Efficient Conditional Proxy Re-encryption with Chosen-Ciphertext Security

Jian Weng[1,2], Yanjiang Yang[3], Qiang Tang[4], Robert H. Deng[1], and Feng Bao[3]

[1] School of Information Systems,
Singapore Management University, Singapore 178902
[2] Department of Computer Science, Jinan University, Guangzhou 510632, P.R. China
cryptjweng@gmail.com, robertdeng@smu.edu.sg
[3] Institute for Infocomm Research (I2R), Singapore, 119613
yyang@i2r.a-star.edu.sg, baofeng@i2r.a-star.edu.sg
[4] DIES, Faculty of EEMCS, University of Twente, The Netherlands
q.tang@utwente.nl

Abstract. Recently, a variant of proxy re-encryption, named conditional proxy re-encryption (C-PRE), has been introduced. Compared with traditional proxy re-encryption, C-PRE enables the delegator to implement fine-grained delegation of decryption rights, and thus is more useful in many applications. In this paper, based on a careful observation on the existing definitions and security notions for C-PRE, we reformalize more rigorous definition and security notions for C-PRE. We further propose a more efficient C-PRE scheme, and prove its chosen-ciphertext security under the decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle model. In addition, we point out that a recent C-PRE scheme fails to achieve the chosen-ciphertext security.

Keywords: Conditional proxy re-encryption, chosen-ciphertext security, random oracle.

1 Introduction

In 1998, Blaze, Bleumer and Strauss [1] introduced the notion of proxy re-encryption (PRE). In a PRE scheme, a proxy is given a re-encryption key, and thus can translate ciphertexts under Alice's public key into ciphertexts under Bob's public key[1]. The proxy, however, cannot learn anything about the messages encrypted under either key. PRE turns out to be a useful primitive, and has found many applications requiring delegation of decryption right, such as encrypted email forwarding, secure distributed file systems, and outsourced filtering of encrypted spam.

Nevertheless, there exist some situations which are hard for traditional PRE to tackle. For example, suppose some of Alice's second level ciphertexts are *highly*

[1] In [2,3,4], the original ciphertext is called *second level ciphertext*, and the transformed ciphertext is named *first level ciphertext*. Through out this paper, we will follow these notations.

P. Samarati et al. (Eds.): ISC 2009, LNCS 5735, pp. 151–166, 2009.

secret, and she wants to decrypt these ciphertexts *only* by herself. Unfortunately, traditional PRE enables the proxy to convert *all* of Alice's second level ciphertexts, without any discrimination. To address this issue, two variants of PRE were independently introduced: one is named type-based proxy re-encryption (TB-PRE) introduced by Tang [5], and the other is named conditional proxy re-encryption (C-PRE) introduced by Weng *et al.* [6]. Although different in naming, C-PRE and TB-PRE are the same in spirit (for consistency, in the rest of the paper, we use C-PRE to denote the two variants.). In such systems, ciphertexts are generated with respect to a certain condition, and the proxy can translate a ciphertext only if the associated condition is satisfied. Compared with traditional PRE, C-PRE enables the delegator to implement fine-grained delegation of decryption rights, thereby more useful in many applications.

1.1 Our Motivations and Results

We first investigate the definitions and security notions for C-PRE defined in [6,5]. Both have their respective pros and cons: (i) In Weng *et al.*'s definition, the proxy needs *two* key pairs (i.e., the partial re-encryption key and the condition key) to perform the transformation, while the proxy in Tang *et al.*'s definition has only one key pair; (ii) In Tang's definition, the delegators and the delegatees have to be in different systems, which means that the user in a given system can *only* act as either (not both) a delegator or a delegatee. In contrast, in Weng *et al.*'s definition, a user can be the delegator for any other users, and can also be the delegatee for any other users. (iii) Both of the security notions in [5,6] only consider the second level ciphertext security, and do not address the first level ciphertext security.

In this paper, we re-formalize the definition for C-PRE by incorporating the advantages in [6,5]. More specifically, in our formalization the proxy holds only one key (re-encryption key) for performing transformations, and a user can act as the delegator or the delegatee for any other users. We also define the first level ciphertext security for C-PRE. We then propose a new C-PRE scheme, and prove its CCA-security under the well-studied decisional bilinear Diffie-Hellman (DBDH) assumption in the random oracle model. Our scheme has better overall efficiency in terms of both computation and communication than Tang's and Weng *et al.*'s schemes. In addition, we show that Weng *et al.*'s C-PRE scheme fails to achieve the CCA-security.

1.2 Related Work

Mambo and Okamoto [7] firstly introduced the concept of delegation of decryption rights, as a better-performance alternative to the trivial approach of decrypting-then-encrypting of ciphertexts. Blaze, Bleumer and Strauss [1] formalized the concept of proxy re-encryption, and proposed the first bidirectional PRE scheme (in which the delegation from Alice to Bob also allows re-encryption from Bob to Alice). In 2005, Ateniese *et al.* [2,3] presented unidirectional PRE schemes based on bilinear pairings.

The schemes in [1,2,3] are only secure against chosen-plaintext attacks (CPA). However, applications often require the CCA-security. In ACM CCS'07, Canetti and Hohenberger [8] presented a CCA-secure bidirectional PRE scheme from bilinear pairings. Later, Libert and Vergnaud [4] gave a unidirectional PRE scheme secure against replayable chosen-ciphertext attacks (RCCA) [9]. In their extended version, Libert and Vergnaud [10] further consider the the problem of conditional proxy re-encryption, and suggested a RCCA-secure C-PRE scheme in the standard model without assuming registered public keys[2].

Previous PRE schemes rely on the costly bilinear pairings. Thus Canetti and Hohenberger [8] left an open question to construct CCA-secure PRE without pairings. In CANS'08, Deng *et al.* [11] proposed a CCA-secure bidirectional PRE scheme without pairings. In PKC'09, Shao and Cao [12] proposed a unidirectional PRE scheme without pairings, and claimed that their scheme is CCA-secure. However, Weng *et al.* [13] pointed out that Shao and Cao's PRE scheme is not CCA-secure by presenting a concrete attack. Weng *et al.* [13] further presented an efficient CCA-secure unidirectional PRE scheme without pairings.

Traceable proxy re-encryption, introduced by Libert and Vergnaud [14], attempts to solve the problem of disclosing re-encryption keys, by tracing the proxies who have done so. Proxy re-encryption has also been studied in identity-based scenarios, such as [15, 16, 17]. Recently, Chu *et al.* [18] introduced a generalized version of C-PRE named conditional proxy broadcast re-encryption (CPBRE), in which the proxy can re-encrypt the ciphertexts for a set of users at a time.

2 Model of Conditional Proxy Re-encryption

Before re-formalizing the definition and security notions for C-PRE, we first explain some notations used in the rest of this paper. For a finite set S, $x \in_R \mathcal{S}$ means choosing an element x from S with a uniform distribution. For a string x, $|x|$ denotes its bit-length. We use $\mathcal{A}(x, y, \cdots)$ to indicate that $\mathcal{A}$ is an algorithm with the input $(x, y, \cdots)$. By $z \leftarrow \mathcal{A}(x, y, \cdots)$, we indicate the running of $\mathcal{A}(x, y, \cdots)$ and letting z be the output. We use $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$ to denote that $\mathcal{A}$ is an algorithm with the input $(x, y, \cdots)$ and can access to oracles $\mathcal{O}_1, \mathcal{O}_2, \cdots$. By $z \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$, we denote the running of $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$, and letting z be the output.

2.1 Definition of C-PRE Systems

Weng *et al.*'s definition differentiates between partial re-encryption key and condition key. A more standard model should combine them into an integral entity. Our definition is standard in this regard, having only re-encryption key; and we allow the delegators and the delegatees to share the same systems, unlike Tang's model. Formally, a C-PRE scheme consists of the following algorithms:

[2] We sincerely thank one of the anonymous reviewers for pointing out that, Libert and Vergnaud [10] also suggested a C-PRE scheme in the standard model without assuming registered public keys.

Setup(1^κ)**:** On input a security parameter 1^κ, this algorithm outputs a global parameter $param$, which includes the message space $\mathcal{M}$. For brevity, we assume that $param$ is implicitly included in the input of the rest algorithms.

KeyGen(1^κ)**:** all parties use this randomize key generation algorithm to generate a public/private key pair (pk_i, sk_i).

ReKeyGen(sk_i, w, pk_j)**:** On input the delegator's private key sk_i, a condition w and the delegatee's public key pk_j, the re-encryption key generation algorithm outputs a re-encryption key $rk_{i \xrightarrow{w} j}$.

Enc$_2(pk, m, w)$**:** On input a public key pk, a plaintext $m \in \mathcal{M}$ and a condition w, the second encryption algorithm outputs a second level ciphertext CT, which can be re-encrypted into a first level one (intended for a possibly different receiver) using the suitable re-encryption key.

Enc$_1(pk, m)$**:** On input a public key pk and a plaintext $m \in \mathcal{M}$, this first encryption algorithm outputs a first level ciphertext CT that cannot be re-encrypted for another party.

ReEnc$(\mathbf{CT}_i, rk_{i \xrightarrow{w} j})$**:** On input a second level ciphertext CT_i associated with w under public key pk_i, and a re-encryption key $rk_{i \xrightarrow{w} j}$, this re-encryption algorithm, run by the proxy, outputs a first level ciphertext CT_j under public key pk_j.

Dec$_2(\mathbf{CT}, sk)$**:** On input a second level cipertext CT and a private key sk, this second decryption algorithm outputs a message m or the error symbol $\perp$.

Dec$_1(\mathbf{CT}, sk)$**:** On input a first level cipertext CT and a private key sk, this first decryption algorithm outputs a message m or the error symbol $\perp$.

The correctness of C-PRE means that, for any condition w, any $m \in \mathcal{M}$, and any couple of private/public key pairs (pk_i, sk_i), (pk_j, sk_j), it holds that

$$\mathsf{Dec}_2(\mathsf{Enc}_2(pk_i, m, w), sk_i) = m, \quad \mathsf{Dec}_1(\mathsf{Enc}_1(pk_i, m), sk_i) = m,$$
$$\mathsf{Dec}_1\left(\mathsf{ReEnc}(\mathsf{Enc}_2(pk_i, m, w), \mathsf{ReKeyGen}(sk_i, w, pk_j)), sk_j\right) = m.$$

2.2 Security Notions

In this subsection, we will define the security notions for C-PRE systems. Before giving these security notions, we first consider the following oracles which together model the ability of an adversary. These oracles are provided for the adversary $\mathcal{A}$ by a challenger $\mathcal{C}$ who simulates an environment running C-PRE.

- Uncorrupted key generation oracle $\mathcal{O}_u(i)$: $\mathcal{C}$ runs algorithm KeyGen to generate a public/private key pair (pk_i, sk_i), and returns pk_i to $\mathcal{A}$.
- Corrupted key generation oracle $\mathcal{O}_c(i)$: $\mathcal{C}$ runs algorithm KeyGen to generate a public/private key pair (pk_j, sk_j),, and returns (pk_j, sk_j) to $\mathcal{A}$.
- Re-encryption key oracle $\mathcal{O}_{rk}(pk_i, w, pk_j)$: Challenger $\mathcal{C}$ first runs $rk_{i \xrightarrow{w} j} \leftarrow \mathsf{ReKeyGen}(sk_i, w, pk_j)$, and then returns $rk_{i \xrightarrow{w} j}$ to $\mathcal{A}$.
- Re-encryption oracle $\mathcal{O}_{re}(pk_i, pk_j, (w, \mathrm{CT}_i))$: Challenger $\mathcal{C}$ first runs $\mathrm{CT}_j \leftarrow \mathsf{ReEnc}(\mathrm{CT}_i, rk_{i \xrightarrow{w} j})$, where $rk_{i \xrightarrow{w} j} = \mathsf{ReKeyGen}(sk_i, w, pk_j)$, and then returns CT_j to $\mathcal{A}$.

– First level decryption oracle $\mathcal{O}_{1d}(pk, \mathrm{CT})$: Here CT is a first level ciphertext. $\mathcal{C}$ runs $\mathsf{Dec}_1(\mathrm{CT}, sk)$, and returns the corresponding result to $\mathcal{A}$.

Note that for the last three oracles, it is required that pk_i, pk_j and pk were generated beforehand by either $\mathcal{O}_c$ or $\mathcal{O}_u$.

We are now ready to define the semantic security for C-PRE under chose-ciphertext attacks. Libert and Vergnaud [4]differentiated two kinds of semantic security for traditional (single-hop) unidirectional PRE systems: *first level ciphertext security* and *second level ciphertext security*. We here follow Libert and Vergnaud's definitions, and define these two kinds security notions for C-PREs.

Second level ciphertext security. Intuitively speaking, second level ciphertext security models the scenario that the adversary $\mathcal{A}$ is challenged with a second level ciphertext CT^* encrypted under a target public key pk_{i^*} and a target condition w^*. $\mathcal{A}$ can issue a series of queries to the above five oracles. These queries are allowed as long as they would not allow $\mathcal{A}$ to decrypt trivially. For examples, $\mathcal{A}$ should not query on $\mathcal{O}_{rk}(pk_{i^*}, w^*, pk_j)$ to obtain an re-encryption key $rk_{i^* \xrightarrow{w^*} j}$ where pk_j came from oracle $\mathcal{O}_c$. Otherwise, $\mathcal{A}$ can trivially decrypt the challenge ciphertext by first re-encrypting it into a first level ciphertext and then decrypting it with sk_j. Similarly, $\mathcal{A}$ cannot query on $\mathcal{O}_{re}(pk_{i^*}, pk_j, (w^*, \mathrm{CT}^*))$ where pk_j came from oracle $\mathcal{O}_c$. Also, for a first level ciphertext $\mathrm{CT}' = \mathsf{ReEnc}(\mathrm{CT}^*, rk_{i^* \xrightarrow{w^*} j})$, $\mathcal{A}$ is disallowed to query on $\mathcal{O}_{1d}(pk_j, \mathrm{CT}')$. One might wonder that why we do not provide the second level decryption oracle for $\mathcal{A}$. In fact, explicitly providing adversary $\mathcal{A}$ with this oracle is useless, since (i). for the challenge ciphertext CT^*, $\mathcal{A}$ is obviously not allowed to ask the second level decryption oracle to decrypt it; (ii). while for any other second level ciphertext CT_t encrypted under public key pk_t and condition w such that $(pk_t, w, \mathrm{CT}_t) \neq (pk_{i^*}, w^*, \mathrm{CT}^*)$, adversary $\mathcal{A}$ can first issue a re-encryption query $\mathcal{O}_{re}(pk_t, pk_j, (w, \mathrm{CT}_t))$ to obtain a first level ciphertext CT_j, and then issue a first level decryption query $\mathcal{O}_{1d}(pk_j, \mathrm{CT}_j)$ to obtain the underlying plaintext. Below gives the formal definition for second level ciphertext's sematic security under adaptive chosen ciphertext attack ($\mathsf{IND\text{-}2CPRE\text{-}CCA}$).

Definition 1. *For a C-PRE scheme $\mathcal{E}$ and a probabilistic polynomial time adversary $\mathcal{A}$ running in two stages* `find` *and* `guess`*, we define $\mathcal{A}$'s advantage against the* $\mathsf{IND\text{-}2CPRE\text{-}CCA}$ *security of $\mathcal{E}$ as*

$$\mathrm{Adv}^{\mathsf{IND\text{-}2CPRE\text{-}CCA}}_{\mathcal{E},\mathcal{A}}(1^\kappa) = \left| \Pr\left[\delta' = \delta \;\middle|\; \begin{array}{c} param \leftarrow \mathsf{Setup}(1^\kappa) \\ (pk_{i^*}, w^*, (m_0, m_1), \mathtt{st}) \leftarrow \mathcal{A}^{\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{1d}}_{\mathtt{find}}(param) \\ \delta \in_R \{0,1\}, \mathrm{CT}^* \leftarrow \mathsf{Enc}_2(pk_{i^*}, m_\delta, w^*) \\ \delta' \leftarrow \mathcal{A}^{\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}, \mathcal{O}_{re}, \mathcal{O}_{1d}}_{\mathtt{guess}}(param, \mathrm{CT}^*, \mathtt{st}) \end{array} \right] - \frac{1}{2} \right|,$$

where `st` *is some internal state information of adversary $\mathcal{A}$. Here it is mandated that $|m_0| = |m_1|$, and the following requirements are simultaneously satisfied: (i). pk_{i^*} is generated by oracle $\mathcal{O}_u$; (ii). For a public key pk_j generated by oracle $\mathcal{O}_c$, $\mathcal{A}$ cannot issue the query $\mathcal{O}_{rk}(pk_{i^*}, w^*, pk_j)$; (iii) For a public key pk_j generated*

by oracle $\mathcal{O}_c$*,* $\mathcal{A}$ *cannot issue the query* $\mathcal{O}_{re}(pk_{i^*}, pk_j, (w^*, \mathrm{CT}^*))$*; (iv). For a public key* pk_j *and the first level ciphertext* $\mathrm{CT}' = \mathsf{ReEnc}(\mathrm{CT}^*, rk_{i^* \overset{w^*}{\rightarrow} j})$*,* $\mathcal{A}$ *cannot issue the query* $\mathcal{O}_{1d}(pk_j, \mathrm{CT}')$*.*

We refer to adversary $\mathcal{A}$ *as an* IND-2CPRE-CCA *adversary. A C-PRE scheme* $\mathcal{E}$ *is said to be* $(t, q_u, q_c, q_{rk}, q_{re}, q_{1d}, \epsilon)$-IND-2CPRE-CCA *secure, if for any* t*-time* IND-2CPRE-CCA *adversary* $\mathcal{A}$*, who makes at most* q_u, q_c, q_{rk}, q_{re} *and* q_d *queries to* $\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}, \mathcal{O}_{re}$ *and* $\mathcal{O}_{1d}$*, respectively, we have* $\mathrm{Adv}^{\text{IND-2CPRE-CCA}}_{\mathcal{E},\mathcal{A}}(1^\kappa) \leq \epsilon$*.*

First Level Ciphertext Security. The above definition provides the adversary with a second level ciphertext in the challenge phase. Next, we define a complementary definition of security (denote by IND-1CPRE-CCA) by providing the adversary with a first level ciphertext in the challenge phase. Note that, since the first level ciphertext cannot be re-encrypted in a single hop C-PRE scheme, $\mathcal{A}$ is allowed to obtain *any* re-encryption keys. Furthermore, given these re-encryption keys, $\mathcal{A}$ can re-encrypt ciphertexts by himself, and hence there is no need to provide the re-encryption oracle $\mathcal{O}_{re}$ for him. As argued before, the second level decryption oracle is also unnecessary.

Definition 2. *For a C-PRE scheme* $\mathcal{E}$ *and a probabilistic polynomial time adversary* $\mathcal{A}$ *running in two stages* `find` *and* `guess`*, we define* $\mathcal{A}$*'s advantage against the* IND-1CPRE-CCA *security of* $\mathcal{E}$ *as*

$$\mathrm{Adv}^{\text{IND-1CPRE-CCA}}_{\mathcal{E},\mathcal{A}}(1^\kappa) = \left| \Pr\left[\delta' = \delta \;\middle|\; \begin{array}{c} param \leftarrow \mathsf{Setup}(1^\kappa) \\ (pk_{i^*}, (m_0, m_1), \mathtt{st}) \leftarrow \mathcal{A}^{\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}, \mathcal{O}_{1d}}_{\mathtt{find}}(param) \\ \delta \in_R \{0,1\}, \mathrm{CT}^* \leftarrow \mathsf{Enc}_1(pk_{i^*}, m_\delta) \\ \delta' \leftarrow \mathcal{A}^{\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}, \mathcal{O}_{1d}}_{\mathtt{guess}}(param, \mathrm{CT}^*, \mathtt{st}) \end{array} \right] - \frac{1}{2} \right|,$$

where `st` *is some internal state information of adversary* $\mathcal{A}$*. Here it is mandated that,* $|m_0| = |m_1|$*,* pk_{i^*} *is generated by* $\mathcal{O}_u$*, and* $\mathcal{A}$ *cannot issue the query* $\mathcal{O}_{1d}(pk_{i^*}, \mathrm{CT}^*)$*.*

We refer to the above adversary $\mathcal{A}$ *as an* IND-1CPRE-CCA *adversary. We say that a C-PRE scheme* $\mathcal{E}$ *is* $(t, q_u, q_c, q_{rk}, q_{1d}, \epsilon)$-IND-1CPRE-CCA *secure, if for any* t*-time* IND-1CPRE-CCA *adversary* $\mathcal{A}$ *that makes at most* q_u, q_c, q_{rk} *and* q_d *queries to oracles* $\mathcal{O}_u, \mathcal{O}_c, \mathcal{O}_{rk}$ *and* $\mathcal{O}_{1d}$*, respectively, we have* $\mathrm{Adv}^{\text{IND-1CPRE-CCA}}_{\mathcal{E},\mathcal{A}}(1^\kappa) \leq \epsilon$*.*

Remark. In [2], Ateniese *et al.* defined the notion *master secret security*, for unidirectional proxy re-encryption. This security notion catches the intuition that, even if the dishonest proxy colludes with the delegatee, it is still impossible for them to derive the delegator's private key. Note that for C-PREs, there is no need to define master secret security, since this security is implied by the first level ciphertext security. This is due to the fact that, if the dishonest proxy and the delegatee can collude to derive the delegator's private key, they can certainly use this private key to decrypt the challenge ciphertext, and thus break the first level ciphertext security.

3 Proposed CCA-Secure C-PRE Scheme

In this section, we propose a new C-PRE scheme with CCA-security. Before presenting our scheme, we list three important and *necessary* principles for designing CCA-secure C-PRE schemes: (i) the validity of the second level ciphertexts should be *publicly verifiable*; otherwise, it will suffer from a similar attack as illustrated in [11, 19]; (ii) the second level ciphertexts should be able to resist the adversary's malicious manipulating; (iii) it should also be impossible for the adversary to maliciously manipulate the first level ciphertext. We remark that it is non-trivial to design a C-PRE scheme satisfying these three requirements, especially the last one. To help understand our scheme, we first present an insecure attempt, and then improve it to obtain our final CCA-secure scheme.

3.1 A First Attempt

We denote this first attempt by $\mathsf{S1}$, which is specified as below:

Setup(1^κ)**:** On input a security parameter 1^κ, the setup algorithm first determines $(q, \mathbb{G}, \mathbb{G}_T, e)$, where q is a κ-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 $g \in_R \mathbb{G}$, and five hash functions H_1, H_2, H_3, H_4 and H_5 such that $H_1 : \{0,1\}^* \to \mathbb{Z}_q, H_2 : \{0,1\}^* \to \mathbb{G}$, $H_3 : \mathbb{G} \to \{0,1\}^n$, $H_4 : \{0,1\}^* \to \mathbb{G}$ and $H_5 : \mathbb{G} \to \mathbb{Z}_q$, where n is polynomial in κ and the message space is $\mathcal{M} = \{0,1\}^n$. The global parameter is $param = ((q, \mathbb{G}, \mathbb{G}_T, e), g, n, H_1, \cdots, H_5)$.

KeyGen(1^κ)**:** To generate the public/private key pair for user U_i, it picks $x_i \in_R \mathbb{Z}_q$, and sets the public key and private key to be $pk_i = g^{x_i}$ and $sk_i = x_i$, respectively.

ReKeyGen(sk_i, w, pk_j)**:** On input a private key sk_i, a condition w and a public key pk_j, this algorithm randomly picks $s \in_R \mathbb{Z}_q$, and outputs the re-encryption key as

$$rk_{i \xrightarrow{w} j} = (rk_1, rk_2) = \left(\left(H_2(pk_i, w) pk_j^s \right)^{-sk_i}, pk_i^s \right). \tag{1}$$

Enc$_2(pk, m, w)$**:** On input a public key pk, a condition w and a message $m \in \mathcal{M}$, the sender first picks $R \in_R \mathbb{G}_T$. Then he computes $r = H_1(m, R)$, and outputs the second level ciphertext $\mathrm{CT} = (C_1, C_2, C_3, C_4)$ as

$$\left(g^r,\ R \cdot e(pk, H_2(pk, w))^r,\ m \oplus H_3(R),\ H_4(C_1, C_2, C_3)^r \right). \tag{2}$$

Note that the last ciphertext component, C_4, is used to ensure the public verifiability of the ciphertext, while the first three components, (C_1, C_2, C_3), are in fact the ciphertext of the CCA-secure ElGamal encryption scheme [20] applying the Fujisaki-Okamoto transformation [21].

Enc$_1(pk, m)$**:** On input a public key pk and a message $m \in \mathcal{M}$, the sender first picks $R \in_R \mathbb{G}_T$ and $\overline{s} \in_R \mathbb{Z}_q^*$. Then he computes $r = H_1(m, R)$, and outputs the first level ciphertext CT as

$$\mathrm{CT} = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4) = \left(g^r,\ R \cdot e(g, pk)^{-r \cdot \overline{s}},\ m \oplus H_3(R),\ g^{\overline{s}} \right). \tag{3}$$

ReEnc$(\mathbf{CT}_i, rk_{i \xrightarrow{w} j})$**:** On input a second level ciphertext $\mathrm{CT}_i = (C_1, C_2, C_3, C_4)$ associated with condition w under public key pk_i, and a re-encryption key $rk_{i \xrightarrow{w} j} = (rk_1, rk_2)$, it generates the first level ciphertext under public key pk_j as follows: Check whether the following equality holds:

$$e(C_1, H_4(C_1, C_2, C_3)) = e(g, C_4). \tag{4}$$

If not, output $\perp$; else output $\mathrm{CT}_j = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ as

$$\overline{C}_1 = C_1, \quad \overline{C}_2 = C_2 \cdot e(C_1, rk_1), \quad \overline{C}_3 = C_3, \quad \overline{C}_4 = rk_2. \tag{5}$$

Observe that $\mathrm{CT}_j = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ is indeed of the following form:

$$\overline{C}_1 = g^r, \quad \overline{C}_3 = m \oplus H_3(R), \quad \overline{C}_4 = pk_i^s = g^{s \cdot sk_i},$$
$$\overline{C}_2 = R \cdot e(pk_i, H_2(pk_i, w))^r \cdot e\left(g^r, \left(H_2(pk_i, w) pk_j^s\right)^{-sk_i}\right) = R \cdot e\left(g, pk_j\right)^{-r \cdot s \cdot sk_i}.$$

Letting $\overline{s} = s \cdot sk_i$, it can be seen that the above first level ciphertext has the same form as Eq. (3).

Dec$_2(\mathbf{CT}, sk)$**:** On input a private key sk and a second level ciphertext $\mathrm{CT} = (C_1, C_2, C_3, C_4)$, it first checks whether Eq. (4) holds. If not, it returns $\perp$. Otherwise, it computes $R = \dfrac{C_2}{e(C_1, H_2(pk, w))^{sk}}$, $m = C_3 \oplus H_3(R)$, and check whether $g^{H_1(m,R)} = C_1$ holds. If yes, it returns m; else it returns $\perp$.

Dec$_1(\mathbf{CT}, sk)$**:** On input a private key sk and a first level ciphertext $\mathrm{CT} = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ under public key pk, it computes $R = \overline{C}_2 \cdot e(\overline{C}_1, \overline{C}_4)^{sk}$ and $m = \overline{C}_3 \oplus H_3(R)$. Return m if $g^{H_1(m,R)} = \overline{C}_1$ holds and $\perp$ otherwise:

Analysis. At first glance, it seems that scheme S1 is CCA-secure. Unfortunately, this is not true, since the adversary can maliciously manipulate the first level ciphertext to get a *new* yet *valid* one. Concretely, given the first level ciphertext as in Eq. (3), the adversary can pick $\ell \in_R \mathbb{Z}_q$ and produces another first level ciphertext $\mathrm{CT}' = (\overline{C}'_1, \overline{C}'_2, \overline{C}'_3, \overline{C}'_4)$ such that:

$$\overline{C}'_1 = \overline{C}_1 = g^r, \quad \overline{C}'_2 = \overline{C}_2 \cdot e(\overline{C}_1, pk)^{-\ell} = R \cdot e(g, pk)^{-r \cdot (\overline{s} + \ell)}.$$
$$\overline{C}'_3 = \overline{C}_3 = m_\delta \oplus H_3(R), \quad \overline{C}'_4 = \overline{C}_4 \cdot g^\ell = g^{\overline{s} + \ell}.$$

Letting $\overline{s} =' \overline{s} + \ell$, we can easily see that CT' is another new and valid ciphertext as Eq. (3). Thus the CCA-security can be trivially broken.

3.2 CCA-Secure C-PRE Scheme

Indeed, the insecurity of S1 lies in the construction of the re-encryption key, i.e., rk_2 is loosely integrated with rk_1. This enables the adversary to maliciously manipulate the resulting first level ciphertext and obtain another *valid* first level ciphertext. So, to design a CCA-secure C-PRE scheme, we should carefully design the re-encryption key, so that the resulting first level ciphertext cannot be maliciously manipulated by the adversary. Based on this observation, we present our CCA-secure C-PRE scheme (denoted by S2) as below:

Setup(1^κ) **and KeyGen**(1^κ)**:** The same as in S1.

ReKeyGen(sk_i, w, pk_j)**:** On input a private key sk_i, a condition w and a public key pk_j, this algorithm picks $s \in_R \mathbb{Z}_q$, and outputs $rk_{i \xrightarrow{w} j} = (rk_1, rk_2)$ as

$$rk_2 = pk_i^s, \quad rk_1 = \left(H_2(pk_i, w) pk_j^{s \cdot H_5(pk_j^{s \cdot sk_i})}\right)^{-sk_i}.$$

Observe that in the re-encryption key $rk_{i \xrightarrow{w} j}$, rk_2 is now *seamlessly* integrated with rk_1. That is, we integrate rk_2 with rk_1 by embedding $H_5(pk_j^{s \cdot sk_i}) = H_5(rk_2^{sk_j})$ in rk_1. This is an important trick for scheme S2 to achieve the CCA-security.

Enc$_2(pk, m, w)$**:** The same as in S1.

Enc$_1(pk, w)$**:** On input a public key pk and a message $m \in \mathcal{M}$, the sender first picks $R \in_R \mathbb{G}_T$ and $\overline{s} \in_R \mathbb{Z}_q^*$. Then he computes $r = H_1(m, R)$, and outputs the first level ciphertext $\mathrm{CT} = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ as

$$\left(g^r,\ R \cdot e(g, pk)^{-r \cdot \overline{s} \cdot H_5(pk^{\overline{s}})},\ m \oplus H_3(R),\ g^{\overline{s}}\right). \tag{6}$$

ReEnc$(\mathbf{CT}_i, rk_{i \xrightarrow{w} j})$**:** The same as in S1. Note that, since the re-encryption key is different from that in S1, the resulting first level ciphertext $\mathrm{CT}_j = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ is of the following forms:

$$\left(g^r,\ \ R \cdot e\,(g, pk_j)^{-r \cdot s \cdot sk_i \cdot H_5(pk_j^{s \cdot sk_i})},\ \ m \oplus H_3(R),\ \ g^{s \cdot sk_i}\right),$$

where $r = H_1(m, R)$ and $R \in_R \mathbb{G}_T$. Letting $\overline{s} = s \cdot sk_i$, it can be seen that the above first level ciphertext has the same form as Eq. (6).

Note also that, now $\overline{C}_4$ is tightly integrated with $\overline{C}_2$ by embedding $\overline{C}_4$ in $H_5(\overline{C}_4^{sk_j}) = H_5(pk_j^{s \cdot sk_i})$, and hence it is unable for the adversary to modify the first level ciphertext to obtain a new and *valid* one. Therefore, the attack against scheme S1 does not apply to scheme S2.

Dec$_2(\mathbf{CT}, sk)$**:** The same as in S1.

Dec$_1(\mathbf{CT}, sk)$**:** On input a private key sk and a first level ciphertext $\mathrm{CT} = (\overline{C}_1, \overline{C}_2, \overline{C}_3, \overline{C}_4)$ under public key pk, this algorithm first computes $R = \overline{C}_2 \cdot e(\overline{C}_1, \overline{C}_4)^{sk \cdot H_5(\overline{C}_4^{sk})}$ and $m = \overline{C}_3 \oplus H_3(R)$. Next, it returns m if $g^{H_1(m,R)} = \overline{C}_1$ holds and $\perp$ otherwise.

3.3 Security Analysis

The CCA-security of our schemes S2 is based on a complexity assumption called decisional Bilinear Diffie-Hellman (DBDH) assumption. The DBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$ is, given a tuple $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \in_R \mathbb{Z}_q$, to decide whether $Z = e(g, g)^{abc}$. A polynomial-time algorithm $\mathcal{B}$ has *advantage* ϵ in solving the DBDH problem in groups $(\mathbb{G}, \mathbb{G}_T)$, if

$$\left|\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^c, Z = e(g, g)^{abc}\right) = 1\right] - \Pr\left[\mathcal{B}\left(g, g^a, g^b, g^c, Z = e(g, g)^{d}\right) = 1\right]\right| \geq \epsilon,$$

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

Definition 3. *We say that the (t, ϵ)-DBDH assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$, if there exists no t-time algorithm $\mathcal{B}$ that has advantage ϵ in solving the DBDH problem in $(\mathbb{G}, \mathbb{G}_T)$.*

For our scheme's CCA-security at the second level, we have the following theorem, whose detailed proof can be found in Appendix B.

Theorem 1. *Our scheme* S2 *is* IND-2CPRE-CCA *secure in the random oracle model, assuming the DBDH assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$. More specifically, if there exists an* IND-2CPRE-CCA *adversary $\mathcal{A}$, who asks at most q_{H_i} random oracle queries to H_i with $i \in \{1, \cdots, 5\}$ and breaks the $(t, q_u, q_c, q_{rk}, q_{re}, q_d, \epsilon)$-*IND-2CPRE-CCA *security of scheme* S2*, then there exists an algorithm $\mathcal{B}$ that can break the (t', ϵ')-DBDH assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with*

$$\epsilon' \geq \frac{\epsilon}{\dot{e}(1 + q_{rk})} - \frac{q_{H_1} + q_{H_5} + q_{re} + q_d}{q},$$
$$t' \leq t + \mathcal{O}(\tau(q_{H_2} + q_{H_4} + q_u + q_c + 3q_{rk} + q_{H_1} q_{re} + (q_{H_1} + q_{H_5})q_d)),$$

where τ is the maximum over the time to compute an exponentiation in $\mathbb{G}$,$\mathbb{G}_T$, and the time to compute a pairing; $\dot{e}$ denotes the base of the natural logarithm.

The first level ciphertext security of S2 is ensured by the following theorem.

Theorem 2. *Our scheme* S2 *is* IND-1CPRE-CCA *secure in the random oracle model, assuming the DBDH assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$. More specifically, if there exists an* IND-1CPRE-CCA *adversary $\mathcal{A}$, who asks at most q_{H_i} random oracle queries to H_i with $i \in \{1, \cdots, 5\}$ and can break the $(t, q_u, q_c, q_{rk}, q_d, \epsilon)$-*IND-1CPRE-CCA *security of scheme* S2*, then there exists an algorithm $\mathcal{B}$ that can break the (t', ϵ')-DBDH assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with*

$$\epsilon' \geq \epsilon - \frac{q_{H_1} + q_{H_5} + q_d}{q},$$
$$t' \leq t + \mathcal{O}(\tau(q_{H_2} + q_{H_4} + q_u + q_c + 3q_{rk} + (q_{H_1} + q_{H_5})q_d)),$$

where τ and $\dot{e}$ have the same meaning as in Theorem 1.

The proof for Theorem 2 is similar to that of Theorem 1 with some modifications. For example, the simulation for the random oracle H_2 no longer need to flip a biased coin, and the simulation for oracle $\mathcal{O}_{rk}$ has to successfully answer all the re-encryption key queries without aborting. Due to the space limit, we give the detailed proof in the full paper.

3.4 Comparisons

In Table 1, we compare our scheme with Tang's scheme [5] [3], Weng *et al.*'s scheme [6] and Livert-Vergnaud's scheme [10]. We first explain some notations used in

[3] Tang presented two schemes: one is CPA-secure, and the other is CCA-secure. To be fair, we here choose Tang's CCA-secure scheme for comparison.

Table 1. Comparisons among Ours Scheme and the C-PRE Schemes in [5, 6, 4]

Schemes		Our Scheme S2	Tang's Scheme [5]	Weng's Scheme [6]	Livert-Vergnaud's Scheme [10]
Length	2nd-level ciphtxt	$2\|\mathbb{G}\|+1\|\mathbb{G}_T\|+1\|\mathcal{M}\|$	$2\|\mathbb{G}\|+1\|\mathbb{G}_T\|+1\|\mathcal{M}\|$	$3\|\mathbb{G}\|+1\|\mathcal{M}\|+l_1$	$\|svk\|+3\|\mathbb{G}\|+1\|\mathbb{G}_T\|+\|\sigma\|$
	1st-level ciphtxt	$2\|\mathbb{G}\|+1\|\mathbb{G}_T\|+1\|\mathcal{M}\|$	$2\|C_{\mathsf{PKE}}\|+1\|\mathbb{G}\|+1\|\mathbb{G}_T\|+1\|\mathcal{M}\|$	$1\|\mathbb{G}_T\|+1\|\mathcal{M}\|+l_1$	$\|svk\|+7\|\mathbb{G}\|+1\|\mathbb{G}_T\|+1\|\sigma\|$
	public key	$1\|\mathbb{G}\|$	$1\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$(n+2)\|\mathbb{G}\|$
	private key	$1\|\mathbb{Z}_q\|$	$1\|\mathbb{Z}_q\|$	$1\|\mathbb{Z}_q\|$	$1\|\mathbb{Z}_q\|$
	re-encryption key	$2\|\mathbb{G}\|$	$1\|C_{\mathsf{PKE}}\|+1\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$
Cost	Enc_2	$1t_p+3t_e$	$1t_p+3t_e$	$1t_p+5t_e$	$1t_s+4t_e$
	Enc_1	$1t_p+4t_e$	$1t_p+2t_e+2t_{\mathsf{Enc}_{\mathsf{PKE}}}$	$1t_p+2t_e$	$1t_s+8t_e$
	ReEnc	$3t_p$	$3t_p+1t_{\mathsf{Enc}_{\mathsf{PKE}}}$	$3t_p+2t_e$	$4t_p+6t_e$
	Dec_2	$3t_p+2t_e$	$3t_p+2t_e$	$4t_p+5t_e$	$1t_p+1t_e+1t_v$
	Dec_1	$1t_p+3t_e$	$2t_{\mathsf{Dec}_{\mathsf{PKE}}}+1t_p+1t_e$	$2t_e$	$9t_p+1t_e+1t_v$
Security		CCA	CCA	Not CCA	RCCA
Without RO?		No	No	No	Yes

Table 1. Here $|\mathcal{M}|$, $|\mathbb{G}|$, $|\mathbb{G}_T|$, $|svk|$ and $|\sigma|$ denote the bit-length of a plaintext, an element in groups $\mathbb{G}$ and $\mathbb{G}_T$, the verification key and signature of one-time signature, respectively. We use t_p, t_e, t_s, t_v to represent the computational cost of a bilinear pairing, an exponentiation, signing and verifying a one-time signature, respectively. l_1 denotes the security parameter used in Weng *et al.*'s scheme. Tang's scheme needs an additional public key encryption scheme PKE, which is assumed to be deterministic and one-way[4]. We here use $t_{\mathsf{Enc}_{\mathsf{PKE}}}$ and $t_{\mathsf{Dec}_{\mathsf{PKE}}}$ to represent the computational cost of an encryption and a decryption in the public key encryption(PKE) scheme used in Tang's scheme. For $|C_{\mathsf{PKE}}|$, it denotes the ciphertext length of scheme PKE used in Tang's scheme.

The comparison results indicate that our scheme S2 outperforms Tang's scheme in terms of both computational and communicational costs. Our scheme has a better overall performance than Weng *et al.*'s scheme: The ciphertext length and computation cost for first level encryption and decryption in Weng *et al.*'s scheme lead ours, while ours beats theirs in the other metrics; most importantly, our scheme is CCA-secure, while theirs fails. Our scheme also has a better overall performance than Libert-Vergnaud's scheme. Besides, ours is CCA-secure under the well-studied DBDH assumption, while Libert-Vergnaud's scheme only satisfies the RCCA-security (which is a weaker variant of CCA-security assuming a harmless mauling of the challenge ciphertext is tolerated) under a less studied assumption, named 3-weak decisional bilinear Diffie-Hellman inversion (3-wDBDH) assumption. However, like Tang and Weng *et al.*'s schemes, our scheme suffers from a limitation that its security relies on the random oracle in the know secret key model, while Libert-Vergnaud's scheme can be proved without random oracles in the chosen-key model.

4 Conclusions

We re-formalized the definition and security notions for conditional proxy re-encryption (C-PRE), and proposed an efficient CCA-secure C-PRE scheme un-

[4] To the best of our knowledge, the ciphertext in such a PKE scheme needs at least two group elements, and its computational cost for encryption and decryption involves at least two exponentiations and one exponentiation respectively. Hence, we have $|C_{\mathsf{PKE}}| \geq 2|\mathbb{G}|$, $t_{\mathsf{Enc}_{\mathsf{PKE}}} \geq 2t_e$, $t_{\mathsf{Dec}_{\mathsf{PKE}}} \geq 1t_e$.

der our model. In addition, we gave an attack to Weng *et al.*'s C-PRE scheme, showing that it fails to achieve the CCA-security.

This work motivates some interesting open questions. One is how to construct a CCA-secure (instead of RCCA-secure) C-PRE scheme without random oracles. Another is how to construct CCA-secure C-PRE schemes supporting "OR" and "AND" gates over conditions.

Acknowledgement

We are grateful to the anonymous reviewers for their helpful comments. This work is partially supported by the Office of Research, Singapore Management University.

References

1. Blaze, M., Bleumer, G., Strauss, M.: Divertible Protocols and Atomic Proxy Cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In: NDSS, The Internet Society (2005)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)
4. 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)
5. Tang, Q.: Type-based proxy re-encryption and its construction. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 130–144. Springer, Heidelberg (2008)
6. Weng, J., Deng, R.H., Ding, X., Chu, C.K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: ASIACCS, pp. 322–332 (2009)
7. Mambo, M., Okamoto, E.: Proxy cryptosystems: delegation of the power to decrypt ciphertexts. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E80-A(1), 54–63 (1997)
8. Canetti, R., Hohenberger, S.: Chosen-Siphertext Cecure Proxy Re-Encryption. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM Conference on Computer and Communications Security, pp. 185–194. ACM, 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. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption, `http://hal.inria.fr/inria-00339530/en/`, This is the extended version of [4]
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. Shao, J., Cao, Z.: CCA-Secure Proxy Re-encryption without Pairings. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
13. Weng, J., Chow, S.S., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. Cryptology ePrint Archive, Report 2009/189 (2009), http://eprint.iacr.org/
14. 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)
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. 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)
17. 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)
18. Chu, C.K., Weng, J., Chow, S.S.M., Zhou, J., Deng, R.H.: Conditional proxy broadcast re-encryption. In: ACISP, pp. 327–342 (2009)
19. Weng, J., Deng, R.H., Liu, S., Chen, K., Lai, J., Wang, X.: Chosen-ciphertext secure proxy re-encryption without pairings. Cryptology ePrint Archive, Report 2008/509 (2008), http://eprint.iacr.org/, This is the full paper of [11]
20. Gamal, T.E.: A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985)
21. Fujisaki, E., Okamoto, T.: Secure Integration of Asymmetric and Symmetric Encryption Schemes. In: Wiener, M.J. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999)
22. Coron, J.S.: On the Exact Security of Full Domain Hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (2000)

Appendix

A Cryptanalysis of Weng *et al.*'s C-PRE Scheme

In this section, we will explain why Weng *et al.*'s C-PRE scheme [6] fails to achieve the CCA-security. Due to the space limit, here we only give a brief review of the scheme (please refer to [6] for the detailed scheme and the corresponding security notions). In Weng *et al.*'s scheme, a user's private key for the user is $sk = x \in \mathbb{Z}_q^*$, and his public key is $pk = (g^x, g_1^{1/x})$. The re-encryption key, from one public key $pk_i = (g^{x_i}, g_1^{1/x_i})$ to another public key $pk_j = (g^{x_j}, g_1^{1/x_j})$ associated with condition w, consists of two parts: a partial re-encryption key $rk_{i,j} = g^{x_j/x_i}$ and a condition key $ck_{i,w} = H_3(w, pk_i)^{1/x_i}$. A second level ciphertext $\mathrm{CT}_i = (A, B, C, D)$ under pk_i is

$$\left(g_1^r,\ \ (g^{x_i})^r\,,\ H_2\left(e(g,g)^r\right) \oplus (m \| r') \oplus H_4\left(e(Q_i, H_3(w, pk_i))^r\right),\ H_5(A, B, C)^r\right),$$

while a first level ciphertext $\mathrm{CT}_j = (B', C)$ re-encrypted from pk_i to pk_j is

$$\left(e(g, g^{sk_j})^r,\ \ H_2(e(g,g)^r) \oplus (m \| r')\right).$$

According to the security model defined in [6], for a target public key pk_{i^*} and a target condition w^*, even if the adversary has corrupted another user's secret key sk_j, he is still allowed to obtain one (not *both*) of the partial re-encryption key $rk_{i^*,j}$ and the condition key ck_{i^*,w^*}. Now, we explain how an adversary can break the CCA-security of Weng *et al.*'s scheme: she first obtains $sk_j = x_j$ and $rk_{i^*,j} = g^{x_j/x_{i^*}}$, and then computes $g^{1/x_{i^*}} = \left(g^{x_j/x_{i^*}}\right)^{1/x_j}$. Next, she calculates $e(g,g)^r$ as $e\left((g^{x_{i^*}})^r, g^{1/x_{i^*}}\right)$, where $(g^{x_{i^*}})^r$ is exactly the second component of the second level ciphertext. Using $e(g,g)^r$, she can certainly decrypt the first level ciphertext to obtain the underlying plaintext.

B Security Proof for Theorem 1

Proof. Suppose algorithm $\mathcal{B}$ is given a DBDH instance $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \in_R \mathbb{Z}_q$. $\mathcal{B}$'s goal is to decide whether $Z = e(g,g)^{abc}$. $\mathcal{B}$ works by interacting with adversary $\mathcal{A}$ in the IND-2CPRE-CCA game as follows:

`Initialize Stage.` $\mathcal{B}$ gives $param = ((q, \mathbb{G}, \mathbb{G}_T, e), g, n, H_1, \cdots, H_5)$ to $\mathcal{A}$. Here $H_1, \cdots, H_5$ are the random oracles controlled by $\mathcal{B}$ and can be adaptively asked by $\mathcal{A}$ at any time. $\mathcal{B}$ maintains five hash lists H_i^{list} with $i \in \{1, \cdots, 5\}$, which are initially empty, and responds the random oracle queries for $\mathcal{A}$ as shown in Figure 1.

- $H_1(m, R)$: If this query already appears on H_1^{list} in a tuple (m, R, r), return r. Otherwise, choose $r \in_R \mathbb{Z}_q$, add the tuple (m, R, r) to the H_1^{list} and respond with $H_1(m, R) = r$.
- $H_2(pk_i, w)$: If this query already appears on the H_2^{list}, then return the predefined value. Otherwise, choose $\mu, \mu' \in_R \mathbb{Z}_q$, and use the Coron's proof technique [22] to flip a biased coin $\mathsf{coin}_i \in \{0,1\}$ that yields 1 with probability θ and 0 with probability $1-\theta$. If $\mathsf{coin}_i = 0$, define $H_2(pk_i, w) = g^{\mu} \cdot (g^b)^{-\mu'}$; otherwise, define $H_2(pk_i, w) = g^{\mu+\mu'}$. Finally, add the tuple $(pk_i, w, \mathsf{coin}_i, \mu, \mu')$ to the list H_2^{list} and respond with $H_2(pk_i, w)$.
- $H_3(R)$: If this query already appears on the H_3^{list}, then return the predefined value. Otherwise, choose $\omega \in_R \{0,1\}^n$, add the tuple (R, ω) to the H_3^{list} and respond with $H_3(R) = \omega$.
- $H_4(C_1, C_2, C_3)$: If this query already appears on the H_4^{list}, then return the predefined value. Otherwise, choose $\gamma \in_R \mathbb{Z}_q$, add the tuple (C_1, C_2, C_3, γ) to the H_4^{list} and respond with $H_4(C_1, C_2, C_3) = g^{\gamma}$.
- $H_5(V)$: If this query already appears on the H_5^{list}, then return the predefined value. Otherwise, choose $\lambda \in_R \mathbb{Z}_q$, add the tuple (V, λ) to the H_5^{list} and respond with $H_5(V) = \lambda$.

Fig. 1. The Simulations for H_i for $i = 1, \cdots, 5$

`Find Stage.` In this stage, adversary $\mathcal{A}$ issues a series of queries subject to the restrictions of the IND-2CPRE-CCA game. $\mathcal{B}$ maintains a list K^{list} which is initially empty, and answers these queries for $\mathcal{A}$ as follows:

- Uncorrupted key generation oracle $\mathcal{O}_u(i)$: Algorithm $\mathcal{B}$ first picks $x_i \in_R \mathbb{Z}_q$, and defines $pk_i = (g^a)^{x_i}$. Next, it sets $c_i = 0$ and adds the tuple (pk_i, x_i, c_i) to the K^{list}. Finally, it returns pk_i to adversary $\mathcal{A}$.
- Corrupted key generation oracle $\mathcal{O}_c(j)$: $\mathcal{B}$ first picks $x_j \in_R \mathbb{Z}_q$ and defines $pk_j = g^{x_j}$ and $c_j = 1$. Next, it adds the tuple (pk_j, x_j, c_j) to the K^{list} and returns (pk_j, x_j) to adversary $\mathcal{A}$.

– Re-encryption key oracle $\mathcal{O}_{rk}(pk_i, w, pk_j)$: $\mathcal{B}$ first recovers $(pk_i, w, \mathsf{coin}_i, \mu, \mu')$ from the H_2^{list} and tuples (pk_i, x_i, c_i) and (pk_j, x_j, c_j) from the K^{list}. Next, it constructs the re-encryption key $rk_{i \xrightarrow{w} j}$ for adversary $\mathcal{A}$ according to the following situations:
  - Case 1: $c_i = 1$, it means that $sk_i = x_i$. Using sk_i, $\mathcal{B}$ can certainly generate the re-encryption key $rk_{i \xrightarrow{w} j}$ for $\mathcal{A}$ as in algorithm $\mathsf{ReKeyGen}$.
  - Case 2: $(c_i = 0 \wedge c_j = 1 \wedge \mathsf{coin}_i = 1)$, it means that $sk_i = ax_i$, $sk_j = x_j$ and $H_2(pk_i, w) = g^{\mu+\mu'}$. $\mathcal{B}$ picks $s \in_R \mathbb{Z}_q$, computes $rk_2 = pk_i^s, rk_1 = (g^a)^{-(\mu+\mu'+x_j \cdot s \cdot H_5((g^a)^{x_i \cdot s \cdot x_j}))x_i}$ and returns (rk_1, rk_2) to $\mathcal{A}$. Observe that this is indeed a valid re-encryption key, since
$$rk_1 = (g^a)^{-(\mu+\mu'+x_j \cdot s \cdot H_5((g^a)^{x_i \cdot s \cdot x_j}))x_i} = \left(g^{\mu+\mu'+sk_j \cdot s \cdot H_5(pk_j^{s \cdot sk_i})}\right)^{-a \cdot x_i}$$
$$= \left(g^{\mu+\mu'} g^{sk_j \cdot s \cdot H_5(pk_j^{s \cdot sk_i})}\right)^{-sk_i} = \left(H_2(pk_i, w) pk_j^{s \cdot H_5(pk_j^{s \cdot sk_i})}\right)^{-sk_i}.$$
  - Case 3: $(c_i = 0 \wedge c_j = 0 \wedge \mathsf{coin}_i = 1)$, it means that $sk_i = ax_i$, $sk_j = ax_j$ and $H_2(pk_i, w) = g^{\mu+\mu'}$. $\mathcal{B}$ picks $s' \in_R \mathbb{Z}_q$, computes $rk_2 = g^{x_i s'}, rk_1 = (g^a)^{-(\mu+\mu'+x_j s' \cdot H_5(pk_j^{s' \cdot x_i}))x_i}$, and returns (rk_1, rk_2) to $\mathcal{A}$. Observe that, letting $s = \frac{s'}{a}$, one can see that it is indeed a valid re-encryption key.
  - Case 4: $(c_i = 0 \wedge c_j = 0 \wedge \mathsf{coin}_i = 0)$, it means that $sk_i = ax_i$, $sk_j = ax_j$ and $H_2(pk_i, w) = g^{\mu} \cdot (g^b)^{-\mu'}$. $\mathcal{B}$ picks $s \in_R \mathbb{Z}_q$, computes $rk_2 = pk_i^s$, $rk_1 = pk_i^{-u}$, and returns returns $rk_{i \xrightarrow{w} j} = (rk_1, rk_2)$ to $\mathcal{A}$. Observe that, if implicitly let $H_5(pk_j^{s \cdot sk_i}) = \frac{b \cdot \mu'}{s \cdot a \cdot x_j}$ (note that $pk_j^{s \cdot sk_i}$ is unknown to $\mathcal{A}$, since sk_i, sk_j and s are all unknown to him), we can easily see that this is indeed a valid re-encryption key as required.
  - Case 5: $(c_i = 0 \wedge c_j = 1 \wedge \mathsf{coin}_i = 0)$, $\mathcal{B}$ outputs $\beta' \in_R \{0, 1\}$ and **aborts**.

– Re-encryption oracle $\mathcal{O}_{re}(pk_i, pk_j, (w, \text{CT}_i))$: $\mathcal{B}$ parses $\text{CT}_i = (C_1, C_2, C_3, C_4)$. If Eq. (4) does not hold, it outputs $\perp$; otherwise, it works as follows:
  1. Recover (pk_i, x_i, c_i) and (pk_j, x_j, c_j) from the K^{list} and $(pk_i, w, \mathsf{coin}_i, \mu, \mu')$ from the H_2^{list}.
  2. If $(c_i = 0 \wedge c_j = 1 \wedge \mathsf{coin}_i = 0)$ does not hold, then $\mathcal{B}$ can construct the re-encryption key $rk_{i \xrightarrow{w} j}$ as in the re-encryption key query, and then can certainly generate the first level ciphertext CT_j for $\mathcal{A}$.
  3. Otherwise, it implies that $c_j = 1$, i.e., $sk_j = x_j$. In this case, $\mathcal{B}$ picks $s \in_R \mathbb{Z}_q$ and generates the first level ciphertext as follows: search whether there exists a tuple $(m, R, r) \in H_1^{\text{list}}$ such that $g_1^r = C_1, R \cdot e(pk_i, H_2(pk_i, w))^r = C_2, m \oplus H_3(R) = C_3$ and $H_4(C_1, C_2, C_3)^r = C_4$ hold. If yes, pick $s \in_R \mathbb{Z}_q$, compute $\overline{C}_4 = pk_i^s$, $\overline{C}_2 = R \cdot e\big(C_1, pk_i^{s \cdot H_5(\overline{C}_4^{x_j})}\big)^{-x_j}$, and return $\text{CT}_j = (C_1, \overline{C}_2, C_3, \overline{C}_4)$ as the first level ciphertext to $\mathcal{A}$; otherwise return $\perp$. Note that we can store s in a table to keep the consistency of s for the same re-encryption queries $\mathcal{O}_{re}(pk_i, pk_j, (w, *))$.

– First level decryption oracle $\mathcal{O}_{1d}(pk_j, \mathrm{CT})$: $\mathcal{B}$ first recovers (pk_j, x_j, c_j) from the K^{list}. If $c_j = 1$ (meaning $sk_j = x_j$), $\mathcal{B}$ decrypts the ciphertext using sk_j and returns the plaintext to $\mathcal{A}$. Otherwise, it searches H_1^{list} and H_5^{list} to see whether there exist a tuple $(m, R, r) \in H_1^{\mathrm{list}}$ and a tuple $(V, \lambda) \in H_5^{\mathrm{list}}$ such that $g^r = \overline{C}_1, R \cdot e\left(\overline{C}_4, pk_j\right)^{-r \cdot \lambda} = \overline{C}_2, m \oplus H_3(R) = C_3$ and $e(V, g) = e(\overline{C}_4, pk_j)$. If yes, return m to $\mathcal{A}$; else return $\perp$.

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

1. Recover $(pk_{i^*}, x_{i^*}, c_{i^*})$ from the K^{list} and $(pk_{i^*}, w^*, \mathsf{coin}_{i^*}, \mu, \mu')$ from the H_2^{list}. If $\mathsf{coin}_{i^*} = 1$, output a random bit $\beta' \in_R \{0,1\}$ and **aborts**. Otherwise, it means that $H_2(pk_{i^*}, w^*) = g^{\mu} \cdot (g^b)^{-\mu'}$.
2. Flip a random coin $\delta \in_R \{0,1\}$ and pick $R^* \in_R \mathbb{G}_T$. Compute $C_1^* = g^c$, $C_2^* = R^* \cdot Z^{-\mu' \cdot x_{i^*}} \cdot e(g^a, g^c)^{x_{i^*}\mu}$ and $C_3^* = m_\delta \oplus H_3(R^*)$.
3. Issue an H_4 query on (C_1^*, C_2^*, C_3^*) to obtain the tuple $(C_1^*, C_2^*, C_3^*, \gamma^*)$, and define $C_4^* = (g^c)^{\gamma^*}$.
4. Finally, give $\mathrm{CT}^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ to $\mathcal{A}$.

Note that by the above construction, if $Z = e(g,g)^{abc}$, CT^* is indeed a valid ciphertext for m_δ under pk_{i^*} and w^*. To see this, implicit letting $H_1(m_\delta, R^*) = c$, we have

$$
\begin{aligned}
C_2^* &= R^* \cdot Z^{-\mu' \cdot x_{i^*}} \cdot e(g^a, g^c)^{x_{i^*}\mu} = R^* \cdot e(g,g)^{-\mu' \cdot abc \cdot x_{i^*}} \cdot e(g^a, g^c)^{x_{i^*}\mu} \\
&= R^* \cdot e(g^{a \cdot x_{i^*}}, g^{\mu} g^{-\mu' \cdot b})^c = R^* \cdot e(pk_{i^*}, H_2(pk_{i^*}, w^*))^c, \\
C_1^* &= g^c, \quad C_3^* = m_\delta \oplus H_3(R^*), \quad C_4^* = (g^c)^{\gamma^*} = \left(g^{\gamma^*}\right)^b = H_4(C_1^*, C_2^*, C_3^*)^c.
\end{aligned}
$$

On the other hand, when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext CT^* is independent of δ in the adversary's view.

`Guess Stage.` $\mathcal{A}$ continues to issue the rest of queries as in `Find` stage, with the restrictions described in the IND-2CPRE-CCA game. $\mathcal{B}$ responds to these queries as in `Find` stage.

`Output Stage.` Eventually, adversary $\mathcal{A}$ returns a guess $\delta' \in \{0,1\}$ to $\mathcal{B}$. If $\delta' = \delta$, $\mathcal{B}$ outputs $\beta' = 1$; otherwise, $\mathcal{B}$ outputs $\beta' = 0$.

This completes the description of the simulation. Due to space limit, in the full paper, we will show that $\mathcal{B}$'s advantage against the DBDH assumption is at least $\epsilon' \geq \frac{\epsilon}{\hat{e}(1+q_{rk})} - \frac{q_{H_1}+q_{H_5}+q_{re}+q_d}{q}$, and $\mathcal{B}$'s running time is bounded by $t' \leq t + \mathcal{O}(\tau(q_{H_2} + q_{H_4} + q_{\mathrm{u}} + q_{\mathrm{c}} + 3q_{rk} + q_{H_1}q_{re} + (q_{H_1} + q_{H_5})q_d))$. □