security of CCA-HICBE and the TCR hash function, this distinguishability is negligible. Moreover, if $\mathcal{A}$ uses a randomly chosen re-encryption key to re-encrypt a ciphertext and issues it to Decrypt-II oracle, $\mathcal{B}$ can make a decryption query on the original ciphertext and respond with a correct message. So, $\mathcal{B}$ has non-negligible advantage in breaking the CCA-HICBE, and hence breaking the decisional n-BDHE assumption, by Theorem 1. □

5 RCCA-Secure Identity-Based Proxy Re-Encryption

Since CPBRE is a generalization of PRE, we can build RCCA-secure IB-PRE from our RCCA-secure CPBRE by letting the broadcast size be 1 and w be user ID. We briefly explain the changes here. Details are deferred to the full paper.

The key generation center executes **Setup**. For user key generation, each user gets a 1-level secret key $(g_1^\gamma F_1(id)^s, g^s)$ for his identity id. Everyone has to encrypt a message under a 2-level identity: recipient's identity and a dummy identity t. In **RKGen**, g^s is encrypted under the recipient's (1-level) identity. Then a proxy colluding with a third party can only get the decryption key of delegator's 2-level identity, which cannot be used to get g^s.

In short, a user Alice with the secret key $(g_1^\gamma F_1(\text{“Alice”})^s, g^s)$ delegates the decryption right to Bob by giving the re-encryption key $(g_1^\gamma F_1(\text{“Alice”})^s F_2(t)^{s'}$, **Encrypt'**$(PK, \text{“Bob”}, g^s), g^{s'})$ to a proxy. The proxy re-encrypts an Alice's ciphertext C in the form (C, C', d_1', d_2'). Then Bob can use his own secret key $(g_1^\gamma F_1(\text{“Bob”})^{s'}, g^{s'})$ to decrypt C' first, and get the decryption of C. The security proof is almost the same as that for the RCCA-secure CPBRE scheme.

We say that a PRE scheme is multi-use if the proxy can re-encrypt a ciphertext multiple times, e.g. re-encrypt from Alice to Bob, then re-encrypt the result from Bob to Carol. To do that in our construction, the proxy only needs to re-encrypt C' to further transform the re-encrypted ciphertext.

6 Conclusions

We introduce conditional proxy broadcast re-encryption, which allows a user to delegate the decryption rights of ciphertexts to a group of users, restricted to a certain condition, via the help of a proxy. Our final scheme is unidirectional and secure against replayable chosen-ciphertext attacks in the standard model, which also gives a unidirectional ID-based proxy re-encryption scheme.

Acknowledgement

Thanks to Junzuo Lai for the discussion that improves our initial construction.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the Network and Distributed System Security Symposium (NDSS 2005). The Internet Society (2005)
2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 161–177. Springer, Heidelberg (2006)
3. 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)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

5. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
6. Boneh, D., Goh, E.-J., Matsuo, T.: Proposal for P1363.3 proxy re-encryption (2006), http://grouper.ieee.org/groups/1363/IBC/submissions
7. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS 2005, pp. 320–329. ACM Press, New York (2005)
8. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of ACM Conference on Computer and Communications Security (CCS 2007), pp. 185–194. ACM Press, New York (2007)
9. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
10. Chu, C.-K., Tzeng, W.-G.: Identity-based proxy re-encryption without random oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)
11. 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)
12. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
13. Libert, B., Vergnaud, D.: Tracing malicious proxies in proxy re-encryption. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 332–353. Springer, Heidelberg (2008)
14. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
15. Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 247–267. Springer, Heidelberg (2007)
16. Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
17. Tang, Q.: Type-based proxy re-encryption and its construction. In: Soomaruga, G. (ed.) Formal Theories of Information. LNCS, vol. 5363, pp. 130–134. Springer, Heidelberg (2008)
18. Weng, J., Deng, R.H., Ding, X., Chu, C.-K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Proceedings of ACM Symposium on Information, Computer & Communication Security (ASIACCS 2009). ACM Press, New York (to appear, 2009)
19. Weng, J., Deng, R.H., Liu, S., Chen, K., Lai, J., Wang, X.: Chosen-ciphertext secure proxy re-encryption schemes without pairings. Technical report, Cryptology ePrint Archive: Report 2008/509 (Version 3) (2008)