Identity-Based Proxy Re-encryption Schemes with Multiuse, Unidirection, and CCA Security

Jun Shao, Dongsheng Xing, and Zhenfu Cao
chn.junshao@gmail.com; dsdsdds@sjtu.edu.cn; zfcao@cs.sjtu.edu.cn
Department of Computer Science and Engineering, Shanghai Jiao Tong University

Abstract

A proxy re-encryption (PRE) scheme allows a proxy to transform a ciphertext under Alice's public key into a ciphertext under Bob's public key on the same message. In 2006, Green and Ateniese extended the above notion to identity-based proxy re-encryption (IB-PRE), and proposed two open problems [7]: building 1. IB-PRE schemes which are CCA-secure in the standard model; 2. multi-use CCA-secure IB-PRE schemes. Chu and Tzeng proposed two identity-based proxy re-encryption schemes afterwards in [6], aiming at solving the two open problems. However, in this paper, we show that they don't solve these two open problems by revealing a security flaw in their solution. Furthermore, we propose an improvement which is a solution to the open problems.

1 Introduction

A proxy re-encryption (PRE) scheme allows a proxy to transform a ciphertext under Alice's public key into a ciphertext under Bob's public key on the same message. This concept is proposed by Blaze, Bleumer, and Strauss [3], and formalized later by Ateniese et al. [1, 2]. In 2006, Green and Ateniese extended the above notion to identity-based proxy re-encryption (IB-PRE) in [7]. The IB-PRE scheme has both the benefits of proxy re-encryption schemes and identity-based encryption schemes. In this kind of scheme, the user's public key is his/her identity information, and the user's private key is generated by a trusted party Private Key Generator (PKG) according to his/her identity information. Hence, the encryptor encrypts the plaintext with the decryptor's identity information, and the proxy can transform a ciphertext under Alice's identity into a ciphertext under Bob's identity on the same message. Furthermore, in [7], the authors proposed two open problems on IB-PRE schemes: building

- IB-PRE schemes which are CCA-secure in the standard model;
- multi-use CCA-secure IB-PRE scheme.

To solve the above two problems, Chu and Tzeng proposed two IB-PRE schemes afterwards in [6], which are based on Waters' identity-based encryption scheme [8]. They showed that the second scheme is the solution to the two open problems. However, their second scheme is not CCA-secure due to a security flaw, such that one valid ciphertext can be modified to another ciphertext. Though the two ciphertexts are different, they are both valid. The reason is that a ciphertext is a tuple including two components $d'_1 = \hat{d}_1 f^{r'_1}$ and $d'_2 = g^{r'_2}$, where any adversary can get the knowledge of f and g, and can modify d'_1 and d'_2 by multiplying f^{r_2} and g^{r_2}, respectively. We will give the details in Section 4. In this paper, we propose an improvement to eliminate the security flaw.

Paper Outline. In Section 2, we present the definitions which will be used in the rest of paper. In Section 3, we review the second scheme of Chu and Tseng, and give an analysis. In what follows, we propose our improvement and its security proof. Finally, we conclude the paper in Section 5.

2 Basic Definitions

In order to prepare for the statements in the following sections, we review some definitions here. The similar definitions can be found in [4, 5, 7, 6].

2.1 Admissible bilinear map

Let $\mathbb{G}$, $\mathbb{G}_1$ be two groups of prime order p, g be a generator of $\mathbb{G}$. Then a function $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is an admissible bilinear map if e meets the following conditions: for any $g_1, g_2 \in \mathbb{G}$, $a \in \mathbb{Z}_p$ and $b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ is efficiently computable; for any generator g, h of $\mathbb{G}$, $e(g, h)$ generates $\mathbb{G}_1$.

2.2 Identity Based Encryption (IBE) Scheme

An Identity Based Encryption scheme consists of four algorithms:

- The `Setup` algorithm takes a security parameter and generates system parameters μ and a master key mk. The system parameters are publicly released. The master key is given to the Private Key Generator (PKG) only.
- The `Extract` algorithm, which is run by the PKG, uses system parameters μ and the master key mk to generate the private key d_{ID} corresponding to a given identity ID.
- The `Encrypt` algorithm uses the system parameters μ and a given identity ID to encrypt a plaintext.
- The `Decrypt` algorithm uses the system parameters μ and a given private key d_{ID} to decrypt ciphertext.

2.3 Negligible Function

A function $\varepsilon(n) : \mathbb{N} \mapsto \mathbb{R}$ is negligible in n if $\dfrac{1}{\varepsilon(n)}$ is a non-polynomially-bounded quantity in n.

2.4 Security Model for IBE

An IBE scheme is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if there is no polynomial time adversary $\mathcal{A}$ which wins the following game against the challenger $\mathcal{C}$ with non-negligible advantage $\mathrm{Adv}(\mathcal{A})$:

- The challenger $\mathcal{C}$ selects a security parameter λ and runs $\texttt{Setup}(1^\lambda)$, then $\mathcal{C}$ keeps the master key himself and gives the public parameter to $\mathcal{A}$.
- $\mathcal{A}$ does Extract or Decrypt queries at most polynomial times. For each query, $\mathcal{A}$ may choose the query arguments depending on her previous queries. $\mathcal{C}$ responds a query as follows:
  - Extract(ID). $\mathcal{C}$ responds to $\mathcal{A}$ by running the algorithm $\texttt{Extract}(\mu, mk, ID)$ to generate the private key corresponding to ID, and $\mathcal{C}$ returns the private key to $\mathcal{A}$.

  - DECRYPT(ID, C_{ID}). $\mathcal{C}$ runs $\texttt{Extract}(\mu, mk, ID)$ to generate the private key d_{ID}, then $\mathcal{C}$ runs $\texttt{Decrypt}(\mu, d_{ID}, C_{ID})$ and $\mathcal{C}$ returns the result to $\mathcal{A}$.

- $\mathcal{A}$ submits two messages m_0, m_1 and an identity ID^* to $\mathcal{C}$. $\mathcal{C}$ selects b randomly from $\{0, 1\}$. Then $\mathcal{C}$ gives $C_{ID^*} := \texttt{Encrypt}(ID^*, m_b)$ to $\mathcal{A}$.

- $\mathcal{A}$ continues making EXTRACT or DECRYPT queries at most polynomial times, and $\mathcal{A}$ is forbidden to query EXTRACT(ID^*) or DECRYPT(ID^*, C_{ID^*}).

- $\mathcal{A}$ outputs $b' \in \{0, l\}$ to $\mathcal{C}$ and $\mathcal{A}$ wins the game if $b' = b$.

The advantage of the adversary is $\text{Adv}(\mathcal{A})=2|\Pr[b' = b] - \frac{1}{2}|$.

An IBE scheme is semantically secure against an adaptive chosen plaintext attack (IND-ID-CPA) if there is no polynomial time adversary $\mathcal{A}$, which cannot do the DECRYPT queries in the above game, and wins the game with non-negligible advantage $\text{Adv}(\mathcal{A})$ in λ.

2.5 Identity Based Proxy Re-encryption (IB-PRE) Scheme

An IB-PRE scheme consists of the following algorithms:

- Given a security parameter 1^λ, the `Setup` algorithm returns a public parameter μ and a master key mk.

- Given μ, the master key mk and an n-bit identity ID, the `Extract` algorithm returns the decryption key d_{ID} corresponding to the identity.

- Given μ, an n-bit identity ID and a plaintext m, the `Encrypt` algorithm returns the ciphertext C_{ID}, which is the encryption of m under the specified identity ID.

- Given μ, two identities (ID_1 and ID_2) and a decryption key d_{ID_1} corresponding to ID_1, the `RKGen` algorithm returns a re-encryption key $d_{ID_1 \to ID_2}$.

- Given a re-encryption key $d_{ID_1 \to ID_2}$ and a ciphertext C_{ID_1} which is under identity ID_1, the `Reencrypt` algorithm returns the ciphertext C_{ID_2} which is under identity ID_2.

- Given a decryption key d_{ID} and a ciphertext C_{ID}, the `Decrypt` algorithm returns the plaintext or $\perp$.

An IB-PRE scheme may have the following properties:

- *Unidirection*: The re-encryption key can be only used to transform the ciphertext from Alice to Bob, but not vice versa.

- *Non-Interactivity*: A non-interactive scheme permits user A to compute a re-encryption key $d_{A \to B}$ without the participation of B or the Private Key Generator.

- *Multi-use*: A multi-use scheme allows the proxy to re-encrypt re-encrypted ciphertexts.

2.6 Security Model for Identity Based Proxy Re-encryption scheme

The security of an IB-PRE scheme is according to the game $Exp^{\mathcal{A},\texttt{IND-PrID-ATK}}$ between an challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, where $ATK \in \{CPA, CCA\}$ [7, 6].

Setup. The challenger $\mathcal{C}$ selects λ and runs $\texttt{Setup}(1^\lambda)$ to get (μ, mk), then $\mathcal{C}$ gives μ to $\mathcal{A}$.

Query phase 1. $\mathcal{A}$ makes the following queries.

1. EXTRACT(ID): $\mathcal{C}$ returns $d_{ID} = \texttt{Extract}(\mu, mk, ID)$ to $\mathcal{A}$.
2. RKEXTRACT(ID_1, ID_2): $\mathcal{C}$ returns $d_{ID_1 \to ID_2} = \texttt{RKGen}(\mu, d_{ID_1}, ID_1, ID_2)$ to $\mathcal{A}$, where $d_{ID_1} = \texttt{Extract}(\mu, mk, ID_1)$.

   If ATK is CCA, $\mathcal{A}$ can make the additional queries:

3. REENCRYPT(ID_1, ID_2, C_{ID_1}):
   $\mathcal{C}$ returns $C_{ID_2} = \texttt{Reencrypt}(\mu, d_{ID_1 \to ID_2}, ID_1, ID_2)$ to $\mathcal{A}$, where
   $d_{ID_1} = \texttt{KeyGen}(\mu, mk, ID_1)$, and $d_{ID_1 \to ID_2} = \texttt{RKGen}(\mu, d_{ID_1}, ID_1, ID_2)$.
4. DECRYPT(ID, C_{ID}): $\mathcal{C}$ returns $m = \texttt{Decrypt}(\mu, d_{ID_1}, C_{ID})$ to $\mathcal{A}$.

Challenge. $\mathcal{A}$ presents (ID^*, m_0, m_1) to $\mathcal{C}$. If the queries

- EXTRACT(ID^*); and
- RKEXTRACT(ID^*, ID') and EXTRACT(ID') for any identity ID',

are never made, $\mathcal{C}$ randomly chooses $b \in \{0, 1\}$, and returns $C^* := \texttt{Encrypt}(\mu, ID^*, m_b)$ to $\mathcal{A}$.

Query phase 2. $\mathcal{A}$ continues making same kinds of queries as in the **Query phase 1**, except for the following queries:

- EXTRACT(ID^*);
- RKEXTRACT(ID^*, ID') and EXTRACT(ID') for any identity ID';
- RKEXTRACT(ID^*, ID') and DECRYPT($ID', C_{ID'}$) for any identity ID' and any ciphertext $C_{ID'}$;
- REENCRYPT(ID^*, ID', C^*) and EXTRACT(ID') for any identity ID';
- DECRYPT(ID^*, C_{ID^*});
- DECRYPT($ID', C_{ID'}$) for any identity ID', where $C_{ID'} :=$ REENCRYPT(ID^*, ID', C^*).

Guess. $\mathcal{A}$ outputs $b' \in \{0, 1\}$ and she wins the game if $b' = b$.

The advantage of $\mathcal{A}$ is $\text{Adv}_{\mathcal{A}}^{\texttt{IND-PrID-ATK}} = |\Pr[b' = b] - \frac{1}{2}|$.

An IB-PRE scheme is IND-PrID-ATK ($ATK \in \{CPA, CCA\}$) secure, if for all probabilistic polynomial time algorithm $\mathcal{A}$ and negligible function ε in λ, $\text{Adv}_{\mathcal{A}}^{\texttt{IND-PrID-ATK}} \leq \varepsilon(\lambda)$.

3 Analysis on Chu and Tzeng's IB-PRE-II scheme

In [6], based on Wa-CCA-IBE [8], Chu and Tzeng proposed the IB-PRE-II scheme, and showed that if there is an adversary $\mathcal{A}$ that has advantage ϵ against the game $Exp^{\mathcal{A},\texttt{IND-PrID-CCA}}$, then there is an algorithm $\mathcal{B}$ that breaks Wa-CCA-IBE with advantage at least $\text{Adv}_{\mathcal{B}}^{\texttt{IND-ID-CCA}} \geq \epsilon/e(1 + q_E)$, where q_E is the maximal number of $\mathcal{A}$'s queries to EXTRACT. On the other hand, Wa-CCA-IBE is CCA-secure based on DBDH assumption [8]. Hence, the IB-PRE-II scheme is semantically secure against IND-PrID-CCA.

However, in this section, we show that their claim is not true. To explain this, we first introduce Wa-CCA-IBE and IB-PRE-II, then we show that why their proof fails.

3.1 Wa-CCA-IBE

For brevity, in the following sections, we use

- "$b \leftarrow_R A$" instead of "uniformly random choose an element b from a set A",
- "$(d_1, d_2) \leftarrow d$" instead of "d is a two-components tuple, decompose d, let d_1 be the first component and d_2 be the second component",
- "$a||0$" instead of "a is a binary string and append a bit 0 at the end of a",
- "$a := e$" instead of "the value of a is e", where e is an expression.

The Wa-CCA-IBE is as follows:

- $\texttt{Setup}(1^\lambda)$:
  - $p \leftarrow_R \mathbb{Z}$ and p is a prime.
  - Choose two groups $\mathbb{G}$, $\mathbb{G}_1$ with prime order p, and an admissible bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1$.
  - Let g be a generator of $\mathbb{G}$.
  - $\alpha \leftarrow_R \mathbb{Z}_p$, $g_1 = g^\alpha$, $g_2 \leftarrow_R \mathbb{G}$, $u'_1, u_{1,1}, \ldots, u_{1,n}, u'_2, u_{2,0}, u_{2,1}, \ldots, u_{2,n} \leftarrow_R \mathbb{G}$.
  - For any n-bit strings ID, let $\mathcal{V}_{ID}$ be the set of all i for which the i-th bit of ID is one.
  - $F_1(ID) \stackrel{def}{=} u'_1 \prod_{i \in \mathcal{V}_{ID}} u_{1,i}$, $F_2(ID) \stackrel{def}{=} u'_2 u_{2,0} \prod_{i \in \mathcal{V}_{ID}} u_{2,i}$.
  - Choose $\lambda' \in \mathbb{Z}$ and $\lambda' > 0$.
  - Choose a one-time signature scheme $(\mathcal{G}, \texttt{Sign}, \texttt{Vrfy})$ which satisfies $\forall (vk, sk), (vk, sk) \leftarrow \mathcal{G}(1^{\lambda'}) : vk \in \{0,1\}^n$.
  - $\mu := \big(g, g_1, g_2, F_1(\cdot), F_2(\cdot), (\mathcal{G}, \texttt{Sign}, \texttt{Vrfy})\big)$, $mk := g_2^\alpha$.
  - Return μ and mk.
- $\texttt{Extract}(\mu, mk, ID)$: $r \leftarrow_R \mathbb{Z}_p$, return $d_{ID} := \big(g_2^\alpha \cdot F_1(ID)^r, g^r\big)$.
- $\texttt{Encrypt}(\mu, ID, m)$: $(vk, sk) := \mathcal{G}(1^{\lambda'})$, $t \leftarrow_R \mathbb{Z}_p$, $\widetilde{C} := \big(m \cdot e(g_1, g_2)^t, g^t, F_1(ID)^t, F_2(vk)^t\big)$, $C := \big(\widetilde{C}, vk, \texttt{Sign}_{sk}(\widetilde{C})\big)$. Return $\widetilde{C}$.
- $\texttt{Decrypt}(\mu, d, C)$: $(d_1, d_2) \leftarrow d$, $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C$. If the verification $\texttt{Vrfy}_{vk}\big((c_1, c_2, c_3, c_4), \sigma\big)$ fails, return $\perp$. Otherwise, $r' \leftarrow_R \mathbb{Z}_p$, $d'_1 := d_1 F_2(vk)^{r'}$, $d'_2 := g^{r'}$, $m := \dfrac{c_1 \cdot e(d_2, c_3) \cdot e(d'_2, c_4)}{e(d'_1, c_2)}$, return m.

3.2 IB-PRE-II

- $\texttt{Setup}(1^\lambda)$: $p, \mathbb{G}, \mathbb{G}_1, g, \alpha, g_1, g_2, e, u'_1, u_{1,1}, u_{1,2}, \ldots, u_{1,n}, u'_2, u_{2,0}, u_{2,1}, \ldots, u_{2,n}, F_1(\cdot), F_2(\cdot)$ are as same as in Wa-CCA-IBE. $0 < l \leq p - 2$, $E_1 : \{0,1\}^{l+1} \rightarrow \mathbb{G}_1$, $E_2 : \{0,1\}^l \rightarrow \mathbb{G}$. $\mu := \big(g, g_1, g_2, F_1(\cdot), F_2(\cdot), (\mathcal{G}, \texttt{Sign}, \texttt{Vrfy})\big)$. $mk := g_2^\alpha$. Return μ and mk.
- $\texttt{Extract}(\mu, mk, ID)$: $r \leftarrow_R \mathbb{Z}_p$, $d_{ID} := \big(g_2^\alpha \cdot F_1(ID)^r, g^r\big)$. Return d_{ID}.
- $\texttt{Encrypt}(\mu, ID, m)$: $(vk, sk) := \mathcal{G}(1^{\lambda'})$, $t \leftarrow_R \mathbb{Z}_p$, $\mathcal{A} := \big(E_1(m||0) \cdot e(g_1, g_2)^t, g^t, F_1(ID)^t, F_2(vk)^t\big)$, $C_{ID} := \big(\mathcal{A}, vk, \texttt{Sign}_{sk}(\mathcal{A})\big)$. Return C_{ID}.

- $\texttt{Encrypt'}(\mu, ID, m)$: The only difference between `Encrypt'` and `Encrypt` is that, in `Encrypt'`, $\mathcal{A} := \big(E_1(m||1) \cdot e(g_1, g_2)^t, g^t, F_1(ID)^t, F_2(vk)^t\big)$.
- $\texttt{RKGen}(\mu, d_{ID_1}, ID_1, ID_2)$: $\big(g_2^\alpha F_1(ID_1)^r, g^r\big) \leftarrow d_{ID_1}$. $k \leftarrow_R \{0,1\}^l$, $d_{ID_1 \to ID_2} := \big(g_2^\alpha \cdot F_1(ID_1)^r \cdot E_2^{-1}(k), g^r, \texttt{Encrypt'}(\mu, ID_2, k)\big)$. Return $d_{ID_1 \to ID_2}$.
- $\texttt{Reencrypt}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$: $\big(\mathcal{A}, vk, \texttt{Sign}_{sk}(\mathcal{A})\big) \leftarrow C_{ID_1}$, where $\mathcal{A}$ consists of four components. Verify C_{ID_1} by $\texttt{Vrfy}_{vk}\big(\mathcal{A}, \texttt{Sign}_{sk}(\mathcal{A})\big)$. If the verification fails, return $\perp$. $\big(g_2^\alpha \cdot F_1(ID_1)^r \cdot E_2^{-1}(k), g^r, \texttt{Encrypt'}(\mu, ID_2, k)\big) \leftarrow d_{ID_1 \to ID_2}$. $r' \leftarrow_R \mathbb{Z}_p$, $C_{ID_2} := \big(C_{ID_1}, \texttt{Encrypt'}(\mu, ID_2, k), g_2^\alpha \cdot F_1(ID_1)^r \cdot E_2^{-1}(k) \cdot F_2(vk)^{r'}, g^r, g^{r'}\big)$. Return C_{ID_2}.
- $\texttt{Decrypt}(\mu, d_{ID}, C_{ID})$:
  - If C_{ID} is not a re-encrypted ciphertext: $(d_1, d_2) \leftarrow d_{ID}$, $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_{ID}$. If $\texttt{Vrfy}_{vk}\big((c_1, c_2, c_3, c_4), \sigma\big)$ fails, return $\perp$. Otherwise, $r' \leftarrow_R \mathbb{Z}_p$, $d_1' := d_1 \cdot F_2(vk)^{r'}$, $d_2' := g^{r'}$. $M := \dfrac{c_1 \cdot e(d_2, c_3) \cdot e(d_2', c_4)}{e(d_1', c_2)}$. $m||b := E_1^{-1}(M)$. If $b = 0$, return m; otherwise return $\perp$.
  - If C_{ID} is a re-encrypted ciphertext: $(C_1, R, d_1', d_2, d_2') \leftarrow C_{ID}$. $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_1$. If $\texttt{Vrfy}_{vk}\big((c_1, c_2, c_3, c_4), \sigma\big)$ fails, return $\perp$. Otherwise $k := \texttt{Decrypt'}(\mu, d_{ID}, R)$. If $e(d_1' k, g) \neq e(g_1, g_2) \cdot e\big(F_1(ID_1), d_2\big) \cdot e\big(F_2(vk), d_2'\big)$, return $\perp$. $K := E_2(k)$. $M := \dfrac{c_1 \cdot e(d_2, c_3) \cdot e(d_2', c_4)}{e(d_1' K, c_2)}$. $m||b := E_1^{-1}(M)$. If $b = 0$, return m; otherwise return $\perp$.
- $\texttt{Decrypt'}(\mu, d_{ID}, C_{ID})$:

  The only difference between `Decrypt'` and `Decrypt` is that after the calculation of $m||b := E_1^{-1}(M)$, `Decrypt'` return m if $b = 1$ and return $\perp$ if $b = 0$.

3.3 The flaws in IB-PRE-II

Chu and Tzeng proposed the following proposition:

Suppose there is an adversary $\mathcal{A}$ that has advantage ϵ against the game $Exp^{\mathcal{A}, \texttt{IND-PrID-CCA}}$, then there is an algorithm $\mathcal{B}$ that breaks Wa-CCA-IBE with advantage at least $Adv_{\mathcal{B}}^{\texttt{IND-ID-CCA}} \geq \epsilon/e(1 + q_E)$, where q_E is the maximal number of $\mathcal{A}$'s queries to EXTRACT.

However, the above proposition is wrong, since there exists an adversary $\mathcal{A}$ that has advantage 1 against the game $Exp^{\mathcal{A}, \texttt{IND-PrID-CCA}}$. For this adversary and for any $\varepsilon > 0$, the algorithm $\mathcal{B}$ proposed by Chu and Tzeng cannot break Wa-CCA-IBE with advantage at least ε.

This adversary $\mathcal{A}$ plays the game $Exp^{\mathcal{A}, \texttt{IND-PrID-CCA}}$ with the challenger $\mathcal{C}$ as follows.

- In the **Setup** phase, $\mathcal{A}$ receives the public parameter μ. $\mathcal{A}$ does nothing.
- In the **Query phase 1**, $\mathcal{A}$ does nothing.
- In the **Challenge** phase, $\mathcal{A}$ randomly chooses an n-bit identity ID_1, an n-bit identity ID_2 ($ID_2 \neq ID_1$), $m_0, m_1 \in \{0,1\}^l$. She gives (ID_1, m_0, m_1) to $\mathcal{C}$, and she receives $C_{ID_1} := \texttt{Encrypt}(\mu, ID_1, m_b)$, where b is randomly chosen from $\{0,1\}$ and $\mathcal{A}$ does not know the value of b.
- In the **Query phase 2**, $\mathcal{A}$ makes a query $C_{ID_2} = \text{REENCRYPT}(ID_1, ID_2, C_{ID_1})$. She decomposes C_{ID_2} into $(C_{ID_1}, R, d_1', d_2, d_2')$, and decomposes C_{ID_1} into $(c_1, c_2, c_3, c_4, vk, \sigma)$. Then she randomly chooses r'' from $\mathbb{Z}_p$. She computes $d_1'' = d_1' F_2(vk)^{r''}$, $d_2'' = d_2' g^{r''}$, and composes $(C_{ID_1}, R, d_1'', d_2, d_2'')$ into C'_{ID_2}. She makes another query $\text{DECRYPT}(ID_2, C'_{ID_2})$ and gets m'.

- In the **Guess** phase, $\mathcal{A}$ compares m' with m_0. If $m' = m_0$, she outputs 0; otherwise she outputs 1.

Now we show that $\mathcal{A}$ can win this game with the probability 1.

Firstly, we show that all queries made by $\mathcal{A}$ are legal.

- Though $\mathcal{A}$ is forbidden to query both $\textsc{Reencrypt}(ID_1, ID^*, C_{ID_1})$ and $\textsc{Extract}(ID^*)$ for any ID^*, $\mathcal{A}$ did not query $\textsc{Extract}(ID_2)$ before. Hence, the query $\textsc{Reencrypt}(ID_1, ID_2, C_{ID_1})$ is legal.
- The query $\textsc{Decrypt}(ID_2, C'_{ID_2})$ is legal, since C'_{ID_2} is not the answer of any query $\textsc{Reencrypt}(ID_1, ID_2, C_{ID_1})$ the adversary made before.

Secondly, the m' received by $\mathcal{A}$ will certainly be m_0 or m_1.

- The result of the first query $\textsc{Reencrypt}(ID_1, ID_2, C_{ID_1})$ is $\big(C_{ID_1}, \texttt{Encrypt'}(\mu, ID_2, k), \hat{d}_1 F_2(vk)^{r'}, g^r, g^{r'}\big)$, where $\texttt{Encrypt'}(\mu, ID_2, k)$, $\hat{d}_1$, g^r come from $d_{ID_1 \to ID_2}$.
- The second query is $\textsc{Decrypt}\big(ID_2, (C_{ID_1}, \texttt{Encrypt'}(\mu, ID_2, k), \hat{d}_1 F_2(vk)^{r'+r''}, g^r, g^{r'+r''})\big)$. The `Decrypt` algorithm first verifies C_{ID_1} by decomposing C_{ID_1} into $(c_1, c_2, c_3, c_4, vk, \sigma)$ and calling $\texttt{Vrfy}_{vk}\big((c_1, c_2, c_3, c_4), \sigma\big)$. Since $\mathcal{A}$ did not modify C_{ID_1}, the verification certainly passed. Then the `Decrypt` algorithm computes $k := \texttt{Decrypt'}\big(\mu, d_{ID_2}, \texttt{Encrypt'}(\mu, ID_2, k)\big)$, and verifies $e\big(\hat{d}_1 \cdot F_2(vk)^{r'+r''} \cdot k, g\big) \stackrel{?}{=} e(g_1, g_2) \cdot e\big(F_1(ID_1), d_2\big) \cdot e\big(F_2(vk), g^{r'+r''}\big)$. This verification also succeed.
- The challenger $\mathcal{C}$ computes $m'||b := E_1^{-1}(M)$ and returns m' to $\mathcal{A}$.

Hence, $m' \in \{m_0, m_1\}$, and $\mathcal{A}$ wins the game after simply comparing m' with m_0 and m_1 and answers 0 or 1.

The Simulation Certainly Fails: For brevity, we put the simulation proposed by Chu and Tzeng in the `Appendix`.

Suppose $\mathcal{B}$ uses the above adversary $\mathcal{A}$ to break Wa-CCA-IBE, then the simulation certainly fails, so $\mathcal{B}$ cannot break Wa-CCA-IBE. Let us see why the simulation fails:

- After the **Setup** phase and the **Query phase 1**, the table of $\mathcal{B}$ is empty.
- In the **Challenge** phase, $\mathcal{B}$ receives (ID_1, m_0, m_1). He checks the table. Since the table is empty, he does not find any $(1, ID_1, *, *)$. So he sends $(ID_1, m_0||0, m_1||0)$ to $\mathcal{C}$ and returns the result C_{ID_1} to $\mathcal{A}$. Clearly $C_{ID_1} = \texttt{Encrypt}_{Wa-CCA-IBE}(\mu, ID_1, m_b)$, $b \in \{0, 1\}$.
- In the **Query phase 2**, $\mathcal{B}$ is queried by $\textsc{Reencrypt}(ID_1, ID_2, C_{ID_1})$.

  $\mathcal{B}$ decomposes C_{ID_1} into $(c_1, c_2, c_3, c_4, vk, \sigma)$. Then he searches the table. Since there is no $(*, ID_1, ID_2, *)$ in the table, $\mathcal{B}$ queries $\textsc{RKExtract}(ID_1, ID_2)$ to himself.

  To respond the query $\textsc{RKExtract}(ID_1, ID_2)$, $\mathcal{B}$ chooses $d_{ID_1 \to ID_2} :=$

  1. $\texttt{Extract}_{IB-PRE-II}\big(\mu, \textsc{Extract}_{Wa-CCA-IBE}(\mu, mk, ID_1), ID_1, ID_2\big)$ with the probability δ.
  2. $\big(x, y, \texttt{Encrypt'}_{IB-PRE-II}(\mu, ID_2, z)\big)$ (x,y,z are random), with the probability $1 - \delta$.

If $\mathcal{B}$ does choice 1. The above choice contains the query $\text{EXTRACT}_{Wa-CCA-IBE}(\mu, mk, ID_1)$. Since $\mathcal{B}$ has submitted $(ID_1, m_0||0, m_1||0)$ to $\mathcal{C}$, hence, by the rules, $\mathcal{B}$ is forbidden to do the query $\text{EXTRACT}_{Wa-CCA-IBE}(\mu, mk, ID_1)$. As a result, the simulation fails.

If $\mathcal{B}$ does choice 2. Then he will records $(1, ID_1, ID_2, d_{ID_1 \to ID_2})$ in the table. Then $\mathcal{B}$ uses $d_{ID_1 \to ID_2}$ to calculate C_{ID_2} and returns $C_{ID_2} := \texttt{Reencrypt}_{IB-PRE-II}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$ to $\mathcal{A}$. Clearly, $C_{ID_2} := \big(C_{ID_1}, \texttt{Encrypt'}_{IB-PRE-II}(\mu, ID_2, z), xF_2(vk)^{r'}, y, g^{r'}\big)$ where $r' \leftarrow_R \mathbb{Z}_p$.

- If in the above REENCRYPT query, the simulation does not fail, $\mathcal{B}$ is queried by $\text{DECRYPT}(ID_2, C'_{ID_2})$. In this case, $\mathcal{B}$ decomposes C'_{ID_2} into $\big(C_{ID_1}, R := \texttt{Encrypt'}_{IB-PRE-II}(\mu, ID_2, z), d'_1 := xF_2(vk)^{r'+r''}, d_2 := y, d'_2 := g^{r'+r''}\big)$, where x, y, z, r' are randomly chosen by $\mathcal{B}$ and r'' is chosen by $\mathcal{A}$. Then he searches the table and will certainly find $(0, ID_1, ID_2, d_{ID_1 \to ID_2})$. At the moment, $\mathcal{B}$ decomposes $d_{ID_1 \to ID_2}$ into $\big(\widetilde{d}_1 := x, \widetilde{d}_2 := y, \widetilde{d}_3 := \texttt{Encrypt'}_{IB-PRE-II}(\mu, ID_2, z)\big)$. Furthermore, the verfications $d_2 = \widetilde{d}_2$ and $R = \widetilde{d}_3$ certainly succeed. The equation $e(d'_1, g) = e(\widetilde{d}_1, g) \cdot e\big(F_2(vk), d'_2\big)$ also holds because the left side equals $e\big(xF_2(vk)^{r''}, g\big)$ and the right side equals $e(x, g) \cdot e\big(F_2(vk), g^{r''}\big)$. However, $\mathcal{B}$ is forbidden to do this query. Hence, the simulation fails.

As a result, Chu and Tzeng's security proof of IB-PRE-II fails, and IB-PRE-II is not IND-ID-CCA secure and is not the solution to Green and Ateniese's open problems neither.

4 Our Improvement

In this section, we propose a CCA-secure improvement by modifying `RKGen`, `Reencrypt` and `Decrypt` algorithms of Chu and Tzeng's scheme.

Our ideas are almost as the same as that in [6], which are as follows.

- For the CCA-security, we protect the re-encrypted ciphertext by one more signature.
- For the multi-use property, since one people can decrypt a multi-times re-encrypted ciphertext as long as she knows k (k is used in $d_{ID_1 \to ID_2}$), the re-encryption is implemented by re-encrypting the encryption of k. A ciphertext can be re-encrypted, and the result ciphertext can be re-encrypted again. If a non-re-encrypted ciphertext C_0 is re-encrypted n times, the format of the final ciphertext is $(C_0, E_1, E_2, \ldots, E_n)$, where each E_i $(1 \le i \le n)$ is eight components. Let $\mathcal{R}$ be the `Reencrypt` algorithm, C_i be the i-times re-encrypted ciphertext. We get that $C_1 := \mathcal{R}(C_0) = (C_0, E_1)$, and $C_{n+1} := \mathcal{R}(C_n) = \big(C_0, E_1, E_2, \ldots, E_n, \mathcal{R}(E_n)\big)$.
- For the integrity test, we add ID in the re-encrypted ciphertext.

The modifications are as follows.

- `Modified Setup`(1^λ):
  $0 \le l \le \lfloor log_2 p \rfloor - 1$, $E_1 : \{0,1\}^{l+1} \to \mathbb{G}_1$, $E_2 : \{0,1\}^l \to \mathbb{G}$.
- `Modified RKGen`$(\mu, d_{ID_1}, ID_1, ID_2)$:
  1. Parse d_{ID_1}, $(g_2^\alpha \cdot F_1(ID_1)^r, g^r) \leftarrow d_{ID_1}$,
  2. Choose a random number k, $k \leftarrow_R \{0,1\}^l$,

3. Compute the re-encryption key $d_{ID_1 \to ID_2}$,

$$d_{ID_1 \to ID_2} := (g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k),\ g^r,\ \texttt{Encrypt}'(\mu, ID_2, k),\ ID_1).$$

4. Return $d_{ID_1 \to ID_2}$.

- $\texttt{Modified Reencrypt}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$:

  Case C_{ID_1} is not a re-encrypted ciphertext:

  1. Parse C_{ID_1}, $(\mathfrak{A}, vk, \texttt{Sign}_{sk}(\mathfrak{A})) \leftarrow C_{ID_1}$.
  2. Verify C_{ID_1} by $\texttt{Vrfy}_{vk}(\mathfrak{A}, \texttt{Sign}_{sk}(\mathfrak{A}))$. If the verification fails, return $\perp$.
  3. Get a new one-time signature key pair, $(vk', sk') \to \mathcal{G}(1^{\lambda''})$.
  4. Choose a random number r', $r' \leftarrow_R \mathbb{Z}_p$.
  5. Parse $d_{ID_1 \to ID_2}$, $(g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k), g^r, \texttt{Encrypt}'(\mu, ID_2, k), ID_1) \leftarrow d_{ID_1 \to ID_2}$.
  6. Compute $\widetilde{C}_{ID_2}$,

  $$\widetilde{C}_{ID_2} = \Big(C_{ID_1}, \texttt{Encrypt}'(\mu, ID_2, k), ID_1, g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k) \cdot F_2(vk)^{r'}, g^r, g^{r'}, F_2(vk')^{r'}\Big),$$

  where $g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k)$, $\texttt{Encrypt}'(\mu, ID_2, k)$, ID_1 all come from $d_{ID_1 \to ID_2}$.
  7. Return $\Big(C_{ID_1}, \texttt{Encrypt}'(\mu, ID_2, k), ID_1, g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k) \cdot F_2(vk)^{r'}, g^r, g^{r'}, F_2(vk')^{r'}, vk', \texttt{Sign}_{sk'}(\widetilde{C}_{ID_2})\Big)$.

  Case C_{ID_1} is a "re-encrypted" ciphertext:

  1. Parse C_{ID_1}, $\Big(C, E_1, I_1, d'_{1,1}, d_{1,2}, d'_{1,2}, d_{1,3}, vk_1, \sigma_1, E_2, I_2, d'_{2,1}, d_{2,2}, d'_{2,2}, d_{2,3}, vk_2, \sigma_2, \ldots, E_n, I_n, d'_{n,1}, d_{n,2}, d'_{n,2}, d_{n,3}, vk_n, \sigma_n\Big) \leftarrow C_{ID_1}$, where C is the non-re-encrypted ciphertext of the plaintext, and each of $E_1, E_2, \ldots, E_n$ is the third component of some re-key, and each of $I_1, I_2, \ldots, I_n$ is an identity.
  2. Get a new one-time signature key pair, $(vk_{n+1}, sk_{n+1}) := \mathcal{G}(1^{\lambda''})$.
  3. Choose a random number r', $r' \leftarrow_R \mathbb{Z}_p$.
  4. Parse $d_{ID_1 \to ID_2}$,

  $$\Big(g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k), g^r, \texttt{Encrypt}'(\mu, ID_2, k), ID_1\Big) \leftarrow d_{ID_1 \to ID_2}.$$

  5. Compute

$$\begin{aligned}
E_{n+1} &= \texttt{Encrypt}'(\mu, ID_2, k),\\
I_{n+1} &= ID_1,\\
d'_{n+1,1} &= g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k) \cdot F_2(vk_n)^{r'},\\
d_{n+1,2} &= g^r,\\
d'_{n+1,2} &= g^{r'},\\
d_{n+1,3} &= F_2(vk_{n+1})^{r'},\\
\sigma_{n+1} = \texttt{Sign}_{vk_{n+1}} &\Big(C, E_1, I_1, d'_{1,1}, d_{1,2}, d'_{1,2}, d_{1,3}, vk_1, \sigma_1, \ldots,\\
&\quad E_{n+1}, I_{n+1}, d'_{n+1,1}, d_{n+1,2}, d'_{n+1,2}, d_{n+1,3}\Big).
\end{aligned}$$

6. Return $\Big(C, E_1, I_1, d'_{1,1}, d_{1,2}, d'_{1,2}, d_{1,3}, vk_1, \sigma_1, \ldots, E_{n+1}, I_{n+1}, d'_{n+1,1}, d_{n+1,2}, d'_{n+1,2}, d_{n+1,3}, vk_{n+1}, \sigma_{n+1}\Big)$.

- `Modified Decrypt`(μ, d_{ID}, C_{ID}):

  Case C_{ID} is not a re-encrypted ciphertext:

  1. Parse d_{ID}, $(d_1, d_2) \leftarrow d_{ID}$.
  2. Parse C_{ID}, $(\mathfrak{A}, vk, \sigma) \leftarrow C_{ID}$.
  3. Parse $\mathfrak{A}$, $(c_1, c_2, c_3, c_4) \leftarrow \mathfrak{A}$.
  4. If $\texttt{Vrfy}_{vk}\Big((c_1, c_2, c_3, c_4), \sigma\Big)$ fails or $e\Big(c_3, F_2(vk)\Big) \neq e\Big(c_4, F_1(ID)\Big)$, return $\perp$.
  5. Compute
  $$d'_1 = d_1 F_2(vk),$$
  $$d'_2 = g,$$
  $$M = \frac{c_1 \cdot e(d_2, c_3) \cdot e(d'_2, c_4)}{e(d'_1, c_2)},$$
  $$m||b = E_1^{-1}(M),\ b \in \{0, 1\}.$$
  6. If $b = 0$, return m; otherwise return $\perp$.

  Case C_{ID} is a re-encrypted ciphertext:

  1. Parse C_{ID}, $\Big(C, E_1, I_1, d'_{1,1}, d_{1,2}, d'_{1,2}, d_{1,3}, vk_1, \sigma_1, E_2, I_2, d'_{2,1}, d_{2,2}, d'_{2,2}, d_{2,3}, vk_2, \sigma_2, \ldots, E_n, I_n, d'_{n,1}, d_{n,2}, d'_{n,2}, d_{n,3}, vk_n, \sigma_n\Big) \leftarrow C_{ID}$.
  2. Compute $k_n = \texttt{Decrypt}'(\mu, d_{ID}, E_n)$.
  3. For any i, $1 \leq i < n$,
     - Parse E_i, $(c_{i,1}, c_{i,2}, c_{i,3}, c_{i,4}) \leftarrow E_i$.
     - Compute
     $$K_i := \frac{c_{i,1} \cdot e(d_{i+1,2}, c_{i,3}) \cdot e(d'_{i+1,2}, c_{i,4})}{e\Big(d'_{i+1,1} \cdot E_2(k_{i+1}), c_{i,2}\Big)},$$
     $$k_i||b_i := E_1^{-1}(K_i), b_i \in \{0, 1\}.$$
  4. For any i, $1 \leq i < n$, if $b_i = 0$, return $\perp$.
     (*k_i can be computed if k_{i+1} is known. Since k_n is known, all of $k_1, k_2, \ldots, k_{n-1}$ can be computed, hence the plain text can be computed.*)
  5. Parse C, $\Big((c_1, c_2, c_3, c_4), vk_0, \sigma_0\Big) \leftarrow C$.
  6. If $\texttt{Vrfy}_{vk_0}\Big((c_1, c_2, c_3, c_4), \sigma_0\Big)$ fails, return $\perp$.
  7. For any i, $1 \leq i \leq n$, if $\texttt{Vrfy}_{vk_i}\Big(\text{All the components between } C \text{ and } d_{i,3}\Big)$ fails, or $e(d_{i,3}, g) \neq e\Big(F_2(vk_i), d'_{i,2}\Big)$, or $e\Big(d'_{i,1} \cdot E_2(k_i), g\Big) \neq e(g_1, g_2) \cdot e\Big(F_1(ID_i), d_{i,2}\Big) \cdot e\Big(F_2(vk_{i-1}), d'_{i,2}\Big)$, return $\perp$.

8. Compute

$$M := \frac{c_1 \cdot e(d_{1,2}, c_3) \cdot e(d'_{1,2}, c_4)}{e\Big(d'_{1,1} \cdot E_2(k_1), c_2\Big)},$$

$$m||b := E_1^{-1}(M),\ b \in \{0,1\}.$$

9. If $b = 0$, return m; otherwise, return $\perp$.

4.1 Security of the Improved IB-PRE-II

Now we prove that the improved IB-PRE-II is IND-ID-CCA secure, such that if there is an adversary $\mathcal{A}$ which has advantage ε against the game $Exp^{\mathcal{A},\texttt{IND-PrID-CCA}}$, then there is a simulator $\mathcal{B}$ which can break Wa-CCA-IBE with advantage at least $\varepsilon \cdot 1/\Big(e\big(1 + (q_R + 1)q_E\big)\Big)$, where q_E is the maximal number of $\mathcal{A}$'s Extract queries, q_R is the maximal number of $\mathcal{A}$'s RKExtract or Reencrypt queries.

We prove this proposition in the following.

$\mathcal{B}$ plays the game $Exp^{\mathcal{A},\texttt{IND-PrID-CCA}}$ with $\mathcal{A}$.

$\mathcal{B}$ maintains a table with tuples (β, ID_1, ID_2, d), where $\beta \in \{0,1\}$, ID_1 and ID_2 are two identities, d is the re-encryption key.

Setup. $\mathcal{B}$ gets the public parameter μ from IND-ID-CCA's challenger $\mathcal{C}$,[1] and returns μ to $\mathcal{A}$.

Query phase 1. $\mathcal{B}$ is queried by $\mathcal{A}$, and $\mathcal{B}$ responses the queries respectively:

1. Extract(ID): Choose a random coin β, $\beta \leftarrow_R \{0,1\}$, such that $Pr[\beta = 1] = \delta$, δ is a constant.
   - If $\beta = 0$ or $(0, ID, *, *)$ or $(0, *, ID, *)$ exists in the table, then abort.
   - Otherwise, $d_{ID} :=$ Extract$_{Wa-CCA-IBE}(ID)$, record $(1, ID, ID, \perp)$ in the table, and return d_{ID} to $\mathcal{A}$.
2. RKExtract(ID_1, ID_2): $\mathcal{B}$ chooses a random coin β as in the Extract query.
   - If $\beta = 1$, or $(1, ID_1, ID_1, \perp)$ or $(1, ID_2, ID_2, \perp)$ already exists on the table, $\mathcal{B}$ queries the key extraction oracle of Wa-CCA-IBE for ID_1, and then computes the re-encryption key as the original scheme and returns it to $\mathcal{A}$.
   - Otherwise, $\mathcal{B}$ returns a random re-encryption key $d_{ID_1 \to ID_2} = \big(x, y, \texttt{Encrypt}'(\mu, ID_2, z), ID_1\big)$, where $x \leftarrow_R \mathbb{G}$, $y \leftarrow_R \mathbb{G}$, $z \leftarrow_R \{0,1\}^l$.

   Finally, $\mathcal{B}$ records $(\beta, ID_1, ID_2, d_{ID_1 \to ID_2})$ in the table.
3. Reencrypt(ID_1, ID_2, C_{ID_1}): Parse C_{ID_1}, $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_{ID_1}$. If $(*, ID_1, ID_2, *)$ is not in the table, $d_{ID_1 \to ID_2}$ = RKExtract(ID_1, ID_2). At last, return $C_{ID_2} = Reencrypt_{IB-PRE-II}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$.
4. Decrypt(ID, C_{ID}):
   - If C_{ID} is not a re-encrypted ciphertext, $\mathcal{B}$ sends C_{ID} to the decryption oracle of Wa-CCA-IBE for ID, and $\mathcal{B}$ returns the result to $\mathcal{A}$ if the result ends with '0', and $\mathcal{B}$ returns $\perp$ to $\mathcal{A}$ if the result ends with '1'.
   - If C_{ID} is a re-encrypted ciphertext,
     (a) Parse C_{ID}, $(C, E_1, I_1, d'_{1,1}, d_{1,2}, d'_{1,2}, d_{1,3}, vk_1, \sigma_1, E_2, I_2, d'_{2,1}, d_{2,2}, d'_{2,2}, d_{2,3}, vk_2, \sigma_2, \ldots, E_n, I_n, d'_{n,1}, d_{n,2}, d'_{n,2}, d_{n,3}, vk_n, \sigma_n,) \leftarrow C_{ID}$.

[1] The IND-ID-CCA's challenger $\mathcal{C}$ holds the oracles of Wa-CCA-IBE, such as Extract$_{Wa-CCA-IBE}$.

(b) Set $I_{n+1} := ID$.
(c) Parse C, $\big((c_1, c_2, c_3, c_4), vk_0, \sigma_0\big) \leftarrow C$.
(d) If $\mathtt{Vrfy}_{vk_0}\big((c_1, c_2, c_3, c_4), \sigma_0\big)$ fails, return $\perp$.
(e) For any i, $1 \le i \le n$,
- If $\mathtt{Vrfy}_{vk_i}$(all the components from C to $d_{i,3}$, σ_i) fails, or $e\big(F_2(vk_i), d'_{i,2}\big) \neq e(d_{i,3}, g)$, return $\perp$.
- If $(0, I_i, I_{i+1}, *)$ exists in the table,
  (i) let $(0, I_i, I_{i+1}, \tilde{d}_i)$ be that tuple,
  (ii) parse d_i, $(\tilde{d}_{i,1}, \tilde{d}_{i,2}, \tilde{d}_{i,3}, \tilde{d}_{i,4}) \leftarrow \tilde{d}_i$,
  (iii) if $e(d'_{i,1}, g) \neq e(\tilde{d}_{i,1}, g) \cdot e\big(F_2(vk_{i-1}), d'_{i,2}\big)$, or $d_{i,2} \neq \tilde{d}_{i,2}$, or $I_i \neq \tilde{d}_{i,4}$, return $\perp$.
- If $(0, I_i, I_{i+1}, *)$ does not exist in the table,
  (i) $k_i || k'_i = \mathtt{Decrypt}_{Wa-CCA-IBE}(ID, E_i), k'_i \in \{0, 1\}$,
  (ii) if $k'_i = 0$, return $\perp$,
  (iii) If $e(g_1, g_2) \cdot e\big(F_1(I_i), d_{i,2}\big) \cdot e\big(F_2(vk_i), d'_{i,2}\big) \neq e\big(d'_{i,1} \cdot E_2(k_i), g\big)$, return $\perp$.

(f) If $(0, I_1, *, *)$ exists in the table, $m||m' = \mathtt{Decrypt}_{Wa-CCA-IBE}(ID, C_{ID})$, $m' \in \{0, 1\}$.
(g) If $(0, I_1, *, *)$ does not exist in the table, $M = \dfrac{c_1 \cdot e(d_{1,2}, c_3) \cdot e(d'_{1,2}, c_4)}{e\big(d'_{1,1} \cdot E_2(k_1), c_2\big)}$, $m||m' = E_1^{-1}(M), m' \in \{0, 1\}$.
(h) If $m' = 0$, return m, else return $\perp$.

Challenge. $\mathcal{B}$ receives (ID^*, m_0, m_1) from $\mathcal{A}$. If $(1, ID^*, *, *)$ exists in the table, $\mathcal{B}$ returns a random bit to $\mathcal{C}$ and aborts. Otherwise, $\mathcal{B}$ sends $(ID^*, m_0||0, m_1||0)$ to $\mathcal{C}$ and gets C^*. Then $\mathcal{B}$ returns C^* to $\mathcal{A}$.

Query phase 2. $\mathcal{B}$ continues responding the following queries. And the query restriction on $\mathcal{A}$ is described in section 2.6. $\mathcal{B}$ responds the Extract, Reencrypt, Decrypt queries in the same way as in **Query phase 1**.

For the RKExtract(ID_1, ID_2) query.

- If $ID_1 \neq ID^*$, $\mathcal{B}$ queries the key extraction oracle of Wa-CCA-IBE for ID_1 to get d_{ID_1}, and then computes the re-encryption key as the IB-PRE-II and returns it to $\mathcal{A}$.
- If $ID_1 = ID^*$, $\mathcal{B}$ returns a random re-encryption key $d_{ID_1 \to ID_2} = \big(x, y, \mathtt{Encrypt}'(\mu, ID_2, z), \mathtt{Encrypt}(\mu, ID_2, ID_1)\big)$ for $x, y \leftarrow_R \mathbb{G}$, $z \leftarrow_R \{0, 1\}^l$, and records the tuple $(0, ID_1, ID_2, d_{ID_1 \to ID_2})$ in the table.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $d' \in \{0, 1\}$. And $\mathcal{B}$ outputs d' to $\mathcal{C}$.

Assume that $\mathcal{A}$ makes Extract queries for q_E times until the simulation ends (q'_E times in **Query phase 1** and q''_E times in **Query phase 2**), and assume that $\mathcal{A}$ makes RKExtract and Reencrypt queries for a total of q_R times until the simulation ends.

In the **Query phase 1**, $\mathcal{B}$ does not abort the simulation with the probability at least $\delta^{(q_R+1)q'_E}$.

Let P_i denote the probability of $\mathcal{B}$ does not abort when $\mathcal{B}$ responses the i-th Extract query, $1 \le i \le q'_E$. Let n_i denote the total times of RKExtract and Reencrypt queries before the i-th Extract query, $n_i \le q_R$. Then when $\mathcal{B}$ responses the i-th Extract query, there is at most n_i tuples in $\mathcal{B}$'s table.

Let Q_i denote the probability of that all the tuples are in the form of $(1, *, *, *)$. Then $Q_i \ge \delta^{n_i}$. *($Q_i = \delta^{n_i}$ corresponds to the scenario that for each* RKExtract *query, $\mathcal{B}$ gets $\beta = 1$, and for each* Reencrypt(ID_1, ID_2, C_{ID_1}) *query, $\mathcal{B}$ did not find any $(*, ID_1, ID_2, *)$ tuple in his table.)*

We have

$$\begin{aligned} P_i &= Pr\big[\beta = 1 \wedge \text{no } (0, ID, *, *) \text{ tuple in the table} \wedge \text{no } (0, *, ID, *) \text{ tuple in the table}\big] \\ &\geq Pr\big[\beta = 1\big] \cdot Pr\big[\text{all the tuples are in the form of (1, *, *, *)}\big] = \delta \cdot \delta^{n_i} = \delta^{n_i+1} \geq \delta^{q_R+1}. \end{aligned}$$

So the probability that $\mathcal{B}$ does not abort in **Query phase 1** is

$$P_1 = \prod_{1 \leq i \leq q'_E} P_i \geq \prod_{1 \leq i \leq q'_E} \delta^{q_R+1} = \delta^{(q_R+1)q'_E}$$

In the **Challenge** phase, $\mathcal{B}$ does not abort the simulation with the probability at least $(1 - \delta)$.

$\mathcal{B}$ does not abort in the **Challenge** phase if and only if there is no tuple with the form $(1, ID^*, *, *)$ in the table. Since the tuples, whose second component is ID^*, are all with the same first component. i.e., they are all $(1, ID^*, *, *)$ or all $(0, ID^*, *, *)$. We have the probability that there is no tuple with the form $(1, ID^*, *, *)$ is no less than $1 - \delta$.

In the **Query Phase 2**, $\mathcal{B}$ does not abort the simulation with the probability at least $\delta^{(q_R+1)q''_E}$. *The analysis is as same as the analysis of* **Query phase 1**.

So the probability that $\mathcal{B}$ does not abort the simulation is $\delta^{(q_R+1)q_E}(1 - \delta)$. The maximal value of this probability is $\frac{1}{e\big(1+(q_R+1)q_E\big)}$, where $\delta = \frac{(q_R+1)q_E}{1+(q_R+1)q_E}$.

If $\mathcal{B}$ does not abort during the game, the view of $\mathcal{A}$ is identical to an real attack to IB-PRE-II except for some incorrect re-encryption keys. $\mathcal{A}$ cannot distinguish these random generated keys from the real keys. Because each of the random generated key is $\big(x, y, \texttt{Encrypt}'(\mu, ID_2, z), ID_1\big)$. Though x and y are random, there certainly exists $r \in \mathbb{Z}_p$, $k \in \{0,1\}^l$, satisfying $g_2^{\alpha} \cdot F_1(ID_1)^r \cdot E_2^{-1}(k) = x$, $g^r = y$. Moreover, $\texttt{Encrypt}'(\mu, ID_2, z)$ and $\texttt{Encrypt}'(\mu, ID_2, z)$ are indistinguishable. So the simulation is indistinguishable from a real attack.

As a result, we get the expected probability.

5 Conclusion

In this paper, we introduced Chu and Tzeng's two identity-based proxy re-encryption schemes, and pointed out a security flaw in their second scheme. Furthermore, we proposed an improvement, which is proven-CCA-secure in the standard model. Our improvement is a solution to Green and Ateniese's two open problems. However, our result is not perfect, we think the perfect result for these two problems is no ciphertext expansion[2], which we left as future work.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Internet Society (ISOC): NDSS 2005*, pages 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, 2006.

[3] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127–144, 1998.

[2] In our proposal, one time re-encryption, the ciphertext will be added to 8 components.

[4] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 231–229, 2001.

[5] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[6] C. Chu and W. Tzeng. Identity-based proxy re-encryption without random oracles. In *ISC 2007*, volume 4779 of *LNCS*, pages 189–202, 2007.

[7] M. Green and G. Ateniese. Identity-based proxy re-encryption. In *ACNS 2007*, volume 4521 of *LNCS*, pages 288–306, 2007. Full version: Cryptology ePrint Archieve: Report 2006/473.

[8] B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127, 2005.

Appendix. The original simulation in IB-PRE-II

$\mathcal{B}$ plays the game $Exp^{\mathcal{A},\texttt{IND-PrID-CCA}}$ with $\mathcal{A}$.

$\mathcal{B}$ maintains a table with tuples (β, ID_1, ID_2, d), where $\beta \in \{0,1\}$, ID_1 and ID_2 are two identities, d is the re-encryption key.

Setup. $\mathcal{B}$ gets the public parameter μ from IND-ID-CCA challenger $\mathcal{C}$, and returns μ to $\mathcal{A}$.

Query phase 1. $\mathcal{B}$ is queried by $\mathcal{A}$, and $\mathcal{B}$ responses the queries respectively:

1. EXTRACT(ID):
   (a) $\beta \leftarrow_R \{0,1\}, Pr[\beta = 1] = \delta, 0 \le \delta \le 1$, δ is a constant.
   (b) `If` $\beta = 0$ or $(0, ID, *, *)$ or $(0, *, ID, *)$ exists in the table,
   $c \leftarrow_R \{0,1\}$, give c to $\mathcal{C}$ then abort.
   (c) `Otherwise`,
   $d_{ID} :=$ EXTRACT$_{Wa-CCA-IBE}(ID)$, record $(1, ID, ID, \perp)$ in the table and return d_{ID} to $\mathcal{A}$.
2. RKEXTRACT(ID_1, ID_2):
   (a) $\beta \leftarrow_R \{0,1\}, Pr[\beta = 1] = \delta$.
   - If $\beta = 1$ or $(1, ID_1, ID_1, \perp)$ or $(1, ID_2, ID_2, \perp)$ exists in the table,
     $d_{ID_1} :=$ EXTRACT$_{Wa-CCA-IBE}(ID_1)$,
     $d_{ID_1 \to ID_2} :=$ RKEXTRACT$_{IB-PRE-II}(ID_1, ID_2)$.
   - Otherwise,
     $x, y \leftarrow_R \mathbb{G}$, $z \leftarrow_R \{0,1\}$, $d_{ID_1 \to ID_2} := (x, y, \texttt{Encrypt}'_{IB-PRE-II}(\mu, ID_2, z)$.

   (b) Record $(\beta, ID_1, ID_2, d_{ID_1 \to ID_2})$ in the table,
   (c) Return $d_{ID_1 \to ID_2}$ to $\mathcal{A}$.
3. REENCRYPT(ID_1, ID_2, C_{ID_1}):
   $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_{ID_1}$,
   `If` $(*, ID_1, ID_2, *)$ is not in the table, $d_{ID_1 \to ID_2} :=$ RKEXTRACT(ID_1, ID_2). `Endif`.
   Return $C_{ID_2} := Reencrypt_{IB-PRE-II}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$.
4. DECRYPT(ID, C_{ID}):
   - If C_{ID} is not a re-encrypted ciphertext,
     $m :=$ DECRYPT$_{Wa-CCA-IBE}(ID, C_{ID})$.
     If the last bit of m is '0', return m to $\mathcal{A}$ else return $\perp$.

- If C_{ID} is a re-encrypted ciphertext,
  $(C_{ID_1}, R, d'_1, d_2, d'_2) \leftarrow C_{ID}$,
  $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_{ID_1}$,
  - If for any ID_1 and any $\widetilde{d}$, $(0, ID_1, ID, \widetilde{d})$ exists in the table,
    * $(\widetilde{d}_1, \widetilde{d}_2, \widetilde{d}_3) \leftarrow d$,
    * If $d_2 = \widetilde{d}_2$, $R = \widetilde{d}_3$ and $e(d'_1, g) = e(\widetilde{d}_1, g) \cdot e\big(F_2(vk), d'_2\big)$,
      $m := \textsc{Decrypt}_{Wa-CCA-IBE}(ID_1, C_{ID_1})$.
      If the last bit of m is '0', return m to $\mathcal{A}$ else return $\perp$.
  - Else,
    * $k||k' \leftarrow \textsc{Decrypt}_{Wa-CCA-IBE}(ID, R), k' \in \{0,1\}$.
    * If $k' = 0$, return $\perp$.
    * $M := \dfrac{c_1 \cdot e(d_2, c_3) \cdot e(d'_2, c_4)}{e(d'_1 \cdot E_2(k), c_2)}$.
    * $m||m' \leftarrow E_1^{-1}(M), m' \in \{0,1\}$.
    * If $m' = 0$ return m to $\mathcal{A}$, else return $\perp$ to $\mathcal{A}$.

Challenge. $\mathcal{B}$ receives (ID^*, m_0, m_1) from $\mathcal{A}$. If $(1, ID^*, *, *)$ exists in the table, $\mathcal{B}$ returns a random bit to $\mathcal{C}$ and aborts. Otherwise, $\mathcal{B}$ sends $(ID^*, m_0||0, m_1||0)$ to $\mathcal{C}$ and gets C^*. Then $\mathcal{B}$ returns C^* to $\mathcal{A}$.

Query Phase 2. $\mathcal{B}$ continues responding the following queries. And the query restriction on $\mathcal{A}$ is described in section 2.6.

1. $\textsc{Extract}(ID)$:
   $\mathcal{B}$ responds as same as in the **Query phase 1**.
2. $\text{RK}\textsc{Extract}(ID_1, ID_2)$:
   - If $ID_1 \neq ID^*$,
     $\mathcal{B}$ queries $\textsc{Extract}_{Wa-CCA-IBE}(ID_1)$,
     returns $\text{RK}\textsc{Gen}_{IB-PRE-II}(\mu, d_{ID_1}, ID_1, ID_2)$ to $\mathcal{A}$.
   - If $ID_1 = ID^*$,
     $\mathcal{B}$ returns $d_{ID_1 \to ID_2} := (x, y, \texttt{Encrypt}'_{IB-PRE-II}(\mu, ID_2, z))$, where $x \leftarrow_R \mathbb{G}, y \leftarrow_R \mathbb{G}, z \leftarrow_R \mathbb{Z}_p$ and $\mathcal{B}$ records $(0, ID, ID', d_{ID_1 \to ID_2})$ in the table.
3. $\textsc{Reencrypt}(ID_1, ID_2, C_{ID_1})$:
   $(c_1, c_2, c_3, c_4, vk, \sigma) \leftarrow C_{ID_1}$,
   `If` $(*, ID_1, ID_2, *)$ is not in the table, $d_{ID_1 \to ID_2} := \text{RK}\textsc{Extract}(ID_1, ID_2)$.
   Return $C_{ID_2} := \textsc{Reencrypt}_{IB-PRE-II}(\mu, d_{ID_1 \to ID_2}, C_{ID_1})$.
   (Note that the $\text{RK}\textsc{Extract}(ID_1, ID_2)$ *query used here is the one defined in* ***Query Phase 2****, not the one defined in* ***Query Phase 1****).*
4. $\textsc{Decrypt}(ID, C_{ID})$:
   $\mathcal{B}$ responds as same as in **Query phase 1**.

Guess. $\mathcal{B}$ gets b' from $\mathcal{A}$ and gives b' to $\mathcal{C}$.