Conditional Proxy Re-Encryption Secure against Chosen-Ciphertext Attack

ABSTRACT

In a proxy re-encryption (PRE) scheme [4], a proxy, authorized by Alice, transforms messages encrypted under Alice's public key into encryptions under Bob's public key without knowing the messages. Proxy re-encryption can be used applications requiring delegation, such as delegated email processing. However, it is inadequate to handle scenarios where a fine-grained delegation is demanded. For example, Bob is only allowed Alice's encrypted emails containing a specific keyword. To overcome the limitation of existing PRE, we introduce the notion of *conditional proxy re-encryption* (or *C-PRE*), whereby only ciphertext satisfying one condition set by Alice can be transformed by the proxy and then decrypted by Bob. We formalize its security model and propose an efficient C-PRE scheme, whose chosen-ciphertext security is proven under the 3-quotient bilinear Diffie-Hellman assumption. We further extend the construction to allow multiple conditions with a slightly higher overhead.

1. INTRODUCTION

The notion of proxy re-encryption (PRE) was initially introduced by Blaze, Bleumer and Strauss introduced in [4]. In a PRE system, Bob is allowed to decipher public key encryptions for Alice with the assistance from an authorized proxy. Specifically, Alice authorizes the proxy by giving it a re-encryption key. The proxy can then convert any ciphertext under Alice's public key into ciphertext under Bob's public key. The requirement is that the semantic security of encryptions for Alice is preserved throughout the conversion, such that the proxy gains no information about the involved plaintext messages.

Proxy re-encryption has found many practical applications, such as encrypted email forwarding, secure distributed file systems, and outsourced filtering of encrypted spam. We use the encrypted email forwarding as an example to illustrate the usage of PRE and to motivate our work as well. Imagine that a department manager, Alice, is to take a vacation. She delegates her secretary Bob to process her routine emails. Among the incoming emails, some could be encrypted under Alice's public key. Traditional public key encryption schemes does not allow Bob to process such emails, following the security norm that one's private key should never be shared with other. With a PRE system, Alice can simply give the email server a re-encryption key. For an encrypted incoming email, the email server (i.e. the proxy in PRE's jargon) transforms it into an encryption for Bob. Then Bob can read this email using his secret key. When Alice is back, she instructs the email server to stop the transformation.

The existing notion of PRE does not facilitate flexible delegation. Suppose that Alice instructs Bob to process emails only when its subject contains the keyword *urgent*. For other emails, Alice prefers to read them by herself after back to office. Obviously, the existing PRE schemes do not meet such needs. To show further motivation, we consider the case that Alice wants Bob to process only emails with keyword *market* and prefers Charlie to process emails only with keyword *sales*. Using existing PRE mandates an escalated trust on the email server, since the email server is trusted to enforce the access control policy specified by Alice. We observe that such trust model is unrealistic in many applications.

In this paper, we introduce the notion of *conditional proxy re-encryption* or C-PRE, whereby Alice has a fine-grained control over the delegation. As a result, Alice can flexibly assign her delegate (Bob) the decryption capability based on the conditions attached to the messages, using a proxy with no higher trust than in existing PRE schemes.

1.1 Our Results

Our contribution includes a formal definition of conditional proxy re-encryption and its security notion. Briefly speaking, a C-PRE scheme involves three principals: a delegator (say user U_i), a proxy, and a delegatee (say user U_j), similar to existing PRE systems. A message sent to U_i with condition w is encrypted by the sender using both U_i's public key and w. To authorize U_j to decrypt such an encryption associated with w, U_i gives the proxy a *partial* re-encryption key $rk_{i,j}$ and a condition key $ck_{i,w}$ corresponding to the condition w. These two keys form the secret trapdoor used by the proxy to perform ciphertext translation. The proxy is unable to translate those ciphertext whose corresponding condition keys are not available. Therefore, U_i has a flexible control on delegation by releasing condition keys properly.

We also construct a concrete C-PRE scheme using bilinear

pairings. Under the 3-Quotient Bilinear Diffie-Hellman (3-QBDH) assumption, we prove its chosen-ciphertext security in the random oracle model. We further extend this basic scheme to a conditional proxy re-encryption with multiple conditions (MC-PRE). In the MC-PRE system, a proxy with a partial re-encryption key can translate a ciphertext associated with multiple conditions, if and only if he has *all* the required condition keys. The proposed MC-PRE scheme is efficient, since the number of bilinear pairings in use is independent of the number of the conditions.

1.2 Related Work

In the pioneer work due to Blaze, Bleumer and Strauss [4], they presented the first bidirectional PRE scheme (refer to Remark 1 for the definitions of bidirectional/unidirectional PRE). In 2005, Ateniese et al. [1, 2] presented a unidirectional PRE scheme based on bilinear pairings. Both of these schemes are only secure against chosen-plaintext attack (CPA). However, applications often require security against chosen-ciphertext attacks (CCA). To fill this gap, Canetti and Hohenberger [11] presented a construction of CCA-secure bidirectional PRE scheme from bilinear pairings. Later, Libert and Vergnaud [23] presented a CCA-secrue unidirectional PRE scheme from bilinear pairings. Recently, Deng et al. [15] proposed a CCA-secure bidirectional PRE scheme without pairings. All these constructions are standard PRE schemes, and hence can not control the proxy at a fine-grained level. In Pairing'08, Libert and Vergnaud [24] introduced the notion of traceable proxy re-encryption, where malicious proxies leaking their re-encryption keys can be identified.

Proxy re-encryption has also been studied in identity-based scenarios. Based on the ElGamal-type public key encryption system [16] and Boneh-Boyen's identity-based encryption system [3], Boneh, Goh and Matsuo [7] described a hybrid proxy re-encryption system. Based on Boneh and Franklin's identity-based encryption system [6], Green and Ateniese [18] presented CPA and CCA-secure identity-based proxy re-encryption (IB-PRE) schemes in the random oracle model. Chu and Tzeng [13] presented the constructions of CPA and CCA-secure IB-PRE schemes without random oracles

Another related work is the *proxy encryption* cryptosystem introduced by Mambo and Okamoto [26]. In a proxy encryption scheme [14, 21, 26], A delegator allows a delegatee to decrypt ciphertext intended for her with the help of a proxy: an encryption for the delegator is first partially decrypted by the proxy, and then fully decrypted by the delegatee. However, this scheme requires the delegatee to possess an additional secret for each delegation from Alice. In contrast, the delegatee in proxy re-encryption systems only needs his own private key as in a standard PKE.

Proxy re-encryption should not be confused with the universal re-encryption [19], in which ciphertext is only re-randomized, instead of replacing the public keys.

1.3 Organization

The rest of the paper is organized as follows. Section 2 gives an introduction to bilinear pairings and related complexity assumptions. In Section 3, we formalize the definition and security notions for C-PRE systems. In Section 4, we propose a C-PRE scheme, and prove its chosen-ciphertext security under the 3-QBDH assumption. In Section 5, we further extend our C-PRE scheme to obtain a conditional proxy re-encryption scheme with multiple conditions. Finally, Section 6 lists some open problems and concludes this paper.

2. PRELIMINARIES

2.1 Notations

Throughout this paper, for a prime q, let $\mathbb{Z}_q$ denote $\{0, 1, 2, \cdots, q-1\}$, and $\mathbb{Z}_q^*$ denote $\mathbb{Z}_q \backslash \{0\}$. For a finite set S, $x \xleftarrow{\$} \mathcal{S}$ means choosing an element x from S with a uniform distribution. Finally, a function $f : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $f(k) < k^{-c}$ for all $k > k_c$.

2.2 Bilinear Groups and Bilinear Pairings

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups with the same prime order q. A bilinear pairing is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1, g_2 \in \mathbb{G}, \forall a, b \in \mathbb{Z}_q^*$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- Non-degeneracy: There exist $g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$;
- Computability: There exists an efficient algorithm to compute $e(g_1, g_2)$ for $\forall g_1, g_2 \in \mathbb{G}$.

2.3 Complexity Assumptions

The security of our proposed schemes is based on a complexity assumption called 3-Quotient Bilinear Diffie-Hellman (3-QBDH) assumption. The decisional version of this assumption was recently used to construct a unidirectional proxy re-encryption with chosen-ciphertext security [23]. We briefly review the n-QBDH assumption, a generalized version of 3-QBDH.

The n-QBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given a tuple $(g, g^{1/a}, g^a, \cdots, g^{(a^{n-1})}, g^b) \in \mathbb{G}^{n+2}$ with unknown $a, b \xleftarrow{\$} \mathbb{Z}_q^*$, to compute $e(g, g)^{\frac{b}{a^2}}$. A polynomial-time algorithm $\mathcal{B}$ has *advantage* ϵ in solving the n-QBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$, if

$$\Pr[\mathcal{B}(g, g^{\frac{1}{a}}, g^a, \cdots, g^{(a^{n-1})}, g^b) = e(g, g)^{\frac{b}{a^2}}] \geq \epsilon,$$

where the probability is taken over the random choices of a, b in $\mathbb{Z}_q$, the random choice of g in $\mathbb{G}$, and the random bits consumed by $\mathcal{B}$.

Definition 1. *We say that the (t, ϵ)-n-QBDH assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$ if no t-time adversary $\mathcal{B}$ has advantage at least ϵ in solving the n-QBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$.*

3. MODEL OF CONDITIONAL PROXY RE-ENCRYPTION

In this section, we formalize the definition and security notions for C-PRE systems.

3.1 Definition of C-PRE systems

Formally, a C-PRE scheme consists of the following seven algorithms:

$\mathsf{GlobalSetup}(\lambda)$: The key generation algorithm takes as input a security parameter λ. It generates the global parameters param.

$\mathsf{KeyGen}(i)$: The key generation algorithm generates the public/secret key pair (pk_i, sk_i) for user U_i.

$\mathsf{RKeyGen}(sk_i, pk_j)$: The partial re-encryption key generation algorithm, run by U_i, takes as input a secret key sk_i and another public key pk_j. It outputs a partial re-encryption key $rk_{i,j}$.

$\mathsf{CKeyGen}(sk_i, w)$: The condition key generation algorithm, run by user i, takes as input a secret key sk_i and a condition w. It outputs a condition key $ck_{i,w}$.

$\mathsf{Encrypt}(pk, m, w)$: The encryption algorithm takes as input a public key pk, a plaintext $m \in \mathcal{M}$ and a condition w. It outputs ciphertext CT associated with w under pk. Here $\mathcal{M}$ denotes the message space.

$\mathsf{ReEncrypt}(\mathbf{CT}_i, rk_{i,j}, ck_{i,w})$: The re-encryption algorithm, run by the proxy, takes as input a ciphertext CT_i associated with w under public key pk_i, a partial re-encryption key $rk_{i,j}$ and a condition key $ck_{i,w}$. It outputs a *re-encrypted* ciphertext CT_j under public key pk_j.

$\mathsf{Decrypt}(\mathbf{CT}, sk)$: The decryption algorithm takes as input a secret key sk and a cipertext CT. It outputs a message $m \in \mathcal{M}$ or the error symbol $\perp$.

The correctness of C-PRE means that, for any condition w, any $m \in \mathcal{M}$, any $(pk_i, sk_i) \leftarrow \mathsf{KeyGen}(i)$, $(pk_j, sk_j) \leftarrow \mathsf{KeyGen}(j)$, and $CT_i = \mathsf{Encrypt}(pk_i, m, w)$,

$$\Pr\big[\mathsf{Decrypt}(CT_i, sk_i) = m\big] = 1, \textit{and}$$

$$\Pr\big[\mathsf{Decrypt}\left(\mathsf{ReEncrypt}(CT_i, \mathsf{RKeyGen}(sk_i, pk_j), \mathsf{CKeyGen}(sk_i, w)), sk_j\right) = m\big] = 1;$$

while for any other condition w' and user j' with $w' \neq w$ and $j' \neq j$, we have

$$\Pr\big[\mathsf{Decrypt}\left(\mathsf{ReEncrypt}(CT_i, \mathsf{RKeyGen}(sk_i, pk_j), \mathsf{CKeyGen}(sk_i, w')), sk_j\right) = \perp\big] = 1 - \mathsf{neg}(\lambda)$$

$$\Pr\big[\mathsf{Decrypt}\left(\mathsf{ReEncrypt}(CT_i, \mathsf{RKeyGen}(sk_i, pk_{j'}), \mathsf{CKeyGen}(sk_i, w)), sk_j\right) = \perp\big] = 1 - \mathsf{neg}(\lambda).$$

Remark 1. Blaze, Bleumer and Strauss [4] differentiated two types of proxy re-encryption systems: *bidirectional* PRE and *unidirectional* PRE. In bidirectional PRE systems, the re-encryption key allows the proxy to translate Alice's ciphertext to Bob's and vice versa. In unidirectional PRE systems, the re-encryption key can used only for one direction. In general, unidirectional PRE systems are preferable to bidirectional ones, since (i) any unidirectional scheme can be easily transformed to a bidirectional one [11], and (ii) in bidirectional PRE systems, if the proxy and the delegatee collude, they can recover the delegator's secret key. The same argument applies to C-PRE systems. Therefore, in this paper, we only consider unidirectional C-PRE.

3.2 Security Notions

In plain words, the semantic security of a C-PRE encryption should be preserved against both the delegate and the proxy if they do not possess the proper condition key. More formally, the semantic security against chose-ciphertext attacks for a C-PRE scheme Π can be defined via the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup. Challenger $\mathcal{C}$ runs algorithm $\mathsf{GlobalSetup}(\lambda)$ and gives the global parameters param to $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively issues queries $q_1, \cdots, q_m$ where query q_i is one of the following:

- *Uncorrupted key generation query* $\langle i \rangle$: $\mathcal{C}$ first runs algorithm $\mathsf{KeyGen}(i)$ to obtain a public/secret key pair (pk_i, sk_i), and then sends pk_i to $\mathcal{A}$.
- *Corrupted key generation query* $\langle j \rangle$: $\mathcal{C}$ first runs algorithm $\mathsf{KeyGen}(j)$ to obtain a public/secret key pair (pk_j, sk_j), and then gives (pk_j, sk_j) to $\mathcal{A}$.
- *Partial re-encryption key query* $\langle pk_i, pk_j \rangle$: $\mathcal{C}$ runs algorithm $\mathsf{RKeyGen}(sk_i, pk_j)$ to generate a partial re-encryption key $rk_{i,j}$ and returns it to $\mathcal{A}$. Here sk_i is the secret key with respect to pk_i. It is required that pk_i and pk_j were generated beforehand by algorithm KeyGen.
- *Condition key query* $\langle pk_i, w \rangle$: $\mathcal{C}$ runs algorithm $\mathsf{CKeyGen}(sk_i, w)$ to generate a condition key $ck_{i,w}$ and returns it to $\mathcal{A}$. It is required that pk_i was generated beforehand by algorithm KeyGen.
- *Re-encryption query* $\langle pk_i, pk_j, (w, \mathrm{CT}_i) \rangle$: To reply this query, challenger $\mathcal{C}$ runs algorithm $\mathsf{ReEncrypt}(\mathrm{CT}_i, \mathsf{R}$ and returns the resulting ciphertext CT_j to $\mathcal{A}$. It is required that pk_i and pk_j were generated beforehand by algorithm KeyGen.
- *Decryption query* $\langle pk, (w, \mathrm{CT}) \rangle$ or $\langle pk_j, \mathrm{CT}_j \rangle$: Here $\langle pk, (w, \mathrm{CT}) \rangle$ and $\langle pk, \mathrm{CT} \rangle$ denote the queries on original ciphertexts and re-encrypted ciphertexts respectively. Challenger $\mathcal{C}$ returns the result of $\mathsf{Decrypt}(\mathrm{CT}, sk)$ to $\mathcal{A}$. It is required that pk was generated beforehand by algorithm KeyGen.

Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a target condition w^* and two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$. $\mathcal{C}$ flips a random coin $\delta \in \{0, 1\}$, and sets the challenge ciphertext to be $\mathrm{CT}^* = \mathsf{Encrypt}(pk_{i^*}, m_\delta, w^*)$, which is sent to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ adaptively issues queries as in Phase 1, and $\mathcal{C}$ answers them as before.

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

During the above game, adversary $\mathcal{A}$ is subject to the following restrictions:

(i). $\mathcal{A}$ can not issue corrupted key generation queries on $\langle i^* \rangle$ to obtain the target secret key sk_{i^*}.

(ii). $\mathcal{A}$ can issue decryption queries on neither $\langle pk_{i^*}, (w^*, \mathrm{CT}^*) \rangle$ nor $\langle pk_j, \mathsf{ReEncrypt}(\mathrm{CT}^*, rk_{i^*,j}, ck_{i^*,w^*}) \rangle$.

(iii). $\mathcal{A}$ can not issue re-encryption queries on $\langle pk_{i^*}, pk_j, (w^*, \mathrm{CT}^*) \rangle$ if pk_j appears in a previous corrupted key generation query.

(iv). $\mathcal{A}$ can not obtain both the condition key ck_{i^*,w^*} and the partial re-encryption key $rk_{i^*,j}$ if pk_j appears in a previous corrupted key generation query.

Remark 3. The above four restrictions rule out cases where $\mathcal{A}$ can trivially win the game. To illustrate this, we use the restriction (iv) as an example. Suppose $\mathcal{A}$ obtains both the condition key ck_{i^*,w^*} and the partial re-encryption key $rk_{i^*,j}$ with j is a corrupted user, then she can run algorithm $\mathsf{ReEncrypt}(\mathrm{CT}_{i^*}, rk_{i^*,j}, ck_{i^*,w^*})$ to obtain a re-encrypted ciphertext CT_j under public key pk_j. Using the secret key sk_j, $\mathcal{A}$ can decrypt the ciphertext to recover m_δ, and hence break the challenge.

We refer to the above adversary $\mathcal{A}$ as an IND-CPRE-CCA adversary. His advantage in attacking scheme Π is defined as

$$\mathrm{Adv}_{\Pi,\mathcal{A}}^{\text{IND-CPRE-CCA}} = \left|\Pr[\delta' = \delta] - 1/2\right|$$

where the probability is taken over the random coins consumed by the challenger and the adversary.

Definition 2. *A C-PRE scheme Π is $(t, q_u, q_c, q_{rk}, q_{ck}, q_{re}, q_d, \epsilon)$-IND-CPRE-CCA secure, if and only if for any t-time IND-CPRE-CCA adversary $\mathcal{A}$ that makes at most q_u uncorrupted key generation queries, at most q_c corrupted key generation queries, at most q_{rk} partial re-encryption key queries, at most q_{ck} condition key queries, at most q_{re} re-encryption queries and at most q_d decryption queries, $Adv_{\Pi,\mathcal{A}}^{IND\text{-}CPRE\text{-}CCA} \leq \epsilon$.*

For the sake of an easy understanding of the security proofs for our proposed C-PRE scheme, we differentiate two subcases of restriction (iv) described in the above IND-CPRE-CCA game, and distinguish between two types of IND-CPRE-CCA adversaries:

- Type I IND-CPRE-CCA adversary: During the IND-CPRE-CCA game, adversary $\mathcal{A}$ does not obtain the partial re-encryption key $rk_{i^*,j}$ with pk_j is corrupted.
- Type II IND-CPRE-CCA adversary: During the IND-CPRE-CCA game, adversary $\mathcal{A}$ does not obtain the condition key ck_{i^*,w^*}.

Note that these cases are mutually exclusive (by definition) and complete. Therefore, the IND-CPRE-CCA security of a C-PRE scheme can be proven by showing that neither Type I nor Type II IND-CPRE-CCA adversary can win with a non-negligible advantage.

4. A SECURE C-PRE SCHEME

In this section, we first present our C-PRE scheme, and then briefly explain the intuition behind the construction. Finally, based on the 3-QBDH assumption, we prove the security for the proposed scheme.

4.1 Construction

The proposed C-PRE scheme consists of the following seven algorithms:

$\mathsf{GlobalSetup}(\lambda)$: The setup algorithm takes as input a security parameter λ. It first generates $(q, \mathbb{G}, \mathbb{G}_T, e)$, where q is a λ-bit prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups with prime order q, and e is the bilinear pairing $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$. Next, it chooses $g, g_1, f, f_1 \xleftarrow{\$} \mathbb{G}$, and five hash functions H_1, H_2, H_3, H_4 and H_5 such that $H_1 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_2 : \mathbb{G}_T \to \{0,1\}^{l_0+l_1}$, $H_3 : \{0,1\}^* \to \mathbb{G}$, $H_4 : \mathbb{G}_T \to \{0,1\}^{l_0+l_1}$ and $H_5 : \{0,1\}^* \to \mathbb{Z}_q^*$. Here l_0 and l_1 are determined by the security parameter, and the message space is $\{0,1\}^{l_0}$. The global parameters are

$$\mathsf{param} = ((q, \mathbb{G}, \mathbb{G}_T, e), g, g_1, f, f_1, H_1, \cdots, H_5).$$

$\mathsf{KeyGen}(i)$: To generate the public/secret key pair for user U_i, it picks $x_i \xleftarrow{\$} \mathbb{Z}_q^*$, and sets the public key and secret key to be $pk_i = (P_i, Q_i) = (g^{x_i}, g_1^{1/x_i})$ and $sk_i = x_i$, respectively.

$\mathsf{RKeyGen}(sk_i, pk_j)$: On input a secret key $sk_i = x_i$ and a public key $pk_j = (P_j, Q_j) = (g^{x_j}, g_1^{1/x_j})$, it outputs the partial re-encryption key $rk_{i,j} = P_j^{1/x_i} = g^{x_j/x_i}$.

$\mathsf{CKeyGen}(sk_i, w)$: On input a secret key $sk_i = x_i$ and a condition $w \in \{0,1\}^*$, it outputs the condition key $ck_{i,w} = H_3(w, pk_i)^{1/x_i}$.

$\mathsf{Encrypt}(pk_i, m, w)$: On input a public key $pk_i = (P_i, Q_i)$, a condition w and a message $m \in \{0,1\}^{l_0}$, it works as following:

1. Pick $r' \xleftarrow{\$} \{0,1\}^{l_1}$ and compute $r = H_1(m, r', w)$.
2. Compute and output the ciphertext $\mathrm{CT} = (A, B, C, D)$ as:

$$A = g_1^r,\ B = P_i^r,\ C = H_2\left(e(g,g)^r\right) \oplus (m \| r') \oplus H_4\left(e(Q_i, H_3(\tag{1}$$

$\mathsf{ReEncrypt}(\mathbf{CT}_i, rk_{i,j}, ck_{i,w})$: On input the ciphertext CT_i associated with condition w under public key pk_i, a condition key $ck_{i,w}$ and a partial re-encryption key $rk_{i,j}$, this algorithm generates the *re-encrypted* ciphertext under key pk_j as follows:

1. Parse pk_i as (P_i, Q_i) and CT_i as (A, B, C, D).
2. Check whether both of the following equalities hold:

$$e(A, P_i) = e(g_1, B), \quad e(A, f^{H_5(A,B,C)} f_1) = e(g_1, D). \tag{2}$$

If not, output $\perp$; otherwise, output the re-encrypted ciphertext $\mathrm{CT}_j = (B', C')$ as

$$B' = e(B, rk_{i,j}), C' = C \oplus H_4(e(A, ck_{i,w})). \tag{3}$$

Indeed, the re-encrypted ciphertext $\mathrm{CT}_j = (B', C')$ is of the following forms:

$$B' = e(g, g^{sk_j})^r, \quad C' = H_2(e(g,g)^r) \oplus (m \| r'), \tag{4}$$

where $r = H_1(m, r', w)$.

$\mathsf{Decrypt}(\mathbf{CT}_i, sk_i)$: On input a secret key $sk_i = x_i$ and a ciphertext CT_i under public key $pk_i = (P_i, Q_i)$, this algorithm works according to two cases:

- CT is an original ciphertext associated with a condition w, i.e., $\mathrm{CT} = (A, B, C, D)$: If Eq. (2) does not hold, output $\perp$; otherwise, compute $(m\|r') = C \oplus H_4(e(A, H_3(w, pk_i)^{1/x_i})) \oplus H_2(e(B, g)^{1/x_i})$, and return m if $g^{x_i \cdot H_1(m,r',w)} = B$ holds and $\perp$ otherwise.
- CT is a re-encrypted ciphertext, i.e., $\mathrm{CT} = (B', C')$: Compute $(m\|r') = C' \oplus H_2(B'^{\frac{1}{x_i}})$, and return m if $e(g, g)^{x_i \cdot H_1(m,r',w)} = B'$ holds and $\perp$ otherwise.

Correctness: It can be verified that, all the correctly generated original/re-encrypted ciphertexts can be correctly decrypted. We here explain why a re-encrypted ciphertext, generated by a proxy who *does not* have both the right partial re-encryption key and the right condition key, can not be decrypted by the delegatee with non-neglibible probability. For example, given an original ciphertext $\mathrm{CT}_i = (A, B, C, D)$ associated with condition w under public key $pk_i = (P_i, Q_i)$ as in Eq. (1). Suppose a proxy, who has a partial re-encryption key $rk_{i,j} = g^{x_j/x_i}$ and a condition key $ck_{i,w'} = H_3(w', pk_i)^{1/x_i}$ with $w' \neq w$, runs ReEncrypt to translate ciphertext CT_i into a ciphertext intended for U_j. Obviously, CT_i can pass the validity check of Eq. (2), and hence he generates the re-encrypted ciphertext $\mathrm{CT}'_j = (B', C')$ as

$$\begin{aligned} B' &= e(B, rk_{i,j}) = e(P_i^r, g^{x_j/x_i}) = e(g^{x_i r}, g^{x_j/x_i}) = e(g, g)^{x_j r}, \\ C' &= C \oplus H_4(e(A, ck_{i,w'})) = H_2\left(e(g, g)^r\right) \oplus (m\|r') \oplus H_4\left(e(Q_i, H_3(w, pk_i))^r\right) \oplus H_4(e(A, ck_{i,w'})) \\ &= H_2\left(e(g, g)^r\right) \oplus (m\|r') \oplus H_4\left(e(Q_i, H_3(w, pk_i))^r\right) \oplus H_4\left(e(Q_i, H_3(w', pk_i))^r\right). \end{aligned}$$

Since $w' \neq w$, the term $H_4\left(e(Q_i, H_3(w, pk_i))^r\right)$ in component C' can not be canceled by $H_4\left(e(Q_i, H_3(w', pk_i))^r\right)$ with overwhelming probability. So, when user j with secret key $sk_j = x_j$ receives the above re-encrypted ciphertext CT'_j, he computes $C' \oplus H_2(B'^{1/x_j})$ and obtains $(m\|r') \oplus H_4\left(e(Q_i, H_3(w, pk_i))^r\right) \oplus H_4\left(e(Q_i, H_3(w', pk_i))^r\right)$ instead of $(m\|r')$. Obviously, this resulting value can not pass the validity check as shown in algorithm Decrypt. Therefore, re-encrypted ciphertext CT'_j can not be decrypted by U_j with overwhelming probability.

Security Intuitions: Next, we briefly explain why the proposed scheme can ensure the chosen-ciphertext security. It follows two important facts. First, the re-encrypted ciphertext given in Eq. (4) is indeed a ciphertext of the "hashed" CCA-secure ElGamal encryption [9, 16, 17] using the bilinear pairings, and hence it is impossible for the adversary to gain any advantage through malicious manipulating the re-encrypted ciphertext. Second, the validity of the original ciphertext given in Eq. (1) can be publicly verified by checking Eq. (2). Thus, it is also impossible for the adversary to maliciously manipulate the original ciphertext. In the next subsection, we show detailed security proofs.

4.2 Security

The proposed C-PRE scheme is IND-CPRE-CCA secure in the random oracle model as stated below.

THEOREM 1. *Our C-PRE scheme is IND-CPRE-CCA secure in the random oracle model, assuming the 3-QBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Theorem 1 follows directly from the following Lemma 1 and 2.

LEMMA 1. *If there exists a* $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5}, q_u, q_c, q_{rk}, q_{ck},$ *Type I IND-CPRE-CCA adversary* $\mathcal{A}$ *against our scheme, then there exists an algorithm* $\mathcal{B}$ *which can solve the* (t', ϵ')*-3-QBDH problem in groups* $(\mathbb{G}, \mathbb{G}_T)$ *with*

$$\begin{aligned} \epsilon' &\geq \frac{1}{q_{H_2}}\Big(\frac{\epsilon}{e(1+q_{rk})} - \frac{q_{H_1}(1+q_d)}{2^{l_0+l_1}} - \frac{q_{re}+q_d}{q}\Big), \\ t' &\leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_u + q_c + q_{rk} + q_{ck} + q_{re} + q \\ &\quad + (2q_u + 2q_c + q_{rk} + q_{ck} + q_{re} + q_{H_1}q_{re} + 2q_{H_1}q_d + 3)t_e + (6q_{re} - \end{aligned}$$

where t_e *denotes the running time of an exponentiation in group* $\mathbb{G}$, t_p *denotes the running time of a pairing in groups* $(\mathbb{G}, \mathbb{G}_T)$, $q_{H_i} (i = 1, \cdots, 5)$ *denotes the number of oracle queries to* H_i, *and* $q_u, q_c, q_{rk}, q_{ck}, q_{re}$ *and* q_d *have the same meaning as those in Definition 2.*

PROOF. Suppose $\mathcal{B}$ is given as input a 3-QBDH challenge tuple $(g, g^{1/a}, g^a, g^{(a^2)}, g^b)$ with unknown $a, b \xleftarrow{\$} \mathbb{Z}_q^*$. Algorithm $\mathcal{B}$'s goal is to output $e(g, g)^{\frac{b}{a^2}}$. Algorithm $\mathcal{B}$ first picks $u, \alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_q^*$, defines $g_1 = \left(g^{(a^2)}\right)^u, f = \left(g^{(a^2)}\right)^{\alpha_1}, f_1 = \left(g^{(a^2)}\right)^{\alpha_2}$, and gives (g, g_1, f, f_1) to $\mathcal{A}$. Next, $\mathcal{B}$ acts as a challenger and plays the IND-CPRE-CCA game with adversary $\mathcal{A}$ in the following way:

Hash Oracle Queries. At any time adversary $\mathcal{A}$ can issue the random oracle queries H_i with $i \in \{1, \cdots, 5\}$. Algorithm $\mathcal{B}$ maintains five hash lists H_i^{list} with $i \in \{1, \cdots, 5\}$, which are initially empty, and responds as below:

- H_1 *queries*: On receipt of an H_1 query (m, r', w), if this query already appears on the H_1^{list} in a tuple (m, r', w, r), return the predefined value r. Otherwise, choose $r \xleftarrow{\$} \mathbb{Z}_q^*$, add the tuple (m, r', w, r) to the H_1^{list} and respond with $H_1(m, w, r') = r$.
- H_2 *queries*: On receipt of an H_2 query $U \in \mathbb{G}_T$, if this query already appears on the H_2^{list} in a tuple (U, β), return the predefined value β. Otherwise, choose $\beta \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add the tuple (U, β) to the list H_2^{list} and respond with $H_2(U) = \beta$.
- H_3 *queries*: On receipt of an H_3 query (w, pk_i), if this query already appears on the H_3^{list} in a tuple (w, pk_i, s, S), return the predefined value S. Otherwise, choose $s \xleftarrow{\$} \mathbb{Z}_q^*$, compute $S = (g^a)^s$, add the tuple (w, pk_i, s, S) to the H_3^{list} and respond with $H_3(w, pk_i) = S$.
- H_4 *queries*: On receipt of an H_4 query $V \in \mathbb{G}_T$, if this query already appears on the H_4^{list} in a tuple (V, γ), return the predefined value γ. Otherwise, choose $\gamma \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, add the tuple (V, γ) to the H_4^{list} and respond with $H_4(V) = \gamma$.
- H_5 *queries*: On receipt of an H_5 query (A, B, C), if this query already appears on the H_5^{list} in a tuple (A, B, C, η), return the predefined value η. Otherwise, choose $\eta \xleftarrow{\$} \mathbb{Z}_q^*$, add the tuple (A, B, C, η) to the H_5^{list} and respond with $H_5(A, B, C) = \eta$.

Phase 1. In this phase, adversary $\mathcal{A}$ issues a series of queries subject to the restrictions of the Type I IND-CPRE-CCA game. $\mathcal{B}$ answers these queries for $\mathcal{A}$ as follows:

- *Uncorrupted key generation query* $\langle i \rangle$: Algorithm $\mathcal{B}$ first picks $x_i \xleftarrow{\$} \mathbb{Z}_q^*$. Next, as in Coronaŕs proof technique [12], it flips a biased coin $\mathsf{coin}_i \in \{0,1\}$ that yields 0 with probability θ and 1 with probability $1-\theta$. If $\mathsf{coin}_i = 0$, define $pk_i = (P_i, Q_i) = (g^{a^2 x_i}, g_1^{\frac{1}{a^2 x_i}}) = ((g^{(a^2)})^{x_i}, g^{\frac{u}{x_i}})$; else define $pk_i = (P_i, Q_i) = (g^{a x_i}, g_1^{\frac{1}{a x_i}}) = ((g^a)^{x_i}, (g^a)^{\frac{u}{x_i}})$. Finally, it returns pk_i to adversary $\mathcal{A}$ and adds the tuple $(pk_i, x_i, \mathsf{coin}_i)$ to the K^{list}.
- *Corrupted key generation query* $\langle j \rangle$: Algorithm $\mathcal{B}$ first picks $x_j \xleftarrow{\$} \mathbb{Z}_q^*$ and defines $pk_j = (P_j, Q_j) = (g^{x_j}, (g^{(a^2)})^{u/x_j})$ and $\mathsf{coin}_j = \text{`}-\text{'}$. Next, it adds the tuple $(pk_j, x_j, \mathsf{coin}_j)$ to the K^{list} and returns (pk_j, x_j) to adversary $\mathcal{A}$.
- *Partial re-encryption key query* $\langle pk_i, pk_j \rangle$: $\mathcal{B}$ first parses pk_j as (P_j, Q_j), and recovers tuples $(pk_i, x_i, \mathsf{coin}_i)$ and $(pk_j, x_j, \mathsf{coin}_j)$ from the K^{list}. Next, it constructs the partial re-encryption key $rk_{i,j}$ for adversary $\mathcal{A}$ according to the following situations:
  - If $\mathsf{coin}_i = \text{`}-\text{'}$, it means that $sk_i = x_i$. Algorithm $\mathcal{B}$ simply outputs $rk_{i,j} = P_j^{1/x_i}$.
  - If $(\mathsf{coin}_i = 1 \wedge \mathsf{coin}_j = 0)$, it means that $sk_i = a x_i$ and $sk_j = a^2 x_j$. $\mathcal{B}$ returns $rk_{i,j} = (g^a)^{\frac{x_j}{x_i}}$. Note that this is a valid partial re-encryption key since $(g^a)^{\frac{x_j}{x_i}} = g^{\frac{a^2 x_j}{a x_i}} = P_j^{\frac{1}{sk_i}}$.
  - If $(\mathsf{coin}_i = 1 \wedge \mathsf{coin}_j = 1)$ or $(\mathsf{coin}_i = 0 \wedge \mathsf{coin}_j = 0)$, algorithm $\mathcal{B}$ returns $rk_{i,j} = g^{x_j/x_i}$.
  - If $(\mathsf{coin}_i = 1 \wedge \mathsf{coin}_j = \text{`}-\text{'})$ or $(\mathsf{coin}_i = 0 \wedge \mathsf{coin}_j = 1)$, $\mathcal{B}$ returns $rk_{i,j} = \left(g^{1/a}\right)^{x_j/x_i}$.
  - If $(\mathsf{coin}_i = 0 \wedge \mathsf{coin}_j = \text{`}-\text{'})$, $\mathcal{B}$ outputs "$\mathsf{failure}$" and **aborts**.
- *Condition key query* $\langle pk_i, w \rangle$: Algorithm $\mathcal{B}$ first recovers tuple $(pk_i, x_i, \mathsf{coin}_i)$ from the K^{list} and tuple (w, pk_i, s, S) from the H_3^{list}. Next, it constructs the condition key $ck_{i,w}$ for adversary $\mathcal{A}$ according to the following cases:
  - If $\mathsf{coin}_i = \text{`}-\text{'}$, it means that $sk_i = x_i$: Algorithm $\mathcal{B}$ responds with $ck_{i,w} = S^{1/x_i}$.
  - If $\mathsf{coin}_i = 1$, it means that $sk_i = a x_i$: Algorithm $\mathcal{B}$ responds with $ck_{i,w} = g^{s/x_i}$. Note that this is indeed a valid condition key, since $g^{\frac{s}{x_i}} = g^{\frac{as}{a x_i}} = (g^{as})^{\frac{1}{a x_i}} = H(w, pk_i)^{\frac{1}{sk_i}}$.
  - If $\mathsf{coin}_i = 0$, it means that $sk_i = a^2 x_i$: $\mathcal{B}$ responds with $ck_{i,w} = \left(g^{1/a}\right)^{s/x_i}$. Note that this is indeed a valid condition key, since $\left(g^{1/a}\right)^{\frac{s}{x_i}} = g^{\frac{as}{a^2 x_i}} = (g^{as})^{\frac{1}{a^2 x_i}} = H(w, pk_i)^{\frac{1}{sk_i}}$.
- *Re-encryption query* $\langle pk_i, pk_j, (w', \text{CT}_i) \rangle$: Algorithm $\mathcal{B}$ parses CT_i as $\text{CT}_i = (A, B, C, D)$. If Eq. (2) does not hold, it outputs $\perp$; otherwise, it works as follows:
  1. Recover tuples $(pk_i, x_i, \mathsf{coin}_i)$ and $(pk_j, x_j, \mathsf{coin}_j)$ from the K^{list}.
  2. Issue a condition key query $\langle pk_i, w' \rangle$ to obtain the condition key $ck_{i,w'}$.
  3. Finally, generate the re-encrypted ciphertext according to the following two cases:
     - $\mathsf{coin}_i = 0 \wedge \mathsf{coin}_j = \text{`}-\text{'}$: search whether there exists a tuple $(m, r', w, r) \in H_1^{\text{list}}$ such that $g_1^r = A$ and $w = w'$. If yes, compute $B' = e(g, P_j^r)$, $C' = C \oplus H_4(e(A, ck_{i,w'}))$, and return $\text{CT}_j = (B', C')$ as the re-encrypted ciphertext to $\mathcal{A}$; otherwise return $\perp$.
     - Otherwise: Algorithm $\mathcal{B}$ first constructs the partial re-encryption key $rk_{i,j}$ as in the partial re-encryption key queries, and then runs algorithm $\mathsf{ReEncrypt}(\text{CT}_i, rk_{i,j}, ck_{i,w'})$, and finally returns the resulting re-encrypted ciphertext CT_j to $\mathcal{A}$.
- *Decryption query* $\langle pk_i, (w', \text{CT}) \rangle$ or $\langle pk_i, \text{CT} \rangle$: Algorithm $\mathcal{B}$ parses pk_i as $pk_i = (P_i, Q_i)$ and then recovers tuple $(pk_i, x_i, \mathsf{coin}_i)$ from the K^{list}. If $\mathsf{coin}_i = \text{`}-\text{'}$ (meaning $sk_i = x_i$), algorithm $\mathcal{B}$ decrypts the ciphertext using x_i and returns the plaintext to $\mathcal{A}$. Otherwise, it proceeds as follows.
  1. Parse CT as $\text{CT} = (A, B, C, D)$ or $\text{CT} = (B, C)$. When $\text{CT} = (A, B, C, D)$, work as follows: if Eq. (2) does not hold, return $\perp$, else construct the condition key $ck_{i,w'}$ as in the condition key query, and define $C = C \oplus H_4(e(A, ck_{i,w'}))$.
  2. Search lists H_1^{list} and H_2^{list} to see whether there exist tuples $(m, r', w, r) \in H_1^{\text{list}}$ and $(U, \beta) \in H_2^{\text{list}}$ such that
     $$w = w', P_i^r = B, \beta \oplus (m \| r') = C \text{ and } U = e(g, g)^r.$$
     If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a condition w^* and two equal-length messages $m_0, m_1 \in \{0,1\}^{l_0}$. Algorithm $\mathcal{B}$ responds as follows:

1. Recover tuple $(pk_{i^*}, x^*, \mathsf{coin}^*)$ from the K^{list}. If $\mathsf{coin}^* \neq 0$, output "$\mathsf{failure}$" and **abort**. Otherwise (meaning $sk_{i^*} = a^2 x^*$), algorithm $\mathcal{B}$ continues to execute the following steps.
2. Pick $y^* \xleftarrow{\$} \mathbb{Z}_q^*, C^* \xleftarrow{\$} \{0,1\}^{l_0+l_1}$, and define $A^* = g^{uby^*}, B = g^{bx^* y^*}, D^* = g^{by^*(\alpha_1 H_5(A^*, B^*, C^*) + \alpha_2)}$.
3. Construct the condition key ck_{i^*, w^*} as in the condition key query.
4. Pick a random bit $\delta \xleftarrow{\$} \{0,1\}, r' \xleftarrow{\$} \{0,1\}^{l_1}$. *Implicitly* define $H_1(m_\delta, r', w^*) = \frac{by^*}{a^2}$ and $H_2(e(g,g)^{\frac{by^*}{a^2}}) = C^* \oplus (m_\delta \| r') \oplus H_4(e(A^*, ck_{i^*, w^*}))$ (note that $\mathcal{B}$ knows neither $\frac{by^*}{a^2}$ nor $e(g,g)^{\frac{by^*}{a^2}}$).
5. Return $\text{CT}^* = (A^*, B^*, C^*, D^*)$ as the challenged ciphertext to adversary $\mathcal{A}$.

Note that by the construction given above, if let $r^* \triangleq \frac{by^*}{a^2}$, we can see that the challenged ciphertext CT^* has the same distribution as the real one, since H_2 acts as a random oracle, and

$$A^* = g^{uby^*} = \left(g^{(a^2u)}\right)^{\frac{by^*}{a^2}} = g_1^{r^*}, \qquad B^* = g^{bx^*y^*} = \left(g^{a^2x^*}\right)^{\frac{by^*}{a^2}} = g_1^{R^*},$$

$$\begin{aligned} C^* &= C^* \oplus \big((m_\delta \| r') \oplus H_4(e(A^*, ck_{i^*,w^*})\big) \oplus \big((m_\delta \| r') \oplus H_4(e(A^*, ck_{i^*,w^*})\big), \\ &= \big(C^* \oplus (m_\delta \| r') \oplus H_4(e(A^*, ck_{i^*,w^*})\big) \oplus \big((m_\delta \| r') \oplus H_4(e(Q_{i^*}, H_3(w^*, pk_{i^*})^{r^*})\big) \\ &= H_2(e(g,g)^{\frac{by^*}{a^2}}) \oplus \big((m_\delta \| r') \oplus H_4(e(Q_{i^*}, H_3(w^*, pk_{i^*})^{r^*})\big) \\ &= H_2(e(g,g)^{r^*}) \oplus (m_\delta \| r') \oplus H_4(e(Q_{i^*}, H_3(w^*, pk_{i^*})^{r^*}), \end{aligned}$$

$$D^* = g^{by^*(\alpha_1 H_5(A^*,B^*,C^*)+\alpha_2)} = \left(g^{a^2\alpha_1 H_5(A^*,B^*,C^*)} g^{a^2\alpha_2}\right)^{\frac{by^*}{a^2}} = \left(g^{H_5(A^*,B^*,C^*)} f_1\right)^{r^*}$$

Phase 2. $\mathcal{A}$ continues to issue the rest of queries as in Phase 1, with the restrictions described in the Type I IND-CPRE-CCA game. Algorithm $\mathcal{B}$ responds to these queries as in Phase 1.

Guees. Eventually, adversary $\mathcal{A}$ returns a guess $\delta' \in \{0,1\}$ to $\mathcal{B}$. Algorithm $\mathcal{B}$ randomly picks a tuple (U, β) from the list H_2^{list}, and outputs U^{1/y^*} as the solution to the given 3-QBDH instance.

Analysis. Now let's analyze the simulation. The main idea of the analysis is borrowed from [9]. We first evaluate the simulations of the random oracles. From the constructions of H_3, H_4 and H_5, it is clear that the simulations of these oracles are perfect. As long as adversary $\mathcal{A}$ does not query (m_δ, r', w^*) to H_1 nor $e(g,g)^{\frac{by^*}{a^2}}$ to H_2, where δ and r' are chosen by $\mathcal{B}$ in the Challenge phase, the simulations of H_1 and H_2 are perfect. By AskH_1^* we denote the event that (m_δ, r', w^*) has been queried to H_1. Also, by AskH_2^* we denote the event that $e(g,g)^{\frac{by^*}{a^2}}$ has been queried to H_2.

As argued before, the challenged ciphertext provided for $\mathcal{A}$ is identically distributed as the real one from the construction. From the description of the simulation, it can be seen that the responses to $\mathcal{A}$'s uncorrupted key generation queries, corrupted key generation queries, condition key queries are also perfect.

Next, we analyze the simulation of the partial re-encryption key oracle and the Challenge phase. Obviously, if $\mathcal{B}$ does not abort during the simulation of the partial re-encryption key queries, the response to $\mathcal{A}$'s partial re-encryption key queries is perfect. Similarly, if $\mathcal{B}$ does not abort in the Challenge phase, the Challenge phase is also perfect. Let Abort denote the event of $\mathcal{B}$'s aborting during the whole simulation. Then we have $\Pr[\neg\mathsf{Abort}] \geq \theta^{q_{\text{rk}}}(1-\theta)$, which is maximized at $\theta_{\text{opt}} = \frac{q_{\text{rk}}}{1+q_{\text{rk}}}$. Using θ_{opt}, the probability $\Pr[\neg\mathsf{Abort}]$ is at least $\frac{1}{e(1+q_{\text{rk}})}$.

We proceed to analyze the simulation of the re-encryption oracle. The responses to adversary $\mathcal{A}$'s re-encryption queries are perfect, unless $\mathcal{A}$ can submit valid original ciphertexts without querying hash function H_1(denote this event by $\mathsf{ReEncErr}$). However, since H_1 acts as a random oracle and adversary $\mathcal{A}$ issues at most q_{re} re-encryption queries, we have $\Pr[\mathsf{ReEncErr}] \leq \frac{q_{\text{re}}}{q}$.

Now, we evaluate the simulation of the decryption oracle. The simulation of the decryption oracle is perfect, with the exception that simulation errors may occur in rejecting some valid ciphertexts. However, these errors are not significant as shown below: Suppose a ciphertext CT has been queried to the decryption oracle. Even if CT is a *valid* ciphertext, there is a possibility that CT can be produced without querying $e(g,g)^r$ to H_2, where $r = H_1(m, r', w)$. Let Valid be an event that CT is valid. Let AskH_2 and AskH_1 respectively be events that $e(g,g)^r$ has been queried to H_2 and (m, r', w) has been queried to H_1. We then have

$$\begin{aligned} \Pr[\mathsf{Valid}|\neg\mathsf{AskH}_2] &\leq \cdot \Pr[\mathsf{Valid} \wedge \mathsf{AskH}_1|\neg\mathsf{AskH}_2] + \Pr[\mathsf{Valid} \wedge \neg\mathsf{AskH}_1 \\ &\leq \Pr[\mathsf{AskH}_1|\neg\mathsf{AskH}_2] + \Pr[\mathsf{Valid}|\neg\mathsf{AskH}_1 \wedge \neg\mathsf{AskH} \end{aligned}$$

Let DecErr be the event that $\mathsf{Valid}|\neg\mathsf{AskH}_2$ happens during the entire simulation. Then, since q_{d} decryption oracles are issued, we have $\Pr[\mathsf{DecErr}] \leq \frac{q_{H_1} q_{\text{d}}}{2^{l_0+l_1}} + \frac{q_{\text{d}}}{q}$.

Now let Good denote the event $(\mathsf{AskH}_2^* \vee (\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*) \vee \mathsf{ReEncErr} \vee \mathsf{DecErr})\big|\neg\mathsf{Abort}$. If event Good does not happen, due to the randomness of the output of the random oracle H_2, it is clear that adversary $\mathcal{A}$ can not gain any advantage greater than $\frac{1}{2}$ in guessing δ. Namely, we have $\Pr[\delta = \delta'|\neg\mathsf{Good}] = \frac{1}{2}$. Hence, by splitting $\Pr[\delta' = \delta]$, we have

$$\begin{aligned} \Pr[\delta' = \delta] &= \Pr[\delta' = \delta|\neg\mathsf{Good}]\Pr[\neg\mathsf{Good}] + \Pr[\delta' = \delta|\mathsf{Good}]\Pr[\mathsf{Good} \\ &\leq \frac{1}{2}\Pr[\neg\mathsf{Good}] + \Pr[\mathsf{Good}] = \frac{1}{2}(1 - \Pr[\mathsf{Good}]) + \Pr[\mathsf{Good} \end{aligned}$$

and

$$\Pr[\delta' = \delta] \geq \Pr[\delta' = \delta|\neg\mathsf{Good}]\Pr[\neg\mathsf{Good}] = \frac{1}{2}(1 - \Pr[\mathsf{Good}]) = \frac{1}{2} - \frac{1}{2}$$

By definition of the advantage for the Type I IND-CPRE-CCA adversary, we then have

$$\begin{aligned} \epsilon &= \big|2 \times \Pr[\delta' = \delta] - 1\big| \\ &\leq \Pr[\mathsf{Good}] = \Pr[(\mathsf{AskH}_2^* \vee (\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*) \vee \mathsf{ReEncErr} \vee \mathsf{DecErr} \\ &= \frac{(\Pr[\mathsf{AskH}_2^*] + \Pr[\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*] + \Pr[\mathsf{ReEncErr} + \Pr[\mathsf{DecErr}])}{\Pr[\neg\mathsf{Abort}]}. \end{aligned}$$

Since $\Pr[\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*] \leq \frac{q_{H_1}}{2^{l_0+l_1}}$, $\Pr[\mathsf{DecErr}] \leq \frac{q_{H_1} q_{\text{d}}}{2^{l_0+l_1}} + \frac{q_{\text{d}}}{q}$, $\Pr[\mathsf{ReEncErr}] \leq \frac{q_{\text{re}}}{q}$ and $\Pr[\neg\mathsf{Abort}] \geq \frac{1}{e(1+q_{\text{rk}})}$, we obtain

$$\begin{aligned} \Pr[\mathsf{AskH}_2^*] &\geq \Pr[\neg\mathsf{Abort}] \cdot \epsilon - \Pr[\mathsf{AskH}_1^*|\neg\mathsf{AskH}_2^*] - \Pr[\mathsf{DecErr}] - \Pr \\ &\geq \frac{\epsilon}{e(1+q_{\text{rk}})} - \frac{q_{H_1}}{2^{l_0+l_1}} - \frac{q_{H_1} q_{\text{d}}}{2^{l_0+l_1}} - \frac{q_{\text{d}}}{q} - \frac{q_{\text{re}}}{q} = \frac{\epsilon}{e(1+q_{\text{rk}})} \end{aligned}$$

Meanwhile, if event AskH_2^* happens, algorithm $\mathcal{B}$ will be able to solve the 3-QBDH instance by picking $\left(e(g,g)^{\frac{by^*}{a^2}}\right)^{1/y^*}$ from the list H_2^{list}. Consequently, we obtain

$$\epsilon' \geq \frac{1}{q_{H_2}}\Pr[\mathsf{AskH}_2^*] \geq \frac{1}{q_{H_2}}\Big(\frac{\epsilon}{e(1+q_{\text{rk}})} - \frac{q_{H_1}(1+q_{\text{d}})}{2^{l_0+l_1}} - \frac{q_{\text{re}}+q_{\text{d}}}{q}\Big).$$

From the description of the simulation, the running time of algorithm $\mathcal{B}$ can be bounded by

$$\begin{aligned} t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{\mathrm{u}} + q_{\mathrm{c}} + q_{\mathrm{rk}} + q_{\mathrm{ck}} + q_{\mathrm{re}} + q_{\mathrm{d}})\mathcal{O}(1) \\ & + (2q_{\mathrm{u}} + 2q_{\mathrm{c}} + q_{\mathrm{rk}} + q_{\mathrm{ck}} + q_{\mathrm{re}} + q_{H_1}q_{\mathrm{re}} + 2q_{H_1}q_{\mathrm{d}} + 3)t_{\mathrm{e}} + (6q_{\mathrm{re}} + 5q_{\mathrm{d}} + 1)t_{\mathrm{p}}. \end{aligned}$$

This completes the proof of Lemma 1. □

Next, we prove that under the 2-QBDH assumption, there exists no Type II IND-CPRE-CCA adversary $\mathcal{A}$ against our scheme with non-negligible probability. Note that the 2-QBDH assumption is weaker than the 3-QBDH assumption and is implied by the latter.

LEMMA 2. *If there exists a $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_u, q_c, q_{rk}, q_{ck}, q_{re}, q_d, \epsilon)$ Type II IND-CPRE-CCA adversary $\mathcal{A}$ against our scheme, then there exists an algorithm $\mathcal{B}$ which can solve the (t', ϵ')-2-QBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$ with*

$$\epsilon' \geq \frac{1}{q_{H_4}}\Big(\frac{\epsilon}{e(1+q_{ck})} - \frac{q_{H_1}(1+q_d)}{2^{l_0+l_1}} - \frac{q_{re}+q_d}{q}\Big),$$

$$\begin{aligned} t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_u + q_c + q_{rk} + q_{ck} + q_{re} + q_d)\mathcal{O}(1) \\ & + (2q_u + 2q_c + q_{rk} + q_{ck} + 2q_{re} + q_{H_1}q_{re} + 2q_{H_1}q_d + 3)t_e + (6q_{re} + 4q_d + q_{H_1}q_d + 1)t_p. \end{aligned}$$

where $t_e, t_p, q_{H_i} (i = 1, \cdots, 5), q_u, q_c, q_{rk}, q_{ck}, q_{re}$ and q_d have the same meaning as Lemma 1.

The proof is in Appendix A.

5. EXTENSIONS

In this section, we extend our C-PRE scheme to obtain a conditional proxy re-encryption system with multiple conditions (MC-PRE). In a MC-PRE system, the proxy with a partial re-encryption key $rk_{i,j}$ can translate ciphertexts associated with a set of conditions $\{w_t\}_{t\in\{k_1,\cdots,k_n\}}$ from user U_i to user U_j, *if and only if* he has *all* the condition keys $\{ck_{i,w_t}\}_{t\in\{k_1,\cdots,k_n\}}$ with respect to these conditions.

Based on the C-PRE scheme in Section 4.1, we present a MC-PRE scheme. The proposed MC-PRE scheme consists of seven algorithms, where GlobalSetup, KeyGen, RKeyGen and CKeyGen are the same as those in the C-PRE scheme, and the other three algorithms are specified as below:

Encrypt$(pk, m, \{w_t\}_{t\in\{k_1,\cdots,k_n\}})$**:** On input a public key $pk = (P, Q)$, a plaintext $m \in \{0,1\}^{l_0}$ and a set of conditions $\{w_t\}_{t\in\{k_1,\cdots,k_n\}}$, this algorithm works as below:

1. Pick $r' \xleftarrow{\$} \{0,1\}^{l_1}$ and compute $r = H_1(m, r', \{w_t\}_{t\in\{k_1,\cdots,k_n\}})$.
2. Compute $A = g_1^r, B = P^r, C = H_2\left(e(g,g)^r\right) \oplus (m\|r') \oplus H_4\Big(e\big(Q, \big(\prod_{t\in\{k_1,\cdots,k_n\}} H_3(w_t, pk)\big)^r\big)\Big)$, and $D = \Big(f^{H_5(A,B,C)} f_1\Big)^r$.
3. Output the original ciphertext $\mathrm{CT} = (A, B, C, D)$.

ReEncrypt$(\mathbf{CT}_i, rk_{i,j}, \{ck_{i,w_t}\}_{t\in\{k_1,\cdots,k_n\}})$**:** On input a ciphertext CT_i associated with a set of conditions $\{w_t\}_{t\in\{k_1,\cdots,k_n\}}$ under public key pk_i, a partial re-encryption key $rk_{i,j}$ and a set of condition key $\{ck_{i,w_t}\}_{t\in\{k_1,\cdots,k_n\}}$, it generates the ciphertext under key pk_j as follows:

1. Parse $pk_i = (P_i, Q_i)$ and $\mathrm{CT}_i = (A, B, C, D)$.
2. Check whether both of the following equalities hold:
$$e(A, P_i) = e(g_1, B), \quad e(A, f^{H_5(A,B,C)} f_1) = e(g_1, D). \tag{5}$$
If not, output $\perp$; otherwise, compute $B' = e(B, rk_{i,j})$, $C' = C \oplus H_4\Big(e(A, \prod_{t\in\{k_1,\cdots,k_n\}} ck_{i,w_t})\Big)$, and output the re-encrypted ciphertext $\mathrm{CT}_j = (B', C')$.

Decrypt$(\mathbf{CT}, sk)$**:** On input a secret key $sk = x$ and a ciphertext CT under public key pk, this algorithm works according to two cases:

- CT is an original ciphertext associated with a set of conditions $\{w_t\}_{t\in\{k_1,\cdots,k_n\}}$, i.e., $\mathrm{CT} = (A, B, C, D)$: Check whether Eq. (5) holds. If not, output $\perp$. Otherwise, compute $(m\|r') = C \oplus H_4\Big(e(A, \prod_{t\in\{k_1,\cdots,k_n\}} H_3(w$ $H_2\big(e(B,g)^{1/x}\big)$, and if $B = g^{x\cdot H_1(m,r',\{w_t\}_{t\in\{k_1,\cdots,k_n\}})}$, return m, else return $\perp$.
- $\mathrm{CT} = (B', C')$: Compute $(m\|r') = C' \oplus H_2(B'^{1/x})$. If $B' = e(g,g)^{x\cdot H_1(m,r',\{w_t\}_{t\in\{k_1,\cdots,k_n\}})}$, return m, else return $\perp$.

Interestingly, the number of bilinear pairings needed in the above MC-PRE scheme is *independent* of the number of conditions, and the efficiency of this MC-PRE scheme is comparable to that of the C-PRE scheme in Section 4.1.

Similarly to the C-PRE scheme, the chosen-ciphertext security of the proposed scheme can be proved under the 3-QBDH assumption. Of course, the security model should be slightly modified to address the situations under multiple conditions. Due to the space limit, we omit the security model and proofs.

6. CONCLUSIONS AND OPEN QUESTIONS

In this paper, we tackle the problem of how to control the proxy in PRE systems at a fine-grained level. We introduce the concept of conditional proxy re-encryption. We formalize its definition and its security notions, and propose a CCA-secure C-PRE scheme. We further extend this C-PRE scheme to support multiple conditions with reasonable overhead. The conditions in our proposed solution are limited to keywords. It remains as an interesting open problem how to construct CCA-secure C-PRE schemes with anonymous conditions or boolean predicates.

Acknowledgment

This research is supported by the Office of Research, Singapore Management University.

7. REFERENCES

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage. In Proc. of NDSS 2005, pp. 29-43, 2005.

[2] 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, February 2006.

[3] D. Boneh, and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In advances in Cryptology-Eurocrypt'04, LNCS 3027, pp. 223-238, Springer-Verlag, 2004.
[4] M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In advances in Cryptology-Eurocrypt'98, LNCS 1403, pp. 127-144, Springer-Verlag, 1998.
[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In Advanecs in Cryptology-Eurocrypt'04, LNCS 3027, pp. 506-522, Springer-Verlag, 2004.
[6] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In Advanecs in Cryptology-Crypto'01, LNCS 2139, pp. 213-229. Springer-Verlag, 2001.
[7] D. Boneh, E.-J. Goh, and T. Matsuo. Proposal for P1363.3 Proxy Re-encryption. http://grouper.ieee.org/groups/1363/IBC/submissions/NTTDataProposal-for-P1363.3-2006-08-14.pdf.
[8] D. Boneh, E.-J. Goh, K. Nissim. Evaluating 2-DNF Formulas on Ciphertexts. In Proc. of TCC'05, LNCS 3378, pp. 325-341. Springer-Verlag, 2005.
[9] J. Baek, R. Safavi-Naini and W. Susilo. Certificatless Public Key Encryption without Pairing. In Proc. of ISC'05, LNCS 3650, pp. 134-148, Springer-Verlag, 2005.
[10] D. Boneh, B. Waters. Conjunctive, Subset, and Range Queries on Encrypted Data. In Proc. of TCC'07, LNCS 4392, pp. 535-554, Springer-Verlag, 2007.
[11] R. Caneti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In Proc. of ACM CCS 2007, pp. 185-194. ACM Press, 2007.
[12] J.-S. Coron. On the Exact Security of Full Domain Hash. In advances in Cryptology-Crypto'00, LNCS 1880, pp. 229-235, Springer-Verlag, 2000.
[13] C. Chu and W. Tzeng. Identity-Based Proxy Re-Encryption without Random Oracles. In Proc. of ISC'07, LNCS 4779, pp. 189-202, Springer-Verlag, 2007.
[14] Y. Dodis, and A.-A. Ivan. Proxy Cryptography Revisited. In Proc. of NDSS'03, 2003.
[15] R. H. Deng, J. Weng, S. Liu, K. Chen. Chosen-Cipertext Secure Proxy Re-Encryption without Pairings. To appear in CANS'08. Springer-Verlag, 2008.
[16] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In Advances in Cryptology-Crypto'84, LNCS 196, pp.10-18, Springer-Verlag, 1984.
[17] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes, In Advances in Cryptology-Crypto'99, LNCS 1666, pp. 537-554, Springer-Verlag, 1999.
[18] M. Green and G. Ateniese. Identity-Based Proxy Re-Encryption. In Proc. of ACNS'07, LNCS 4521, pp. 288-306, Springer-Verlag, 2007.
[19] P. Golle, M. Jakobsson, A. Juels, and P. F. Syverson. Universal Re-Encryption for Mixnets. In Proc. of CT-RSA'04, LNCS 2964, pp. 163-178, Springer-Verlag, 2004.
[20] P. Golle, J. Staddon, and B. Waters. Secure Conjunctive Keyword Search over Encrypted Data. In Proc. of ACNS'04. LNCS 3089, pp. 31-45, Springer-Verlag, 2004.
[21] M. Jakobsson. On Quorum Controlled Asummetric Proxy Re-Encryption. In Proc. of PKC'99, LNCS 1560, pp. 112-121, Springer-Verlag, 1999.
[22] J. Katz, A. Sahai, and B. Waters: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In advances in Cryptology-Eurocrypt'08, LNCS 4965, pp. 146-162, Springer-Verlag, 2008.
[23] B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In Proc. of PKC'08, LNCS 4929, pp. 360-379, Springer-Verlag, 2008.
[24] B. Libert and D. Vergnaud. Tracing Malicious Proxies in Proxy Re-Encryption. In Proc. of Pairing'08, LNCS 5209, pp. 332-353, Springer-Verlag, 2008.
[25] T. Matsuo. Proxy Re-Encryption Systems for Identity-Based Encryption. In Proc. of Paring'07, LNCS 4575, PP. 247-267, Springer-Verlag, 2007.
[26] Masahiro Mambo and Eiji Okamoto. Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts. IEICE Trans. Fund. Electronics Communications and Computer Science, E80-A/1:54-63, 1997.
[27] E. Shi, B. Waters: Delegating Capabilities in Predicate Encryption Systems. In Proc. of ICALP (2) 2008, LNCS 5126, pp. 560-578, Springer-Verlag, 2008.

Appendix

A Proof of Lemma 2

PROOF. Suppose $\mathcal{B}$ is given as input a 2-QBDH challenge tuple $(g, g^{1/a}, g^a, g^b)$ with unknown $a, b \xleftarrow{\$} \mathbb{Z}_q^*$. Algorithm $\mathcal{B}$'s goal is to output $e(g,g)^{b/a^2}$. Algorithm $\mathcal{B}$ first picks $u, \alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_q^*$, defines $g_1 = g^u, f = (g^a)^{\alpha_1}, f_1 = (g^a)^{\alpha_2}$, and gives (g, g_1, f, f_1) to $\mathcal{A}$. Next, $\mathcal{B}$ acts as a challenger and plays the Type II IND-CPRE-CCA game with adversary $\mathcal{A}$ in the following way:

Hash Oracle Queries. $\mathcal{B}$ maintains five hash lists H_i^{list} with $i \in \{1, \cdots, 5\}$, and responds in the same way as in the proof of Lemma 1, except that H_3 queries are conducted in the following way:

- *H_3 queries*: On receipt of an H_3 query (w, pk_i), if this query already appears on the H_3^{list} in a tuple $(w, pk_i, s, S, \mathsf{coin})$, return the predefined value S. Otherwise, pick $s \xleftarrow{\$} \mathbb{Z}_q^*$ and flip a random biased coin $\mathsf{coin} \in \{0,1\}$ that yields 0 with probability θ and 1 with probability $1-\theta$. If $\mathsf{coin} = 0$ then the hash value $H_3(w, pk_i)$ is defined as $S = g^s$, else $S = g^{bs}$. Finally, S is returned to $\mathcal{A}$ and $(w, pk_i, s, S, \mathsf{coin})$ is added to the H_3^{list}.

Phase 1. In this phase, $\mathcal{A}$ issues a series of queries subject to the restrictions of the Type II IND-CPRE-CCA game. $\mathcal{B}$ maintains a list K^{list}, and answers these queries as follows:

- *Uncorrupted key generation query* $\langle i \rangle$: $\mathcal{B}$ first picks $x_i \xleftarrow{\$} \mathbb{Z}_q^*$, and defines $pk_i = (P_i, Q_i) = (g^{ax_i}, g_1^{\frac{1}{ax_i}}) = ((g^a)^{x_i}, (g^{\frac{1}{a}})^{\frac{u}{x_i}})$. Next, it defines $c_i = 0$, adds the tuple (pk_i, x_i, c_i) to the K^{list}, and returns pk_i to adversary $\mathcal{A}$. Here the bit c_i is used to denote whether the secret key with respect to pk_i is corrupted, i.e., $c_i = 0$ indicates uncorrupted and $c_i = 1$ means corrupted.
- *Corrupted key generation query* $\langle j \rangle$: $\mathcal{B}$ first picks $x_j \xleftarrow{\$} \mathbb{Z}_q^*$ and defines $pk_j = (P_j, Q_j) = (g^{x_j}, g^{\frac{u}{x_j}}), c_j = 1$. Next, it adds the tuple (pk_j, x_j, c_j) to the K^{list} and returns (pk_j, x_j) to $\mathcal{A}$.
- *Partial re-encryption key query* $\langle pk_i, pk_j \rangle$: $\mathcal{B}$ recovers tuples (pk_i, x_i, c_i) and (pk_j, x_j, c_j) from the K^{list}, and constructs the partial re-encryption key $rk_{i,j}$ for $\mathcal{A}$ according to the following cases:
  - If $c_i = c_j$. Respond with $rk_{i,j} = g^{x_j/x_i}$.
  - If $c_i = 1 \wedge c_j = 0$. Respond with $rk_{i,j} = (g^a)^{x_j/x_i}$.
  - If $c_i = 0 \wedge c_j = 1$. Respond with $rk_{i,j} = \left(g^{1/a}\right)^{x_j/x_i}$.

- *Condition key query* $\langle pk_i, w\rangle$: $\mathcal{B}$ first recovers the tuple (pk_i, x_i, c_i) from the K^{list} and the tuple $(w, pk_i, s, S, \mathsf{coin})$ from the H_3^{list}. Next, it constructs the condition key $ck_{i,w}$ for adversary $\mathcal{A}$ according to the following cases:
  - If $c_i = 1$, it means that $sk_i = x_i$. Algorithm $\mathcal{B}$ responds with $ck_{i,w} = S^{1/x_i}$.
  - If $c_i = 0 \wedge \mathsf{coin} = 0$, it means that $sk_i = ax_i$ and $H_3(w, pk_i) = g^{s_i}$. Algorithm $\mathcal{B}$ responds with $ck_{i,w} = \left(g^{1/a}\right)^{s_i/x_i}$.
  - If $c_i = 0 \wedge \mathsf{coin} = 1$, it means that $sk_i = ax_i$ and $H_3(w, pk_i) = g^{bs_i}$. Algorithm $\mathcal{B}$ outputs "$\mathsf{failure}$" and **aborts**.
- *Re-encryption query* $\langle pk_i, pk_j, (w', \text{CT}_i)\rangle$: $\mathcal{B}$ parses $pk_i = (P_i, Q_i), pk_j = (P_j, Q_j)$ and $\text{CT}_i = (A, B, C, D)$. If Eq. (2) does not hold, it outputs $\perp$; otherwise, it acts as follows:
  1. Recover tuples (pk_i, x_i, c_i) and (pk_j, x_j, c_j) from the K^{list}.
  2. Issue a partial re-encryption key query to obtain the partial re-encryption key $rk_{i,j}$.
  3. Recover the tuple $(w', pk_i, s, S, \mathsf{coin})$ from the H_3^{list}, and produce the re-encrypted ciphertext according to the following two cases:
     - $c_i = 0 \wedge \mathsf{coin} = 1$: Search whether there exists a tuple $(m, r', w, r) \in H_1^{\text{list}}$ such that $g_1^r = A$ and $w = w'$. If yes, compute $B' = e(g, P_j^r)$, $C' = C \oplus H_4(e(Q_i, S^r))$, and return $\text{CT}_j = (B', C')$ as the re-encrypted ciphertext to $\mathcal{A}$; otherwise return $\perp$.
     - Otherwise: Algorithm $\mathcal{B}$ first constructs the condition key $ck_{i,w'}$ as in the condition key queries, and then returns $\mathsf{ReEncrypt}(\text{CT}_i, rk_{i,j}, ck_{i,w'})$ to $\mathcal{A}$.
- *Decryption query* $\langle pk_i, (w', \text{CT})\rangle$ or $\langle pk_i, \text{CT}\rangle$: Algorithm $\mathcal{B}$ responds as follows:
  1. Parse pk_i as (P_i, Q_i). Recover the tuple (pk_i, x_i, c_i) from the K^{list}. If $c_i = 1$ (i.e., $sk_i = x_i$), decrypt CT using x_i and return the resulting plaintext to $\mathcal{A}$.
  2. Parse CT as $\text{CT} = (A, B, C, D)$ or $\text{CT} = (B, C)$. When $\text{CT} = (A, B, C, D)$, return $\perp$ if Eq. (2) does not hold.
  3. Search lists H_1^{list} and H_2^{list} to see whether there exist tuples $(m, r', w, r) \in H_1^{\text{list}}$ and $(U, \beta) \in H_2^{\text{list}}$ such that
     $$P_i^r = B, U = e(g, g)^r, w = w',$$
     and $\begin{cases} \beta \oplus (m\|r') = C, & \text{If CT}=(A, B, C, D); \\ \beta \oplus (m\|r') \oplus H_4(e(Q_i, H_3(w, pk_i))^r) = C, & \text{If CT}=(B, C). \end{cases}$
     If yes, return m to $\mathcal{A}$. Otherwise, return $\perp$.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key $pk_{i^*} = (P_{i^*}, Q_{i^*})$, a condition w^* and two equal-length messages $m_0, m_1 \in \{0,1\}^{l_0}$. Algorithm $\mathcal{B}$ responds as follows:

1. Recover the tuple $(w^*, pk_{i^*}, s^*, S^*, \mathsf{coin}^*)$ from the H_3^{list}. If $\mathsf{coin}^* = 0$, output "$\mathsf{failure}$" and abort. Otherwise (meaning that $H_3(w^*, pk_{i^*}) = g^{bs^*}$), continue to execute the rest steps.
2. Recover the tuple (pk_{i^*}, x^*, c^*) from the K^{list}.
3. Pick $y^* \xleftarrow{\$} \mathbb{Z}_q^*$ and $C^* \xleftarrow{\$} \{0,1\}^{l_0+l_1}$. Define $A^* = \left(g^{1/a}\right)^{uy^*}, B = g^{x^*y^*}, D^* = g^{(\alpha_1 H_5(A^*, B^*, C^*) + \alpha_2)y^*}$.
4. Pick $\delta \xleftarrow{\$} \{0,1\}$ and $r' \xleftarrow{\$} \{0,1\}^{l_1}$. Implicitly define $H_1(m_\delta, r', w^*) = \frac{y^*}{a}$ and $H_4(e(g,g)^{\frac{ubs^*y^*}{a^2x^*}}) = C^* \oplus (m_\delta\|r') \oplus H_2(e(g, g^{\frac{1}{a}})^{y^*})$ (note that algorithm $\mathcal{B}$ knows neither $\frac{y^*}{a}$ nor $e(g,g)^{\frac{ubs^*y^*}{a^2x^*}}$).
5. Return $\text{CT}^* = (A^*, B^*, C^*, D^*)$ as the challenged ciphertext to adversary $\mathcal{A}$.

Note that by the construction given above, if let $r^* \triangleq \frac{y^*}{a}$, we can see that the challenged ciphertext CT^* has the same distribution as the real one, since H_2 and H_4 act as random oracles, and

$$\begin{aligned}
A^* &= \left(g^{1/a}\right)^{uy^*} = \left(g^u\right)^{y^*/a} = g_1^{r^*}, \\
B^* &= g^{x^*y^*} = \left(g^{ax^*}\right)^{y^*/a} = P_{i^*}^{r^*}, \\
C^* &= (m_\delta\|r') \oplus H_2(e(g, g^{1/a})^{y^*}) \oplus \left(C^* \oplus (m_\delta\|r') \oplus H_2(e(g, g^{1/a})^{y^*}\right) \\
&= (m_\delta\|r') \oplus H_2(e(g,g)^{\frac{y^*}{a}}) \oplus H_4(e(g,g)^{\frac{ubs^*y^*}{a^2x^*}}) \\
&= (m_\delta\|r') \oplus H_2(e(g,g)^{\frac{y^*}{a}}) \oplus H_4(e((g^u)^{\frac{1}{ax^*}}, g^{bs^*})^{\frac{y^*}{a}}) \\
&= (m_\delta\|r') \oplus H_2(e(g,g)^{r^*}) \oplus H_4(e(Q_{i^*}, H_3(w^*, pk_{i^*})^{r^*}), \\
D^* &= g^{(\alpha_1 H_5(A^*, B^*, C^*) + \alpha_2)y^*} = \left(g^{a(\alpha_1 H_5(A^*, B^*, C^*) + \alpha_2)}\right)^{y^*/a} = \left(f^{H_5(}\right.
\end{aligned}$$

Phase 2. Adversary $\mathcal{A}$ continues to issue the rest of queries as in Phase 1, with the restrictions described in the Type II IND-CPRE-CCA game. $\mathcal{B}$ responds to these queries as in Phase 1.

Guess. Eventually, adversary $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$. Algorithm $\mathcal{B}$ randomly picks a tuple (V, γ) from the list H_4^{list}, and outputs $V^{\frac{x^*}{us^*y^*}}$ as the solution to the given 2-QBDH instance.

Analysis. Similarly to the analysis in Lemma 1, it can be seen that $\mathcal{B}$'s advantage against the 2-QBDH problem is at least

$$\epsilon' \geq \frac{1}{q_{H_4}}\Big(\frac{\epsilon}{e(1+q_{\text{ck}})} - \frac{q_{H_1}(1+q_{\text{d}})}{2^{l_0+l_1}} - \frac{q_{\text{re}}+q_{\text{d}}}{q}\Big),$$

and $\mathcal{B}$'s running time is bounded by

$$t' \leq t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{\text{u}} + q_{\text{c}} + q_{\text{rk}} + q_{\text{ck}} + q_{\text{re}} + q$$
$$+ (2q_{\text{u}} + 2q_{\text{c}} + q_{\text{rk}} + q_{\text{ck}} + 2q_{\text{re}} + q_{H_1}q_{\text{re}} + 2q_{H_1}q_{\text{d}} + 3)t_{\text{e}} + (6q_{\text{r}}$$

This completes the proof of Lemma 2. □