An Efficient Cloud-based Revocable Identity-based Proxy Re-encryption Scheme for Public Clouds Data Sharing

Kaitai Liang[1], Joseph K. Liu[2], Duncan S. Wong[1], Willy Susilo[3]

Department of Computer Science, City University of Hong Kong[1],
Institute for Infocomm Research, A*STAR, Singapore[2],
School of Computer Science and Software Engineering, University of Wollongong[3]
kliang4-c@my.cityu.edu.hk, ksliu9@gmail.com, duncan@cityu.edu.hk,
wsusilo@uow.edu.au

Abstract. Identity-based encryption (IBE) eliminates the necessity of having a costly certificate verification process. However, revocation remains as a daunting task in terms of ciphertext update and key update phases as due to the lack of a certificate revocation list in this infrastructure. In this paper, we provide an affirmative solution to solve the efficiency problem incurred by revocation. We propose the first cloud-based revocable identity-based proxy re-encryption (CR-IB-PRE) scheme that supports user revocation but also delegation of decryption rights. No matter a user is revoked or not, at the end of a given time period the cloud acting as a proxy will re-encrypt all ciphertexts of the user under the current time period to the next time period. If the user is revoked in the forthcoming time period, he cannot decrypt the ciphertexts by using the expired private key anymore. We state that this primitive is applicable to many practical network applications, such as subscription-based cloud storage services. Comparing to some naive solutions which require a private key generator (PKG) to interact with non-revoked users in each time period, the new scheme provides definite advantages in terms of communication and computation efficiency. Our scheme only requires the PKG to publish a constant-size public string for each time period and meanwhile, the workload of ciphertexts update is off-loaded to the cloud server. More importantly, the scheme can be proven secure in the standard model.

Key words: Revocable identity-based encryption, cloud-based revocable identity-based proxy re-encryption, standard model

1 Introduction

In a traditional public-key infrastructure (PKI), user revocation can be conducted via a certificate mechanism. If a user is revoked, his/her certificate will be added to a certificate revocation list (CRL) by certificate authority. Anyone who wants to encrypt a message for this user has to check the certificate of the user against the CRL. If the certificate is on the list, the sender knows that this user has been revoked and therefore, will not further share any sensitive information with him/her. Different from PKI, there is no certificate in identity-based encryption (IBE) cryptosystem. Therefore user revocation remains an elusive open problem in this paradigm.

To solve this problem, Boneh and Franklin [1] proposed a naive but inefficient solution (the first revocable IBE scheme) such that the computation and communication complexity of private key generator (PKG) are both linearly in the total number N of non-revocable system users, i.e. $O(N)$. Although their work is further studied by different scholars in the following decade, most of the existing revocable IBE systems (e.g. [2]) have not considered how to relieve the cost spent on key and ciphertext updated processes. Therefore, this motivates our work.

1.1 Motivation

Ciphertexts Update. To date cloud-based technology gives birth to the next generation of computing system. With the assistance of cloud server, many costly computations can be performed with ease. For example, in a revocable IBE system, a data sender can encrypt the data under an identity and a time period for a specified receiver such that the receiver can gain access to the data by using his decryption key corresponding to the time period. When the key is expired and the receiver is not on the revocation list, a PKG will issue a new key/token for the next time period to the receiver and the corresponding ciphertext will be updated to the next period as well. Suppose the ciphertext of the data is stored in a public cloud, then for each ciphertext update process the sender has to deal with the download-decrypt-then-re-encrypt process. Although the ciphertext might be stored locally (without loss of confidentiality), the sender should execute decrypt-then-re-encrypt mode. This may consume a great amount of computational resources while there is a great amount of data to be dealt with. Therefore, the sender might not afford the consumption upon using some resource-limited devices.

To off-load the computational workload to the cloud, we might allow the cloud server to handle ciphertext update process. A naive solution is to enable the cloud to gain access to the data. This, nevertheless, violates the confidentiality of data. A better solution is to enable the cloud to re-encrypt an original ciphertext under an old time period to another ciphertext under a new time period without leaking knowledge of either the decryption key or the underlying plaintext. In CRYPTO 2012 Sahai, Seyalioglu and Waters [3] proposed a non-re-encryption methodology to enable a server, given some public information, to fulfill ciphertext update process in the attribute-based encryption setting. In this paper we leverage the technology of proxy re-encryption (PRE) into ciphertext update process to tackle the same problem in the context of revocable IBE. Later, we will show that our system enjoys better efficiency compared to [3]. Using PRE, a ciphertext stored in the cloud can be re-encrypted to another ciphertext by the cloud server acting as a *semi-trusted* proxy. No information related to the data, however, leaks to the proxy. Accordingly, the update process can be executed effectively and efficiently on the side of cloud server such that the workload of data sender is lessen.

Key Update. Using the technology of identity-based PRE (IB-PRE), ciphertext update for user revocation can be *somehow* offloaded to the cloud as well. If a user is revoked, all ciphertexts stored in the cloud server will be re-encrypted to another "identity". For instance, ciphertexts for a user with identity `Alice` are re-encrypted to ciphertexts for another "identity" `Alice-1` such that the decryption key associated with `Alice` is not applicable to the decryption of the newly ciphertexts. Nonetheless, there is an undesirable trade-off by simply leveraging IB-PRE. The user needs to update a new identity upon entering to the new time period (corresponding to `Alice-1`). This does not lead to any problem if the revocation of the user is due to the departure from the organization. However, if the revocation is due to key compromise, that might be a potential risk as the user is able to retain original identity even the corresponding decryption key is comprised. A change in identity (e.g. from `Alice` to `Alice-1`) might bring inconvenience to the user who needs to tell all data senders to use the new identity for the further encryption. That already violates the original idea of using identity-based encryption in which the sender only needs to know some simple information, e.g., name, email address of the user, but not other frequently changeable (or periodical updated) information.

In the ideal case, it is desirable to have a cloud-based encryption scheme with the following features:

1. **Efficient Revocation**: It should support both user and decryption key revocation. User revocation guarantees that if a user has left the organization, he/she will be withdrawn from the right of accessing the information (with respect to his/her identity) in clouds. Decryption key revocation ensures that when the decryption key of a user is stolen or compromised by an adversary, the user may have an oppor-

tunity to update the key so as to decrypt updated ciphertexts. With these properties, only a legitimate user is allowed to continually access the data under the encryption (e.g. being issued a new private key by PKG) but not the adversary with a compromised key. More importantly, the complexity of revocation *should not* be linearly in the number of non-revocable system users (i.e. $O(N)$).

2. **Efficient Ciphertext Update**: Ciphertext update process can be off-loaded to cloud server such that a data sender enjoys less computational cost while there is a great deal of ciphertexts to be updated.
3. **Consistency of Identity after Revocation**: If the decryption key of a user is compromised (that is the case of decryption key revocation), the user should retain his/her original identity (i.e. keeping identity consistent). No additional information will be added to the identity or identification string.

1.2 A Naive Solution

Using any existing IB-PRE system (e.g. [4]), a naive solution can be achieved with the above features. We denote by "`Name | Time Period`" the "identity" of a system user. That is, the time period is concatenated to the original identity of the user. For example, the identity of Alice at January 2014 is represented as `Alice | JAN 2014`. Any data sender can use this string as public key to encrypt the data for Alice in January 2014. In the upcoming month, the identity will be changed to `Alice | FEB 2014`. Before the beginning of March, the server will re-encrypt all ciphertexts of Alice stored in the cloud to `Alice | Mar 2014` such that Alice cannot access the data unless she is granted a new key for March 2014. On the other side, if the key (for February 2014) is stolen by adversary, the same action can be taken. However, in this case the user is required to be given the decryption key for the next time period (i.e. March 2014) in advance.

However, this solution leads to an undesirable trade-off where it brings unnecessary workload for PKG. The solution requires the PKG to issue a decryption key to every user at the beginning of each time period. Most of key generation/update algorithms of revocable IBE systems fulfill the issue of updated decryption key (resp. corresponding updated information) by establishing a secure channel from the PKG to a user. The cost brought by building up the secure channel for each user is acceptable for a new user joining the system at the first time. But if the PKG and the user need to repeat this at every time period, it might not be practical. It not only brings inconvenience to (non-revocable) system users, but also incurs undesirable workload for the PKG as the complexity grows linearly with the number of (non-revocable) users at each time period. Thus this naive solution is not scalable and not practical at all.

1.3 Our Contributions

In this paper we present the following contributions.

- We define the notion of cloud-based revocable identity-based proxy re-encryption (CR-IB-PRE) and its corresponding security model.
- We propose an efficient and concrete system achieving the notion we propose above. It is worth mentioning that the present system is the first to support user revocation but also delegation of decryption rights in the identity-based cryptographic setting.
- Our scheme achieving the features mentioned in the previous section only requires the PKG to publish a **constant-size** public string at the beginning of each time period. The PKG does not need to interact with each user by establishing an individual secure channel such that the complexity of the PKG is reduced to $O(1)$. This public string only allows non-revoked users (but not the revoked users) to fulfill the key update phase. Without this key updating process, the revoked users can not decrypt the ciphertexts stored in the cloud any more as the original ciphertexts are already re-encrypted to the next time period when the users are revoked.
- We prove our new scheme to be secure against chosen-plaintext attack (CPA) and collusion resistant in the standard model.

1.4 System Architecture

Fig. 1 and Fig. 2 show the architecture of a CR-IB-PRE system. Fig. 1 mainly depicts re-encryption and decryption phases of the system. Like a revocable IBE system, a PKG first issues a private key sk_{Alice} associated with an identity, say Alice, to the user Alice. When a time period, say $T5$, has come, the PKG delivers a token τ_{T5} to Alice such that Alice can update her private key to a new decryption key $sk_{Alice|T5}$ to decrypt any ciphertext encrypted under Alice and time period $T5$. When a new time period is approaching, Alice may construct a re-encryption key $rk_{Alice|T5 \rightarrow T6}$ under her identity from $T5$ to $T6$, and then send the key to a cloud server whom will update a ciphertext under Alice and $T5$ to a new ciphertext under Alice and $T6$. However, Alice here cannot immediately decrypt the new ciphertext as a token τ_{T6} is not issued by PKG yet. After the token is issued, Alice can update her decryption key to $sk_{Alice|T6}$

accordingly so as to recover the underlying plaintext. Fig. 2 illustrates the key update phase. The PKG only publishes a public token associated with $T5$ such that any user excluded in the revocation list can leverage this token to update his/her decryption key. This makes key update (for N non-revocable users) reduce to constant cost.

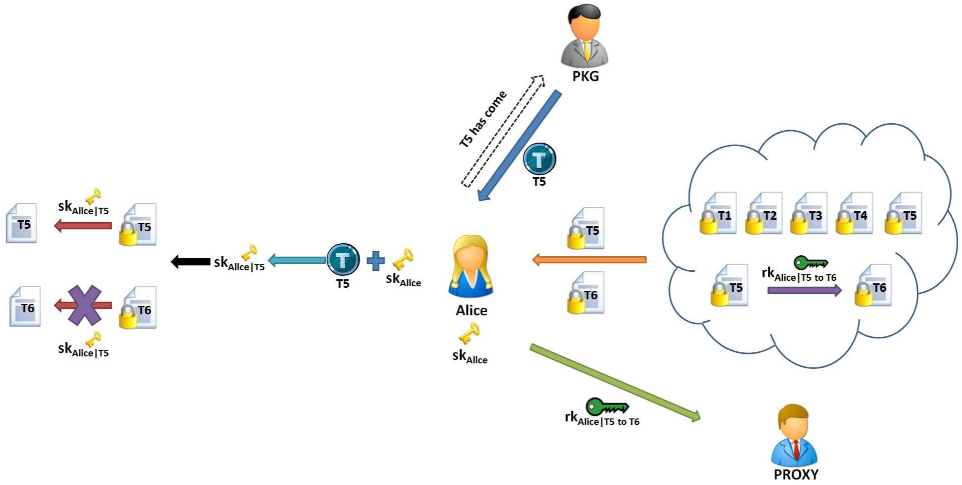

Fig. 1. Re-Encryption and Decryption Phases in a CR-IB-PRE System

One might doubt that the system cannot be seen as a type of IB-PRE because an IB-PRE scheme usually re-encrypts a ciphertext under an identity to another ciphertext under a new identity. Actually, our system does not contradict the notion of IB-PRE by regarding $(Alice, T5)$ and $(Alice, T6)$ as two different identities. One might further question that ciphertext update process may be suspended by a dishonest user if the user refuses to deliver the corresponding re-encryption key to the server. To address this problem, we propose a solution right after our basic construction in Section 5.2.

1.5 Related Work

The first revocable IBE is proposed by Boneh and Franklin [1], in which a ciphertext is encrypted under an identity id and a time period T, and a non-revoked user is issued a private key $sk_{id,T}$ by a PKG such that the user can access the data in T. However, this does not scale well as

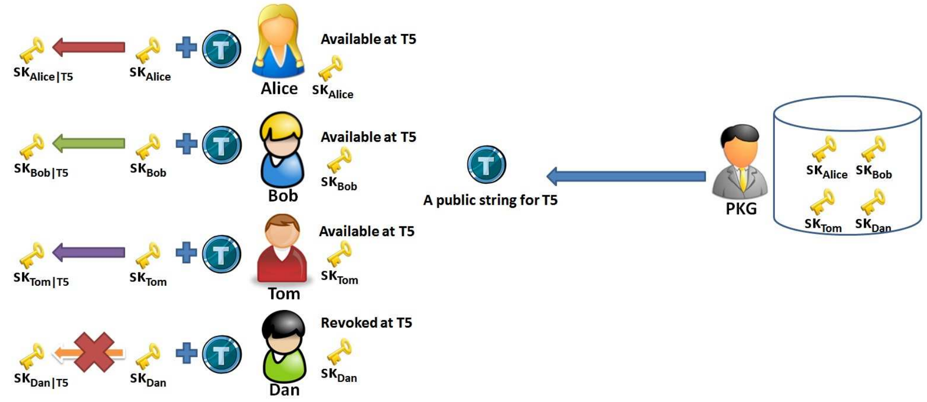

Fig. 2. Key Update Phase in a CR-IB-PRE System

the complexity of the PKG is linearly in the number N of non-revocable users. Subsequently, Boldyreva, Goyal and Kumar [2] proposed the security notion for revocable IBE, and constructed an efficient revocable IBE scheme from a fuzzy IBE scheme [5] with binary tree structure. To achieve adaptive security, Libert and Vergnaud [6] proposed a revocable IBE scheme based on the variant of Waters IBE [7] and Gentry IBE [8]. Recently, Seo and Emura [9] formalized a revised notion for revocable IBE, and proposed a concrete scheme based on [6]. Since its introduction, there are many variants of revocable IBE. For example, several revocable IBE schemes [10,11,12] leverage a semi-trusted authority to enable users to fulfill valid decryption. There are also some functional encryption schemes [13,14,15,3] considering the property of revocation. Inspired by [9] we will build the first CR-IB-PRE scheme in the standard model.

Decryption rights delegation is introduced in [16]. Blaze, Bleumer and Strauss [17] formally defined the notion of PRE. PRE can be classified as: unidirectional and bidirectional PRE, and single-hop and multi-hop PRE, where the definitions are given in [18]. This present work deals with the multi-hop unidirectional case. Many PRE systems have been proposed in the literature, such as [18,19,20,21,22].

To employ PRE in the IBE setting, Green and Ateniese [4] defined the notion of identity-based PRE (IB-PRE), and proposed two constructions in the random oracle model. Later on, Tang, Hartel and Jonker [23] proposed a CPA-secure IB-PRE scheme in the random oracle model, in which delegator and delegatee can belong to different domains. Chu and Tzeng [24] proposed an IB-PRE scheme without random oracles against replayable chosen-ciphertext attacks (RCCA) [25]. The aforementioned schemes, however, enable proxy to compromise the entire private key of delegator by colluding with delegatee. To tackle the problem, the following systems are proposed. Two CPA-secure IB-PRE schemes without random oracles were proposed by Matsuo [26]. Wang et al. [27,28] proposed two IB-PRE schemes in the random oracle model. Minzuno and Doi [29] constructed an IB-PRE scheme in the standard model with CPA security. Two CPA-secure IB-PRE schemes without random oracles were proposed in [30]. Shao and Cao [31] proposed a generic construction for CCA-secure IB-PRE in the standard model. Recently, Liang et al. [32] proposed the first CCA-secure unidirectional single-hop IB-PRE in the standard model supporting conditional re-encryption.

2 Definitions and Security Models

Below we define the notion of CR-IB-PRE. Unless stated otherwise, by a CR-IB-PRE we mean a CR-IB-PRE with unidirectional and multi-hop properties. Note please refer to [18] for more details of these properties.

2.1 Definition of CR-IB-PRE

Definition 1. *A Cloud-Based Revocable Identity-Based Proxy Re-Encryption (CR-IB-PRE) scheme consists of the following algorithms. Below we let $\mathcal{I}, \mathcal{T}, \mathcal{M}$ be identity space, time space and message space, respectively.*

1. *Setup: the setup algorithm intakes a security parameter k and a maximal number of users N, and outputs the public parameters mpk, the master secret key msk, the initial state st and an empty revocation list RL. For simplicity, we assume the following algorithms include mpk implicitly.*
2. *KeyGen: the private key generation algorithm intakes msk, and a user's identity $id \in \mathcal{I}$, and outputs a private key sk_{id} for the user id and an updated state st.*
3. *TokenUp: the token update algorithm intakes msk, an identity id, a token update time period $T_i \in \mathcal{T}$, the current revocation list RL and st, and outputs a token τ_i, where $i \in [1, poly(1^k)]$.*
4. *DeKeyGen: the decryption key generation algorithm intakes sk_{id}, τ_i, and outputs a decryption key $sk_{id|i}$ for the user id under the time period T_i or $\perp$ if id has been revoked, where $i \in [1, poly(1^k)]$.*

5. *ReKeyGen: the re-encryption key generation algorithm intakes $sk_{id|i}$, msk, T_i and $T_{i'}$, and generates the re-encryption key as follows, where $1 \leq i < i'$.*
   (a) *ReKeyToken: the re-encryption key token generation algorithm intakes msk, T_i and $T_{i'}$, outputs a re-encryption key token $\varphi_{i \to i'}$.*
   (b) *ReKey: the re-encryption key algorithm intakes $sk_{id|i}$ and $\varphi_{i \to i'}$, outputs a re-encryption key $rk_{id|i \to i'}$ which can be used to transform a ciphertext under (id, T_i) to another ciphertext under $(id, T_{i'})$.*
6. *Enc: the encryption algorithm intakes id, T_i, and a message $m \in \mathcal{M}$, and outputs an original ciphertext C under (id, T_i) which can be further re-encrypted.*
7. *ReEnc: the re-encryption algorithm intakes $rk_{id|i \to i'}$, and a ciphertext C under (id, T_i), and outputs either a re-encrypted ciphertext C under $(id, T_{i'})$ or a symbol $\perp$ indicating C is invalid, where $1 \leq i < i'$.*
8. *Dec: the decryption algorithm intakes $sk_{id|i}$, and a ciphertext C under (id, T_i), and outputs either a message m or a symbol $\perp$ indicating C is invalid.*
9. *Revoke: the revocation algorithm intakes an identity to be revoked id, a revocation time period T_i, the current revocation list RL, and a state st, and outputs an updated RL.*

Remarks. Definition 1 is for our basic construction. In this paper we also present extensions for the basic construction. For the extended system, we reuse the above definition except that $TokenUp$ takes msk, ID, T_i, RL and st as input, and outputs a token τ_i for a set ID of non-revocable users.

Correctness: For any (mpk, msk) output by $Setup$, any time period $T_i \in \mathcal{T}$ (where $i \in [1, poly(1^k)]$), any message $m \in \mathcal{M}$, and all possible states st and revocation list RL, if sk_{id} is output by $KeyGen(msk,\ id)$, $\tau_i \leftarrow TokenUp(msk,\ id,\ T_i,\ RL,\ st)$, $sk_{id|i} \leftarrow DeKeyGen(sk_{id},\ \tau_i)$, $rk_{id|i \to j} \leftarrow ReKeyGen(sk_{id|i},\ msk,\ T_i,\ T_j)$ (note for simplicity we set $j = i + 1$ here), we have

if id is not revoked by $T_1 : Dec(sk_{id|1}, Enc(id, T_1, m)) = m$;

if id is not revoked by T_i :

$Dec(sk_{id|i}, ReEnc(rk_{id|i-1 \to i}, ..., ReEnc(rk_{id|1 \to 2}, Enc(id, T_1, m)))...) = m.$

2.2 Revocation Procedure

The revocation procedure is described based on different cases as follows.

1. **Decryption Key Compromised.** When the decryption key $sk_{id|i}$ of a user id for time period T_i is compromised by an adversary, the user id reports this issue to a PKG. The PKG then immediately returns a re-encryption key token $\varphi_{i \rightarrow j}$ to the user, where $j \neq i$ such that the user can generate a re-encryption key $rk_{id|i \rightarrow j}$. The user id further sends the re-encryption key to the proxy, and next requests it to re-encrypt all ciphertexts under (id, T_i) to the ones under (id, T_j). Besides, the PKG issues a token τ_j related to a new time period T_j to the user id. After receiving the token, the user id updates his/her decryption key from $sk_{id|i}$ to $sk_{id|j}$, and then uses the newly key to access the data. Note T_j is the time period satisfying $i < j$ such that the user id will update his key for decryption.
2. **Identity Expired.** When the identity of a user is expired (e.g. the resignation of a registered user) at time period T_i, our system notifies the corresponding identity and time period to a PKG. The PKG then generates a re-encryption key $rk_{id|i \rightarrow j}$, and requests the proxy to re-encrypt all ciphertexts under (id, T_i) to the ciphertexts under (id, T_j). Here j must satisfy $i < j$ such that the user id cannot reuse his/her decryption keys $sk_{id|z}$ (where $z \leq i$) to decrypt the re-encrypted ciphertexts. The PKG finally adds this user to the revocation list, that is, a re-encryption token and a token related to a new time period will not be issued to this user (after time period i).

2.3 Security Models

Below we define the CPA (IND-CPA) security notion for CR-IB-PRE systems. Before proceeding, we first define some notations used in our definition.

1. **Delegation chain.** Suppose in a CR-IB-PRE scheme there is a re-encryption key set $RK = \{rk_{id|i_1 \rightarrow i_2}, ..., rk_{id|i_{l-1} \rightarrow i_l}\}$ $(l \geq 2)$, for any re-encryption key $rk_{id|i_j \rightarrow i_{j+1}}$ in RK, $i_j \neq i_{j+1}$. We say that there exists a delegation chain under an identity id from time period T_{i_1} to time period T_{i_l}, denoted as $(id, T_{i_1} \rightarrow, ..., \rightarrow T_{i_l})$.
2. **Honest Delegation Chain.** Suppose there is a delegation chain under an identity id from time period T_i to time period T_j (i.e. $(id, T_i \rightarrow, ..., \rightarrow T_j)$). If there is no $sk_{id|z}$ being compromised by an adversary, then it is an honest delegation chain, where $z \in \{i, ..., j\}$. Otherwise, it is a corrupted one.

Definition 2. *A CR-IB-PRE scheme is IND-CPA secure if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ can win the game below with* *non-negligible advantage. In the game, $\mathcal{B}$ is the challenger, k is the security parameter, and N is a maximal number of users.*

1. ***Setup.*** *$\mathcal{B}$ runs $Setup(1^k, N)$, and next sends mpk to $\mathcal{A}$.*
2. ***Phase 1.***
   (a) *Private key oracle $\mathcal{O}_{sk}(id)$: on input an identity id, $\mathcal{B}$ runs $sk_{id} \leftarrow KeyGen(msk,\ id)$ and returns sk_{id} to $\mathcal{A}$.*
   (b) *TokenUpdate oracle $\mathcal{O}_{tu}(id, T_i)$: on input an identity id and a time period T_i, $\mathcal{B}$ runs $\tau_i \leftarrow TokenUp(msk,\ id,\ T_i, RL, st)$ and returns τ_i to $\mathcal{A}$.*
   (c) *DeKeyGen oracle $\mathcal{O}_{ku}(id, T_i)$: on input an identity id and a time period T_i, $\mathcal{B}$ returns $sk_{id|i} \leftarrow DeKeyGen(sk_{id},\ \tau_i)$ to $\mathcal{A}$, where $sk_{id} \leftarrow KeyGen(msk,\ id)$, and $\tau_i \leftarrow TokenUp(msk,\ id, T_i, RL, st)$.*
   (d) *Re-encryption key oracle $\mathcal{O}_{rk}(id,\ T_i,\ T_{i'})$: on input id, T_i and $T_{i'}$, $\mathcal{B}$ returns $rk_{id|i \rightarrow i'} \leftarrow ReKey(sk_{id|i},\ \varphi_{i \rightarrow i'})$ to $\mathcal{A}$, where $sk_{id|i} \leftarrow DeKeyGen(sk_{id},\ \tau_i)$, $\tau_i \leftarrow TokenUp(msk,\ id,\ T_i, RL, st)$, $sk_{id} \leftarrow KeyGen(msk,\ id)$, $\varphi_{i \rightarrow i'} \leftarrow ReKeyToken(msk,\ T_i,\ T_{i'})$, and $1 \leq i < i'$.*
   (e) *Revocation oracle $\mathcal{O}_{rv}(id, T_i)$: on input id and T_i, it returns the updated revocation list RL, where $RL \leftarrow Revoke(id, T_i, RL, st)$.*
3. ***Challenge.*** *$\mathcal{A}$ outputs two equal-length messages m_0 and m_1, a challenge identity id^*, and a challenge time period T_{i^*} to $\mathcal{B}$. $\mathcal{B}$ returns $C^* = Enc(id^*,\ T_{i^*},\ m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$.*
4. ***Phase 2.*** *$\mathcal{A}$ continues making queries as in Phase 1.*
5. ***Guess.*** *$\mathcal{A}$ outputs a guess bit $b' \in \{0, 1\}$. If $b' = b$, $\mathcal{A}$ wins.*

The following queries are forbidden to issue in the game.

1. *$\mathcal{O}_{tu}(id, T_i)$ and $\mathcal{O}_{rv}(id, T_i)$, where T_i is less than the time of all previous queries; or*
2. *$\mathcal{O}_{rv}(id, T_i)$, where (id, T_i) was already issued to $\mathcal{O}_{tu}$; or*
3. *If id^* was already issued to $\mathcal{O}_{sk}$, $\mathcal{O}_{rv}(id^*, T_i)$, where $T_i \geq T_{i^*}$; or*
4. *(id, T_i) is issued to $\mathcal{O}_{ku}$ before $\mathcal{O}_{tu}$ is queried on T_i.*
5. *$\mathcal{O}_{sk}(id^*)$ and $\mathcal{O}_{tu}(id^*, T_{i^*})$; or*
6. *$\mathcal{O}_{ku}(id^*, T_{i^*})$; or*
7. *$\mathcal{O}_{sk}(id^*)$ and $\mathcal{O}_{tu}(id^*, T_j)$, if id^* is in a delegation chain including T_{i^*} and T_j $(i^* < j)$; or*
8. *$\mathcal{O}_{ku}(id^*, T_j)$, if id^* is in a delegation chain including T_{i^*} and T_j $(i^* < j)$.*

Note $\mathcal{A}$ is allowed to request the challenger $\mathcal{B}$ to construct many delegation chains. However, a challenge identity id^* and a challenge time period T_{i^*} must be chosen from an honest delegation chain. The advantage of $\mathcal{A}$ is defined as $Adv^{IND\text{-}CPA}_{CR\text{-}IB\text{-}PRE,\mathcal{A}}(1^k, N) = |Pr[b' = b] - \frac{1}{2}|$.

We next define the collusion resistance notion for CR-IB-PRE systems. Here collusion resistance guarantees that given a re-encryption key $rk_{id|i \to i'}$ a proxy cannot compromise the entire decryption key $sk_{id|i}$ of a user id associated with time period T_i even compromising a decryption key $sk_{id|i'}$ of the same user associated with time period $T_{i'}$, where $i < i'$. We state that our notion here, to a large extent, is identical to that of traditional proxy re-encryption. We can regard (id, T_i) as an identity A, and $(id, T_{i'})$ as another identity B such that the decryption keys of A and B are $sk_{id|i}$ and $sk_{id|i'}$, respectively. Accordingly, our collusion resistance implies that even colluding with B the proxy cannot compromise the key of A, which is exact the notion of collusion resistance defined in the traditional proxy re-encryption setting.

Definition 3. *A CR-IB-PRE scheme is collusion resistant if the advantage $Adv_{\mathcal{A}}^{CR}(1^k, N)$ is negligible for any PPT adversary $\mathcal{A}$ in the following experiment. Let $\mathcal{O} = \{\mathcal{O}_{sk}, \mathcal{O}_{ku}, \mathcal{O}_{tu}, \mathcal{O}_{rk}, \mathcal{O}_{rv}\}$.*

$$Adv_{\mathcal{A}}^{CR}(1^k, N) = Pr[sk_{id|i} \in \Omega : (mpk, msk, st, RL) \leftarrow Setup(1^k, N); \\ sk_{id|i} \leftarrow \mathcal{A}^{\mathcal{O}}(mpk)]$$

where k is the security parameter, N is a maximal number of users, $sk_{id|i}$ must be uncorrupted decryption key for an identity id under time period T_i, Ω is the valid decryption key space, $\mathcal{O}_{sk}$, $\mathcal{O}_{tu}$, $\mathcal{O}_{ku}$, $\mathcal{O}_{rk}$ and $\mathcal{O}_{rv}$ are the oracles defined in Definition 2, but there is no constraint for querying the oracles except for the followings:

1. *$\mathcal{O}_{tu}(id, T_j)$ and $\mathcal{O}_{rv}(id, T_j)$, where T_j is less than the time of all previous queries; or*
2. *$\mathcal{O}_{rv}(id, T_j)$, where (id, T_j) was already issued to $\mathcal{O}_{tu}$; or*
3. *(id, T_j) cannot be issued to $\mathcal{O}_{ku}$ before $\mathcal{O}_{tu}$ was queried on T_j.*
4. *$\mathcal{O}_{ku}(id, T_i)$ cannot be issued; or*
5. *$\mathcal{O}_{sk}(id)$ and $\mathcal{O}_{tu}(id, T_i)$ cannot be issued simultaneously.*

3 Preliminaries

Bilinear Maps. Let *BSetup* denote an algorithm that, on input the security parameter 1^k, outputs the parameters for a bilinear map as $(q, g, \mathbb{G}, \mathbb{G}_T, e)$, where $\mathbb{G}$ and $\mathbb{G}_T$ are two multiplicative cyclic groups with prime order $q \in \Theta(2^k)$ and g is a generator of $\mathbb{G}$. The efficient mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has three properties: (1) *Bilinearity*: for all $g \in \mathbb{G}$ and $a, b \in_R \mathbb{Z}_q^*$, $e(g^a, g^b) = e(g, g)^{ab}$; (2) *Non-degeneracy*: $e(g, g) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the unit of $\mathbb{G}_T$; (3) *Computability*: e can be efficiently computed.

Definition 4. Computational Diffie-Hellman (CDH) Assumption. *Given a group $\mathbb{G}$ of prime order q with generator g and elements $g^a, g^b \in \mathbb{G}$, the CDH problem in $\mathbb{G}$ is to compute g^{ab}, where $a, b \in_R \mathbb{Z}_q^*$. We say that the CDH assumption holds in $\mathbb{G}$ if no PPT algorithm $\mathcal{A}$ can solve the CDH problem with non-negligible probability.*

Target Collision Resistant Hash Function. TCR hash function was introduced by Cramer and Shoup [33]. A TCR hash function H guarantees that given a random element x which is from the valid domain of H, a PPT adversary A cannot find $y \neq x$ such that $H(x) = H(y)$. We let $Adv_{H,A}^{TCR} = Pr[(x, y) \leftarrow \mathcal{A}(1^k) : H(x) = H(y), x \neq y, x, y \in DH]$ be the advantage of A in successfully finding collisions from a TCR hash function H, where DH is the valid input domain of H, k is the security parameter. If a hash function is chosen from a TCR hash function family, $Adv_{H,A}^{TCR}$ is negligible.

One-time Symmetric Encryption. A one-time symmetric encryption [33] consists of the following algorithms. Note let $\mathcal{K}_D$ be the key space $\{0,1\}^{poly(1^k)}$, and SYM be a symmetric encryption scheme. The encryption algorithm $SYM.Enc$ intakes a key $K \in \mathcal{K}_D$ and a message M, outputs a ciphertext C. The decryption algorithm $SYM.Dec$ intakes K and C, outputs M or a symbol $\perp$. The CCA security model for SYM systems is given in [20], we hence omit the details.

4 A New CPA-Secure CR-IB-PRE

4.1 A Basic Construction

To clearly show the technical roadmap of our scheme, we only propose our basic construction for CR-IB-PRE systems in this section. In this construction, a PKG will suffer from $O(N)$ computational complexity for key update phase. But we will present performance improvements for this basic construction in Section 5 such that the complexity of the PKG will reduce to $O(1)$. Below we assume any identity $id \in \{0,1\}^n$ and any time period $T_i \in \mathbb{Z}_q^*$. Some revocable IBE systems, such as [6], leverage KUNode algorithm [2] for efficient revocation. For simplicity, in our construction we do not focus on which data structure we choose to use to reduce the cost of key update. Instead, we concentrate on the combination of recoverability and re-encryption functionalities. We let state st be an unspecified data structure DS, and this state depends on which structure we use. For example, if using KUNode algorithm, we set st to be a binary tree.

1. **Setup**$(1^k, N)$. The setup algorithm runs $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathcal{G}(1^k)$, where q is the order of group $\mathbb{G}$. It chooses $\alpha, \beta \in_R \mathbb{Z}_q^*$, group elements $g_2, g_3, v_1, v_2 \in_R \mathbb{G}$, a random n-length set $U = \{u_j | 0 \leq j \leq n\}$, and a TCR hash function $TCR_1 : \mathbb{G} \rightarrow \mathbb{Z}_q^*$, where $u_j \in_R \mathbb{G}$. The public parameter is $mpk = (g, g_1, g_2, g_3, v_1, v_2, U, TCR_1)$, the master secret key is $msk = (g_2^\alpha, g_3^\beta)$, $RL = \emptyset$ and $st = DB$, where $g_1 = g^\alpha$.
2. **KeyGen**(msk, id). PKG chooses $r_{id} \in_R \mathbb{Z}_q^*$, sets the partial private key sk_{id} as

$$sk_{id_1} = g_3^\beta \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^{r_{id}}, sk_{id_2} = g^{r_{id}},$$

   where $\mathcal{V}_{id}$ is the set of all j for which the j-th bit (of id) is equal to 1.
3. **TokenUp**(msk, id, T_i, RL, st). PKG executes the token update algorithm works as follows. Choose $r_{T_i} \in_R \mathbb{Z}_q^*$, and set the token τ_i as

$$\tau_{i,1} = (g_2^\alpha / g_3^\beta) \cdot (v_1 \cdot v_2^{T_i})^{r_{T_i}}, \tau_{i,2} = g^{r_{T_i}},$$

   where i is the index for the time period.
4. **DeKeyGen**(sk_{id}, τ_i). A user id runs the algorithm as follows.
   (a) Choose $\tilde{r} \in_R \mathbb{Z}_q^*$, and randomize the token as

$$\tau_{i,1} = \tau_{i,1} \cdot (v_1 \cdot v_2^{T_i})^{\tilde{r}}, \tau_{i,2} = \tau_{i,2} \cdot g^{\tilde{r}}.$$

   (b) Choose $r_1, r_2 \in_R \mathbb{Z}_q^*$, and set the updated secret key $sk_{id|i}$ for identity id and time period T_i as

$$\begin{aligned} sk_{id|i,1} &= sk_{id_1} \cdot \tau_{i,1} \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^{r_1} \cdot (v_1 \cdot v_2^{T_i})^{r_2} \\ &= g_2^\alpha \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^{\hat{r}_1} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r}_2}, \end{aligned}$$

$$sk_{id|i,2} = sk_{id_2} \cdot g^{r_1} = g^{\hat{r}_1}, sk_{id|i,3} = \tau_{i,2} \cdot g^{r_2} = g^{\hat{r}_2},$$

   where $\hat{r}_1 = r_{id} + r_1$, $\hat{r}_2 = r_{T_i} + \tilde{r} + r_2$. Note the user will share $r_1, r_2, \tilde{r}$ with the PKG (suppose it is fully trusted) such that the PKG can store $(id|i, \hat{r}_1, \hat{r}_2)$ in a list $List^{sk_{id|i}}$ for further use.
5. **ReKeyGen**$(sk_{id|i}, msk, T_i, T_{i'})$. The re-encryption key $rk_{id|i \rightarrow i'}$ is generated as follows.

(a) $ReKeyToken(msk, T_i, T_{i'})$: If a user id holding $sk_{id|i}$ is allowed to update his key to another time period $T_{i'}$, PKG first generates the re-encryption key token $\varphi_{i\to i'}$ as

$$\varphi_{i\to i'}^{(1)} = (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)}/(v_1 \cdot v_2^{T_i})^{\hat{r}_2},$$
$$\varphi_{i\to i'}^{(2)} = (\hat{C}_0, \hat{C}_1, \hat{C}_2, \hat{C}_3) \leftarrow Enc(id, T_{i'}, \xi),$$

where $\xi \in_R \mathbb{G}_T$, $\hat{r}_2$ is recovered from $(id|i'$, $\hat{r}_1$, $\hat{r}_2)$ which is stored the $List^{sk_{id|i}}$.

(b) $ReKey(sk_{id|i}, \varphi_{i\to i'})$: After receiving $\varphi_{i\to i'}$ from PKG, the user id generates the re-encryption key as follows.

i. Choose $\rho \in_R \mathbb{Z}_q^*$, and set

$$rk_1 = sk_{id|i,1} \cdot \varphi_{i\to i'}^{(1)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho}, rk_2 = sk_{id|i,2} \cdot g^{\rho}, rk_3 = \varphi_{i\to i'}^{(2)}.$$

ii. Output the re-encryption key $rk_{id|i\to i'} = (rk_1,\ rk_2,\ rk_3)$.

6. **Enc**(id, T_i, m). Given an identity id, a time period T_i, and a message $m \in \mathbb{G}_T$, the encryption algorithm chooses $t \in_R \mathbb{Z}_q^*$, and sets the original ciphertext C as

$$C_0 = m \cdot e(g_1, g_2)^t, C_1 = g^t, C_2 = (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^t, C_3 = (v_1 \cdot v_2^{T_i})^t.$$

We assume that the identity id and the time period T_i are implicitly included in the ciphertext.

7. **ReEnc**$(rk_{id|i\to i'}, C)$. Parse the ciphertext C under (id, T_i) as $(C_0$, C_1, C_2, $C_3)$, and the re-encryption key $rk_{id|i\to i'}$ as $(rk_1$, rk_2, $rk_3)$. The re-encryption algorithm computes $C_4 = \frac{e(C_1, rk_1)}{e(C_2, rk_2)} = e(g^t, g_2^{\alpha} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)})$, and next sets the re-encrypted ciphertext C under $(id, T_{i'})$ as $(C_0$, C_1, C_4, $rk_3)$. Note if C under $(id, T_{i'})$ needs to be further re-encrypted to the time period $T_{i''}$, then the proxy parses rk_3 as $(\hat{C}_0$, $\hat{C}_1$, $\hat{C}_2$, $\hat{C}_3)$. Given a re-encryption key $rk_{id|i'\to i''} = (rk'_1$, rk'_2, $rk'_3)$, the proxy computes $C'_4 = \frac{e(\hat{C}_1, rk'_1)}{e(\hat{C}_2, rk'_2)}$, and sets the ciphertext C under $(id, T_{i''})$ as $(C_0$, C_1, C_4, $\hat{C}_0$, $\hat{C}_1$, C'_4, $rk'_3)$.

8. **Dec**$(sk_{id|i}, C)$. Given a ciphertext C under (id, T_i), the decryption algorithm works as follows.

(a) For the original ciphertext $C = (C_0, C_1, C_2, C_3)$, the decryptor computes $\frac{e(C_1, sk_{id|i,1})}{e(C_2, sk_{id|i,2})e(C_3, sk_{id|i,3})} = e(g_1, g_2)^t$, and outputs the message $C_0/e(g_1, g_2)^t = m \cdot e(g_1, g_2)^t/e(g_1, g_2)^t = m$.

(b) For the re-encrypted ciphertext C:

   i. If the re-encrypted ciphertext is re-encrypted only once, i.e. C=(C_0, C_1, C_4, $rk_3 = (\hat{C}_0,\ \hat{C}_1,\ \hat{C}_2,\ \hat{C}_3)$), then the decryptor computes $\frac{\hat{C}_0 e(\hat{C}_2, sk_{id|i,2}) e(\hat{C}_3, sk_{id|i,3})}{e(\hat{C}_1, sk_{id|i,1})} = \xi$. Accordingly, the decryptor can finally computer $C_0 \frac{e(C_1,(v_1 v_2^{T_i})^{TCR_1(\xi)})}{C_4} = m$.

   ii. If the ciphertext under id is re-encrypted l times from time period T_1 to T_{l+1}, we denote the re-encrypted ciphertext as $C^{(l+1)}$=($C_0^{(1)}$, $C_1^{(1)}$, $C_4^{(1)}$, ..., $C_0^{(l)}$, $C_1^{(l)}$, $C_4^{(l)}$, $rk_3^{(l+1)}$), where $C_0^{(1)}$ and $C_1^{(1)}$ are the components of original ciphertext under (id, T_1), and $rk_3^{(i+1)} = (C_0^{(i+1)},\ C_1^{(i+1)},\ C_2^{(i+1)},\ C_3^{(i+1)})$ is the ciphertext under (id, T_{i+1}), $i \in [1, l]$. We recover the message m as follows.

$$\text{First set:} \frac{C_0^{(l+1)} e(C_2^{(l+1)}, sk_{id|l+1,2}) e(C_3^{(l+1)}, sk_{id|l+1,3})}{e(C_1^{(l+1)}, sk_{id|l+1,1})} = \xi^{(l)},$$

$$\text{from } i = 2 \text{ to } l \text{ set} : C_0^{(i)} \frac{e(C_1^{(i)}, (v_1 v_2^{T_{i+1}})^{TCR_1(\xi^{(i)})})}{C_4^{(i)}} = \xi^{(i-1)},$$

$$\text{finally compute} : C_0^{(1)} \frac{e(C_1^{(1)}, (v_1 v_2^{T_2})^{TCR_1(\xi^{(1)})})}{C_4^{(1)}} = m.$$

9. **Revoke**(id, T_i, RL, st). Update the revocation list by $RL \leftarrow RL \cup \{id, T_i\}$ and return the updated revocation list.

4.2 Security Analysis

Theorem 1. *Suppose the underlying Waters IBE scheme is IND-CPA secure, TCR_1 is the TCR hash function, our CR-IB-PRE scheme is IND-CPA secure in the standard model.*

Due to limited space, we provide the security proof for Theorem 1 in Appendix 9.1.

Theorem 2. *Suppose the CDH assumption holds, our CR-IB-PRE scheme is collusion resistant.*

Due to limited space, we provide the security proof for Theorem 2 in Appendix 9.2.

5 Performance Improvement

5.1 Reduce the complexity of key update

In our basic construction the complexity of the key update phase (in terms of communication and computation) is linearly in the number (say N) of users whom are excluded in the revocation list, i.e. $O(N)$. We use the following method to reduce the complexity $O(N)$ to $O(1)$. From the algorithm $TokenUp$, we can see that the identity id does not take part in the generation of the token τ_i. This gives us a possibility to broadcast this token for time period T_i to all non-revocable users. Below we only employ a broadcast encryption in our basic construction. We choose Phan et al. broadcast encryption system [34] as a building block. Note system implementors may choose an appropriate broadcast encryption for different purposes, e.g., efficiency and security. We let $SYM = (SYM.Enc, SYM.Dec)$ denote a one-time symmetric encryption system in which encryption algorithm $SYM.Enc$ intakes a message m and a symmetric key $K \in \{0,1\}^{poly(1^k)}$ and outputs a ciphertext, and decryption algorithm $SYM.Dec$ intakes a ciphertext and a symmetric key K and outputs a message m. We only show the modification for our basic system as follows.

1. **Setup**$(1^k, N)$. The setup algorithm additionally chooses $\gamma, \hat{\alpha} \in_R \mathbb{Z}_q^*$, a TCR hash function $TCR_2 : \mathbb{G}_T \rightarrow \{0,1\}^{poly(1^k)}$, and adds $v_0 = g^{\gamma}$ and TCR_2 to the public parameter mpk, and $(\gamma, \hat{\alpha})$ to the master secret key msk.
2. **KeyGen**(msk, id). PKG generates a new key component $sk_{id_3} = g_z^{\gamma}$, and sets additional public parameters $g_z = g^{\hat{\alpha}^z}$, $g_{z+1} = g^{\hat{\alpha}^{z+1}}$, $g_{\lambda+1-z} = g^{\hat{\alpha}^{\lambda+1-z}}$, $g_{\lambda+1+z} = g^{\hat{\alpha}^{\lambda+1+z}}$ for user id, where z is the index for identity id, and $\lambda - 1 = N$.
3. **TokenUp**(msk, ID, T_i, RL, st). Note ID now is a set of identities. After constructing the token $\tau_i = (\tau_{i,1}, \tau_{i,2})$, PKG works as follows.
   (a) Choose $\hat{t} \in_R \mathbb{Z}_q^*$, $K \in_R \mathbb{G}_T$, and set an encryption $C_{\tau_i}^{(1)}$ as

   $$ID, \mathcal{T}_1 = K \cdot e(g_{\lambda+1}, g)^{\hat{t}}, \mathcal{T}_2 = g^{\hat{t}}, \mathcal{T}_3 = (v_0 \cdot \prod_{w \in ID} g_{\lambda+1-w})^{\hat{t}}.$$

   (b) Run $C_{\tau_i}^{(2)} \leftarrow SYM.Enc(TCR_2(K), \tau_{i,1}||\tau_{i,1})$, and next upload the token $\tau_i = (C_{\tau_i}^{(1)}, C_{\tau_i}^{(2)})$ for a set ID of identities to the cloud server.

4. **DeKeyGen**(sk_{id}, τ_i). Before constructing a decryption key as in the algorithm $DeKeyGen$ of our basic scheme, a user id (where $id \in ID$) first recovers the token as follows. The user computes

$$\begin{aligned} K &= \mathcal{T}_1 e(\mathcal{T}_3, g_i)/e(sk_{id_3} \prod_{w \in ID \setminus \{z\}} g_{\lambda+1-w+z}, \mathcal{T}_2) \\ &= Ke(g_{\lambda+1}, g)^{\hat{t}} e((g^{\gamma} \prod_{w \in ID} g_{\lambda+1-w})^{\hat{t}}, g_z)/e(g_z^{\gamma} \prod_{w \in ID \setminus \{z\}} g_{\lambda+1-w+z}, g^{\hat{t}}) \end{aligned}$$

   and runs $\sigma_i = \tau_{i,1} || \tau_{i,1} \leftarrow SYM.Dec(TCR_2(K), C_{\tau_i}^{(2)})$.

Note the rest of the algorithms are the same as that of our basic scheme.

5.2 Reduce size of re-encrypted ciphertext and decryption complexity

Our basic construction suffers from a drawback that the size of re-encrypted ciphertext and the complexity of decryption expand linearly in the number of time periods. If there are t available time periods for a system user, the re-encrypted ciphertext size at time period T_t increases from $O(1)$ up to $O(t)$ as well as the corresponding decryption complexity for this re-encrypted ciphertext. To reduce the complexity to be constant, we leverage the following idea.

We can delegate the generation of re-encryption key to PKG as PKG has knowledge of private keys of all system users and tokens of all time periods. Here system users can only focus on decryption key generation, message encryption and decryption, i.e. the universal cost of using a revocable IBE system, such that the re-encryption functionality and its corresponding workload are transparent in the view of the users.

Being granted the rights of re-encryption key generation, PKG works as follows. Suppose the decryption keys of a user id associated with time periods T_i and T_j are $sk_{id|i} = (sk_{id|i,1},\ sk_{id|i,2},\ sk_{id|i,3})$ and $sk_{id|j} = (sk_{id|j,1}, sk_{id|j,2}, sk_{id|j,3})$, and the corresponding tuples stored in the list $List^{sk_{id|z}}$ are $(id|i,\ \hat{r}_{i,1},\ \hat{r}_{i,2})$ and $(id|j,\ \hat{r}_{i,1},\ \hat{r}_{i,2})$, where $i < j$ (for simplicity we may set $j = i + 1$). PKG then constructs the re-encryption key $rk_{id|i \rightarrow j}$ as

$$rk_1 = (v_1 \cdot v_2^{T_j})^{-\hat{r}_{j,2}} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r}_{i,2}} \cdot (v_1 \cdot v_2^{T_1})^{\gamma}, rk_2 = sk_{id|j,3}^{-1} \cdot sk_{id|i,3} = g^{\hat{r}_{i,2} - \hat{r}_{j,2}} \cdot g^{\gamma},$$

where $\gamma \in_R \mathbb{Z}_q^*$.

For simplicity, we suppose a user id has l available time periods (in which T_1 is the first time period, and T_l is the last one). Given $rk_{id|1\to 2} = (rk_{1\to 2,1}, rk_{1\to 2,2})$, the re-encryption algorithm $ReEnc$ computes

$$\begin{aligned} C_4^{(1)} &= \frac{e(C_3, rk_{1\to 2,2})}{e(C_1, rk_{1\to 2,1})} \\ &= \frac{e((v_1 v_2^{T_1})^t, g^{\hat{r}_{1,2}-\hat{r}_{2,2}} g^{\gamma})}{e(g^t, (v_1 v_2^{T_2})^{-\hat{r}_{2,2}} (v_1 v_2^{T_1})^{\hat{r}_{1,2}} (v_1 v_2^{T_1})^{\gamma})} \\ &= \frac{e((v_1 v_2^{T_1})^t, g^{-\hat{r}_{2,2}})}{e(g^t, (v_1 v_2^{T_2})^{-\hat{r}_{2,2}})}, \end{aligned}$$

and next sets the re-encrypted ciphertext C under (id, T_2) as $(C_0$, C_1, C_2, C_3, $C_4^{(1)})$, where an original ciphertext C under (id, T_1) is $C_0 = m{\cdot}e(g_1, g_2)^t$, $C_1 = g^t$, $C_2 = (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^t$, $C_3 = (v_1{\cdot}v_2^{T_1})^t$. At time period T_l, the re-encrypted ciphertext C under (id, T_l) is $(C_0, C_1, C_2, C_3, C_4^{(l-1)})$, in which

$$\begin{aligned} C_4^{(l-1)} &= C_4^{(l-2)} \cdot \frac{e(C_3, rk_{l-1\to l,2})}{e(C_1, rk_{l-1\to l,1})} \\ &= \frac{e((v_1 \cdot v_2^{T_1})^t, g^{-\hat{r}_{l-1,2}})}{e(g^t, (v_1 \cdot v_2^{T_{l-1}})^{-\hat{r}_{l-1,2}})} \cdot \frac{e((v_1 \cdot v_2^{T_1})^t, g^{\hat{r}_{l-1,2}-\hat{r}_{l,2}} \cdot g^{\gamma})}{e(g^t, (v_1 \cdot v_2^{T_l})^{-\hat{r}_{l,2}} \cdot (v_1 \cdot v_2^{T_{l-1}})^{\hat{r}_{l-1,2}} \cdot (v_1 \cdot v_2^{T_1})^{\gamma})} \\ &= \frac{e((v_1 \cdot v_2^{T_1})^t, g^{-\hat{r}_{l,2}})}{e(g^t, (v_1 \cdot v_2^{T_l})^{-\hat{r}_{l,2}})}, \end{aligned}$$

and $C_4^{(l-2)}$ is a component of ciphertext C under (id, T_{l-1}).

The decryption algorithm Dec works as follows. Given $sk_{id|i} = (sk_{id|i,1}$, $sk_{id|i,2}$, $sk_{id|i,3})$ and a re-encrypted ciphertext $(C_0, C_1, C_2, C_3, C_4^{(i-1)})$ under (id, i), set

$$\begin{aligned} & C_0 \cdot C_4^{(i-1)} \cdot \frac{e(sk_{id|i,3}, C_3) \cdot e(C_2, sk_{id|i,2})}{e(sk_{id|i,1}, C_1)} \\ & = m \cdot e(g_1, g_2)^t \cdot \frac{e((v_1 \cdot v_2^{T_1})^t, g^{-\hat{r}_{i,2}})}{e(g^t, (v_1 \cdot v_2^{T_i})^{-\hat{r}_{i,2}})} \cdot \frac{e(g^{\hat{r}_{i,2}}, (v_1 \cdot v_2^{T_1})^t) \cdot e((u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^t, g^{\hat{r}_{i,1}})}{e(g_2^{\alpha} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\hat{r}_{i,1}} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r}_{i,2}}, g^t)} \\ & = \frac{m \cdot e(g_1, g_2)^t}{e(g_2^{\alpha}, g^t)} = m. \end{aligned}$$

6 Comparison

In this section we compare our improved version with an ABE system supporting revocability [3] and the most efficient revocable IBE scheme [9] in terms of security, functionality and efficiency. Table 1 illustrates the comparison of security and functionality, Table 2 depicts the comparison of computational cost, and Table 3 shows the comparison of communication complexity. Note we do not compare our scheme with the existing IB-PRE schemes here as we pay more attention in the functionality of revocability.

To define the notations and parameters used in the Tables, we let $|\mathbb{G}|$ denote the bit-length of an element in $\mathbb{G}$, and $|\mathbb{G}_T|$ denote the bit-length of an element in $\mathbb{G}_T$, $|U|$ denote the number of attributes used in the system, $|f|$ denote the size of an access formula, $|S|$ denote the size of an attribute set, n denote the bit-length of an identity, c_p, c_e, c_e^T denote the computation cost of a bilinear pairing, an exponentiation in $\mathbb{G}$ and in $\mathbb{G}_T$, respectively. Suppose [3], [9] and our scheme share the same number (N) of non-revocable system users. It can be seen that [3] only presents generic constructions for the revocable ABE systems. To bring convenience for the comparison, we use Waters ABE scheme [35] to implement one of the generic constructions of [3]. The implementation yields a revocable CP-ABE system.

Table 1. Security and Functionality Comparison

Schemes	**Security**	**Complexity Assumption**	**Delegation of Decryption Rights**
[3]	CPA	decisional q-parallel-BDHE	×
[9]	CPA	decisional BDH	×
Ours	CPA, Collusion Resistance	decisional BDH and CDH	✓

From Table 1, we see that our scheme supports not only revocability but also re-encryption with CPA security and collusion resistance under the decisional Bilinear Diffie-Hellman (BDH) assumption and CDH assumption, respectively. [3] is CPA secure under the decisional q-parallel Bilinear Diffie-Hellman Exponent (BDHE) assumption (suppose it is built on Waters ABE) but only supporting revocability, whereas ours additionally enjoys the delegation of decryption rights. Compared to [9], supporting revocability with CPA security under the decisional BDH assumption, ours offers additional property without degrading security level. Note [9] and our scheme are secure under simple complexity assumptions, while [3] relies on a complex assumption.

Table 2. Computation Cost Comparison

Schemes	Computation Cost				
	Encryption	Decryption	Key	Update Info. (N Users)	PKG
[3]	$O(\lvert f\rvert)c_e + O(1)c_e^T$	$O(\lvert S\rvert)(c_p + c_e^T)$	$O(\lvert S\rvert)c_e$	$\epsilon_1 = O(N \cdot \lvert S\rvert)c_e$	$O(N)\epsilon_1$
[9]	$O(1)c_e + O(1)c_e^T$	$O(1)c_p$	$O(1)c_e$	$\epsilon_2 = O(N)c_e$	$O(N)\epsilon_2$
Ours	$O(1)c_e + O(1)c_e^T$	$O(1)c_p$	$O(1)c_e$	$\epsilon_3 = O(1)c_e + O(1)c_e^T$	$O(1)\epsilon_3$

From Table 2, we see that [3] suffers from the largest complexity in each merit, and the PKG of [9] suffers from $O(N)$ computational complexity in updating key information for each time period. Besides, both [3] and [9] require secure communication channel from PKG to each non-revocable user (for issuing update key information), while our system can eliminate the cost spent on this. Compared with [3,9], ours enjoys constant complexity in each merit. To achieve the re-encryption property, we need $O(1)c_e$ and $O(1)c_p$ additional cost in generating re-encryption key and re-encrypted ciphertext, respectively. However, when comparing with the linear complexity of [3], we state that the above additional cost for re-encryption purpose is acceptable.

Table 3. Communication Cost Comparison

Schemes	Communication Cost		
	Private Key Size	Ciphertext Size	Update Info. Size (N Users)
[3]	$O(\lvert S\rvert)\lvert\mathbb{G}\rvert$	$O(\lvert f\rvert)\lvert\mathbb{G}\rvert + O(1)\lvert\mathbb{G}_T\rvert$	$O(N \cdot \lvert S\rvert)\lvert\mathbb{G}\rvert$
[9]	$O(1)\lvert\mathbb{G}\rvert$	$O(1)\lvert\mathbb{G}\rvert + O(1)\lvert\mathbb{G}_T\rvert$	$O(N)\lvert\mathbb{G}\rvert$
Ours	$O(1)\lvert\mathbb{G}\rvert$	$O(1)\lvert\mathbb{G}\rvert + O(1)\lvert\mathbb{G}_T\rvert$	$O(1)\lvert\mathbb{G}\rvert + O(1)\lvert\mathbb{G}_T\rvert$

Table 3 shows that [9] and our scheme achieve the least complexity, while [3] still suffers from linear cost in each merit. Although our scheme requires additional cost $O(1)|\mathbb{G}|$ in delivering re-encryption key, it is worth mentioning that our scheme enjoys constant communication cost in updating key information for each time period but [3,9] suffers from $O(N)$ complexity. As N increases, our scheme has better efficiency in communication.

7 Conclusions

In this paper we proposed the first efficient CR-IB-PRE scheme supporting revocability. We also proved our scheme secure against CPA and collusion attacks in the standard model. Without knowledge of key update information, no revoked system users can decrypt the ciphertexts stored in the cloud (due to the revocable functionality). When compared with the naive solution of converting any IB-PRE system to achieve revocable capability, our scheme only requires PKG to publish constant-size key update information for all non-revocable users once at the beginning of each time period such that the computational complexity of the PKG in generating key update information and the cost of building communication channel between the PKG and each user are both reduced to $O(1)$. In addition, our scheme offers more flexible data sharing mechanism to system users by supporting the property of re-encryption compared to the traditional revocable IBE systems.

This paper motivates some interesting open problems, for example, how to construct a CCA-secure CR-IB-PRE scheme in the standard model.

8 Acknowledgements

This work is supported by A*STAR funded project SecDC-112172014.

References

1. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: CRYPTO '01. Volume 2139 of LNCS., Springer (2001) 213–229
2. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In Ning, P., Syverson, P.F., Jha, S., eds.: ACM Conference on Computer and Communications Security, ACM (2008) 417–426
3. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In Safavi-Naini, R., Canetti, R., eds.: CRYPTO. Volume 7417 of Lecture Notes in Computer Science., Springer (2012) 199–217
4. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: ACNS '07. Volume 4512 of LNCS., Springer (2007) 288–306
5. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In Cramer, R., ed.: Advances in Cryptology EUROCRYPT 2005. Volume 3494 of LNCS. Springer Berlin Heidelberg (2005) 457–473
6. Libert, B., Vergnaud, D.: Adaptive-id secure revocable identity-based encryption. In Fischlin, M., ed.: CT-RSA. Volume 5473 of Lecture Notes in Computer Science., Springer (2009) 1–15

7. Waters, B.: Efficient identity-based encryption without random oracles. In Cramer, R., ed.: EUROCRYPT. Volume 3494 of Lecture Notes in Computer Science., Springer (2005) 114–127
8. Gentry, C.: Practical identity-based encryption without random oracles. In: EUROCRYPT '06. Volume 4004 of LNCS. Springer (2006) 445–464
9. Seo, J.H., Emura, K.: Efficient delegation of key generation and revocation functionalities in identity-based encryption. In Dawson, E., ed.: CT-RSA. Volume 7779 of Lecture Notes in Computer Science., Springer (2013) 343–358
10. Baek, J., Zheng, Y.: Identity-based threshold decryption. In Bao, F., Deng, R.H., Zhou, J., eds.: Public Key Cryptography. Volume 2947 of Lecture Notes in Computer Science., Springer (2004) 262–276
11. Ding, X., Tsudik, G.: Simple identity-based cryptography with mediated rsa. In Joye, M., ed.: CT-RSA. Volume 2612 of Lecture Notes in Computer Science., Springer (2003) 193–210
12. Libert, B., Quisquater, J.J.: Efficient revocation and threshold pairing based cryptosystems. In Borowsky, E., Rajsbaum, S., eds.: PODC, ACM (2003) 163–171
13. Attrapadung, N., Imai, H.: Attribute-based encryption supporting direct/indirect revocation modes. In Parker, M.G., ed.: IMA Int. Conf. Volume 5921 of Lecture Notes in Computer Science., Springer (2009) 278–300
14. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In Shacham, H., Waters, B., eds.: Pairing. Volume 5671 of Lecture Notes in Computer Science., Springer (2009) 248–265
15. Nieto, J.M.G., Manulis, M., Sun, D.: Fully private revocable predicate encryption. In Susilo, W., Mu, Y., Seberry, J., eds.: ACISP. Volume 7372 of Lecture Notes in Computer Science., Springer (2012) 350–363
16. Mambo, M., Okamoto, E.: Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. IEICE Transactions **E80-A**(1) (1997) 54–63
17. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In Nyberg, K., ed.: EUROCRYPT. Volume 1403 of LNCS., Springer (1998) 127–144
18. 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) (2006) 1–30
19. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In Ning, P., di Vimercati, S.D.C., Syverson, P.F., eds.: ACM Conference on Computer and Communications Security, ACM (2007) 185–194
20. Isshiki, T., Nguyen, M.H., Tanaka, K.: Proxy re-encryption in a stronger security model extended from ct-rsa2012. In: CT-RSA 2012. Volume 7779 of LNCS., Springer (2013) 277–292
21. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In Cramer, R., ed.: Public Key Cryptography. Volume 4939 of LNCS., Springer (2008) 360–379
22. Hanaoka, G., Kawai, Y., Kunihiro, N., Matsuda, T., Weng, J., Zhang, R., Zhao, Y.: Generic construction of chosen ciphertext secure proxy re-encryption. In Dunkelman, O., ed.: Topics in Cryptology - CT-RSA 2012. Volume 7178 of LNCS. Springer Berlin Heidelberg (2012) 349–364
23. Tang, Q., Hartel, P.H., Jonker, W.: Inter-domain identity-based proxy re-encryption. In Yung, M., Liu, P., Lin, D., eds.: Inscrypt. Volume 5487 of Lecture Notes in Computer Science., Springer (2008) 332–347

24. 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. Volume 4779 of LNCS., Springer (2007) 189–202
25. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In Boneh, D., ed.: CRYPTO. Volume 2729 of LNCS., Springer (2003) 565–582
26. Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Pairing '07. Volume 4575 of LNCS., Springer (2007) 247–267
27. Wang, L., Wang, L., Mambo, M., Okamoto, E.: Identity-based proxy cryptosystems with revocability and hierarchical confidentialities. IEICE Transactions **95-A**(1) (2012) 70–88
28. Wang, L., Wang, L., Mambo, M., Okamoto, E.: New identity-based proxy re-encryption schemes to prevent collusion attacks. In Joye, M., Miyaji, A., Otsuka, A., eds.: Pairing. Volume 6487 of Lecture Notes in Computer Science., Springer (2010) 327–346
29. Mizuno, T., Doi, H.: Secure and efficient IBE-PKE proxy re-encryption. IEICE Transactions **E94-A**(1) (2011) 36–44
30. Luo, S., Shen, Q., Chen, Z.: Fully secure unidirectional identity-based proxy re-encryption. In Kim, H., ed.: ICISC. Volume 7259 of LNCS., Springer, Heidelberg (2011) 109–126
31. Shao, J., Cao, Z.: Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption. Information Sciences **206**(0) (2012) 83 – 95
32. Liang, K., Liu, Z., Tan, X., Wong, D.S., Tang, C.: A cca-secure identity-based conditional proxy re-encryption without random oracles. In Kwon, T., Lee, M.K., Kwon, D., eds.: ICISC. Volume 7839 of LNCS., Springer (2012) 231–246
33. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1) (January 2004) 167–226
34. Phan, D.H., Pointcheval, D., Shahandashti, S.F., Strefler, M.: Adaptive cca broadcast encryption with constant-size secret keys and ciphertexts. Int. J. Inf. Sec. **12**(4) (2013) 251–265
35. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A., eds.: Public Key Cryptography. Volume 6571 of LNCS., Springer (2011) 53–70

9 Appendix

9.1 Proof for Theorem 1

Proof. Suppose there exists a PPT algorithm $\mathcal{A}$ who can break our system, we then construct a reduction algorithm $\mathcal{B}$ to break the IND-CPA security of Waters IBE scheme by using $\mathcal{A}$. $\mathcal{B}$ plays the following game with $\mathcal{A}$ by accessing the challenger $\mathcal{C}_{IBE}$ and the private key extraction oracle $\mathcal{O}_{IBE}$ of the Waters IBE scheme.

Setup. $\mathcal{C}_{IBE}$ sends the public parameter $pp_{IBE} = (g, g_1, g_2, u_0, u_1, ..., u_n)$ of Waters IBE scheme to $\mathcal{B}$. $\mathcal{B}$ then chooses a TCR hash function TCR_1 as in the real scheme, $g_3 \in_R \mathbb{G}$, $\beta, \gamma_1, \gamma_2 \in_R \mathbb{Z}_q^*$ and sets $v_0 = g^{\gamma}$,

$v_1 = g_1^{-T_{i^*}} g^{\gamma_2}$ and $v_2 = g_1 g^{\gamma_1}$, where $\mathcal{B}$ has to guess the challenge time period $T_{i^*} \in \mathcal{T}$, and $\mathcal{T}$ is the space of time period. Finally, $\mathcal{B}$ outputs the public parameter $mpk = (pp_{IBE},\, g_3,\, v_1,\, v_2,\, TCR_1)$ to $\mathcal{A}$. In the view of $\mathcal{A}$, the distribution of pp is identical to that of the real scheme.

$\mathcal{B}$ also maintains the following lists.

1. $List^{sk}$ stores the tuples (id, sk_{id}) which are generated in $\mathcal{O}_{sk}$.
2. $List^{\tau}$ stores the tuples (id, T_i, τ_i) which are generated in $\mathcal{O}_{tu}$.
3. $List^{ku}$ stores the tuples $(id, T_i, sk_{id|i})$ which are generated in $\mathcal{O}_{ku}$.
4. $List^{rk}$ stores the tuples $(id, T_i, T_{i'}, rk_{id|i \to i'}, tag_1, tag_2)$ which are generated in $\mathcal{O}_{rk}$, where tag_1 is used to denote whether the re-encryption key is randomly constructed or not, and tag_2 is used to denote whether the re-encryption key is valid or not.

Phase 1.

1. Private key queries $\mathcal{O}_{sk}(id)$: $\mathcal{A}$ issues id to $\mathcal{B}$, $\mathcal{B}$ responds the private key as follows.

   (a) If $id^* = id$, $\mathcal{B}$ sets $sk_{id_1} = g_3^{\beta} \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^{r_{id}}$ and $sk_{id_2} = g^{r_{id}}$, where $r_{id} \in_R \mathbb{Z}_q^*$.

   (b) Otherwise, $\mathcal{B}$ issues id to the private key extraction oracle $\mathcal{O}_{IBE}$, and receives the key $sk_{id}^{IBE} = (sk_{id_1}^{IBE}, sk_{id_2}^{IBE})$. $\mathcal{B}$ then sets $sk_{id_1} = g_3^{\beta} \cdot sk_{id_1}^{IBE}$ and $sk_{id_2} = sk_{id_2}^{IBE}$.

   Finally, $\mathcal{B}$ returns the key to $\mathcal{A}$ and adds the tuple (id, sk_{id}) to the list $List^{sk}$.

2. Token update queries $\mathcal{O}_{tu}(id, T_i)$: $\mathcal{A}$ issues (id, T_i) to $\mathcal{B}$, $\mathcal{B}$ constructs the updated token for the time period T_i as follows.

   (a) If $id^* = id$ and $T_{i^*} = T_i$, $\mathcal{B}$ outputs $\perp$.

   (b) If $id^* = id$ and $T_{i^*} \neq T_i$, $\mathcal{B}$ the token τ_i as

$$\tau_{i,1} = g_3^{-\beta} \cdot g_2^{\frac{-\gamma_2 - \gamma_1 \cdot T_{i^*}}{T_i - T_{i^*}}} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r'_{T_i}}, \tau_{i,2} = g_2^{\frac{-1}{T_i - T_{i^*}}} g^{r'_{T_i}},$$

where $r'_{T_i} \in_R \mathbb{Z}_q^*$. One can check the validity of the above token by implicitly setting $r_{T_i} = r'_{T_i} - \frac{\kappa}{T_i - T_{i^*}}$ and $g_2 = g^\kappa$. Then we have

$$\begin{aligned}
\tau_{i,1} &= g_3^{-\beta} \cdot g_2^{\frac{-\gamma_2 - \gamma_1 \cdot T_{i^*}}{T_i - T_{i^*}}} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r'_{T_i}} \\
&= g_3^{-\beta} \cdot g_2^{\alpha} \cdot g_2^{-\alpha} \cdot g_2^{\frac{-\gamma_2 - \gamma_1 \cdot T_{i^*}}{T_i - T_{i^*}}} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r'_{T_i}} \\
&= g_3^{-\beta} \cdot g_2^{\alpha} \cdot g^{-\alpha\kappa} \cdot g^{\frac{(-\gamma_2 - \gamma_1 \cdot T_{i^*}) \cdot \kappa}{T_i - T_{i^*}}} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r'_{T_i}} \\
&= g_3^{-\beta} \cdot g_2^{\alpha} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r'_{T_i}} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{\frac{-\kappa}{T_i - T_{i^*}}} \\
&= g_3^{-\beta} \cdot g_2^{\alpha} \cdot (g_1^{T_i - T_{i^*}} g^{\gamma_2 + \gamma_1 \cdot T_{i^*}})^{r_{T_i}} \\
&= g_3^{-\beta} \cdot g_2^{\alpha} \cdot (v_1 \cdot v_2^{T_i})^{r_{T_i}},
\end{aligned}$$

and $\tau_{i,2} = g_2^{\frac{-1}{T_i - T_{i^*}}} g^{r'_{T_i}} = g^{r'_{T_i} - \frac{\kappa}{T_i - T_{i^*}}} = g^{r_{T_i}}$.

(c) Otherwise (i.e. $id^* \neq id$), $\mathcal{B}$ sets $\tau_{i,1} = g_3^{-\beta} \cdot (v_1 \cdot v_2^{T_i})^{r_{T_i}}$ and $\tau_{i,2} = g^{r_{T_i}}$, where $r_{T_i} \in_R \mathbb{Z}_q^*$.
Finally, $\mathcal{B}$ returns the token to $\mathcal{A}$ and adds the tuple (id, T_i, τ_i) to the list $List^\tau$.

3. Decryption key queries $\mathcal{O}_{ku}$: $\mathcal{A}$ issues a tuple (id, T_i) to $\mathcal{B}$, $\mathcal{B}$ constructs the updated key $sk_{id|i}$ by using $\mathcal{O}_{sk}$ and $\mathcal{O}_{tu}$. If $id^* = id$, $\mathcal{B}$ outputs $\perp$. Finally, $\mathcal{B}$ returns the updated key to $\mathcal{A}$ and adds the tuple $(id, T_i, sk_{id|i})$ to the list $List^{ku}$.
4. Re-encryption key queries $\mathcal{O}_{rk}$: $\mathcal{A}$ issues a tuple $(id, T_i, T_{i'})$ to $\mathcal{B}$, $\mathcal{B}$ constructs the re-encryption key as follows.
   (a) If $id^* \neq id$, $\mathcal{B}$ issues id to the private key extraction oracle $\mathcal{O}_{IBE}$ and receives $sk_{id} = (sk_{id_1}, sk_{id_2})$. $\mathcal{B}$ chooses $\xi \in_R \mathbb{G}_T$, $\rho \in_R \mathbb{Z}_q^*$, and then sets $rk_1 = sk_{id_1} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^\rho$, $rk_2 = sk_{id_2} \cdot g^\rho$ and $rk_3 = Enc(id,\ T_{i'},\ \xi)$. Finally, $\mathcal{B}$ returns the re-encryption key to $\mathcal{A}$ and adds the tuple $(id,\ T_i,\ T_{i'},\ rk_{id|i \to i'},\ 0,\ 1)$ to the list $List^{rk}$.
   (b) Otherwise (i.e. id^* is in a honest delegation chain including T_{i^*}), $\mathcal{B}$ sets $rk_1 = (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^{\hat{r}_1} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j \in \mathcal{V}_{id}} u_j)^\rho$, $rk_2 = g^{\hat{r}_1} \cdot g^\rho$ and $rk_3 = Enc(id, T_{i'}, \xi)$, where $\xi \in_R \mathbb{G}_T$, $\rho, \hat{r}_1 \in_R \mathbb{Z}_q^*$. Finally, $\mathcal{B}$ returns the re-encryption key to $\mathcal{A}$ and adds the tuple $(id,\ T_i,\ T_{i'},\ rk_{id|i \to i'},\ 1,\ 0)$ to the list $List^{rk}$.

Challenge. $\mathcal{A}$ outputs two messages m_0, m_1, a challenge identity id^* and a challenge time period T_{i^*} to $\mathcal{B}$. $\mathcal{B}$ then issues the tuple (id^*, m_0, m_1) to $\mathcal{C}_{IBE}$, and receives C_0^*, C_1^*, C_2^*. $\mathcal{B}$ next sets $C_3^* = (C_1^*)^{\gamma_2 + \gamma_1 \cdot T_{i^*}}$. Finally,

$\mathcal{B}$ returns the challenge ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*)$ under (id^*, T_{i^*}) to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continually makes queries as in Phase 1.

Guess. $\mathcal{A}$ outputs a bit b' to $\mathcal{B}$, $\mathcal{B}$ then forwards to $\mathcal{C}_{IBE}$.

This completes the simulations of the game. We next present the probability analysis as follows.

$$\begin{aligned}
&Adv_{\mathcal{A}}^{\text{Waters IBE}}(1^k)\\
&= |Pr[b' = b \wedge b = 1] - Pr[b' \neq b \wedge b = 0]|\\
&= |Pr[b' = b|b = 1] \cdot Pr[b = 1] - Pr[b' \neq b|b = 0] \cdot Pr[b = 0]|\\
&= |Pr[b' = b|b = 1] \cdot \frac{1}{2} - Pr[b' \neq b|b = 0] \cdot \frac{1}{2}|\\
&= |Pr[b' = b|b = 1] \cdot \frac{1}{2} - (1 - Pr[b' = b|b = 0]) \cdot \frac{1}{2}|\\
&= |Pr[b' = b] - \frac{1}{2}| \geq Adv_{CR\text{-}IB\text{-}PRE,\mathcal{A}}^{IND\text{-}CPA}(1^k, N),
\end{aligned}$$

where $Pr[b = 0] = Pr[b = 1] = \frac{1}{2}$. We let Adv_B^{TCR} be the advantage of $\mathcal{B}$ in breaking the security of TCR hash function (i.e. finding a collision in the TCR hash function). From the above we summarize that the success probability of $\mathcal{B}$ in breaking the underlying CPA-secure Waters IBE scheme is bounded as follows:

$$Adv_{\mathcal{A}}^{\text{Waters IBE}}(1^k) \geq Adv_{CR\text{-}IB\text{-}PRE,\mathcal{A}}^{IND\text{-}CPA}(1^k, N) - Adv_B^{TCR}.$$

It can be seen from our simulation that $\mathcal{B}$ has to guess the challenge time period with probability $\frac{1}{|\mathcal{T}|}$ such that our reduction loss in the proof is $|\mathcal{T}|$. Thus we have

$$Adv_{\mathcal{A}}^{\text{Waters IBE}}(1^k) \cdot \frac{1}{|\mathcal{T}|} \geq Adv_{CR\text{-}IB\text{-}PRE,\mathcal{A}}^{IND\text{-}CPA}(1^k, N) - Adv_B^{TCR}.$$

This completes the proof of Theorem 1. □

9.2 Proof for Theorem 2

Proof. Suppose there exists a PPT algorithm $\mathcal{A}$ who can break the collusion resistance of our system, we then construct a reduction algorithm $\mathcal{B}$ to break the CDH assumption by using $\mathcal{A}$. $\mathcal{B}$ is given the problem instance $(g, A = g^a, B = g^b)$.

Setup. $\mathcal{B}$ sets $g_1 = A$, $g_2 = B$, chooses $v_1, v_2, g_3 \in_R \mathbb{G}$, $\beta \in_R \mathbb{Z}_q^*$ and a TCR hash function TCR_1 as in the real scheme. $\mathcal{B}$ then choose the $(n+1)$

vector U so as to set $u_0 \prod_{j\in\mathcal{V}_{id}} u_j = g^{J(id)} \cdot B^{F(id)}$ for some integer-valued functions $F, J : \{0,1\}^n \to \mathbb{Z}$ chosen by $\mathcal{B}$ according to Waters' proof technique [7]. Finally, $\mathcal{B}$ outputs $mpk = (g, g_1, g_2, g_3, v_1, v_2, U, TCR_1)$ to $\mathcal{A}$. Note $\mathcal{B}$ does not know $g_2^{\alpha} = g^{ab}$ but it can compute g_3^{β}. In the view of $\mathcal{A}$, the distribution of pp is identical to that of the real scheme.

Queries. $\mathcal{A}$ issues the following queries to $\mathcal{B}$.

1. Private key queries $\mathcal{O}_{sk}(id)$: $\mathcal{A}$ issues id to $\mathcal{B}$, $\mathcal{B}$ responds the private key $sk_{id} = (sk_{id_1},\ sk_{id_2}) = (g_3^{\beta} \cdot A^{\frac{-J(id)}{F(id)}} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^r,\ A^{\frac{-1}{F(id)}} \cdot g^r)$, where $r \in_R \mathbb{Z}_q^*$.
2. Token update queries $\mathcal{O}_{tu}(id, T_i)$: $\mathcal{A}$ issues (id, T_i) to $\mathcal{B}$, $\mathcal{B}$ constructs the updated token for the time period T_i as $\tau_i = (\tau_{i,1}, \tau_{i,2}) = (g_3^{-\beta} \cdot (v_1 \cdot v_2^{T_i})^{r_{T_i}}, g^{r_{T_i}})$, where $r_{T_i} \in_R \mathbb{Z}_q^*$.
3. Decryption key queries $\mathcal{O}_{ku}$: $\mathcal{A}$ issues a tuple (id, T_i) to $\mathcal{B}$, $\mathcal{B}$ constructs the updated key $sk_{id|i}$ by using $\mathcal{O}_{sk}$ and $\mathcal{O}_{tu}$.
4. Re-encryption key queries $\mathcal{O}_{rk}$: $\mathcal{A}$ issues a tuple $(id, T_i, T_{i'})$ to $\mathcal{B}$, $\mathcal{B}$ constructs the re-encryption key as follows. $\mathcal{B}$ then chooses $\xi \in_R \mathbb{G}_T$, $\rho \in_R \mathbb{Z}_q^*$, and sets $rk_1 = A^{\frac{-J(id)}{F(id)}} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho'} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\hat{r}_1}$, $rk_2 = g^{\hat{r}_1} \cdot A^{\frac{-1}{F(id)}} \cdot g^{\rho'}$ and $rk_3 = Enc(id,\ T_{i'},\ \xi)$, where $\rho', \hat{r}_1 \in_R \mathbb{Z}_q^*$.

Output. $\mathcal{A}$ outputs the decryption key $sk_{id|i} = (sk_{id|i,1}, sk_{id|i,2}, sk_{id|i,3})$ for an honest identity id under time period T_i. If this is a valid key, then $\mathcal{B}$ can compute

$$\begin{aligned}\sigma_1 = \frac{sk_{id|i,1}}{rk_1} &= \frac{g_2^{\alpha} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\hat{r}_1} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho}}{g_2^{\alpha} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\hat{r}_1} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r}_2}} \\ &= \frac{(v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho}}{(v_1 \cdot v_2^{T_i})^{\hat{r}_2}},\end{aligned}$$

where $sk_{id|i} = (sk_{id|i,1}, sk_{id|i,2}, sk_{id|i,3})$, $rk_{id|i\to i'} = (rk_1, rk_2, rk_3)$ and suppose $sk_{id|i'}$ is corrupted by $\mathcal{A}$.

Since $\mathcal{B}$ knows ξ and $\hat{r}_2 = r_{T_i} + \tilde{r} + r_2$ (all chosen by itself in the similation of the game), $\mathcal{B}$ then computes

$$\begin{aligned}
\sigma_2 &= \frac{\sigma_1 \cdot (v_1 \cdot v_2^{T_{i'}})^{-TCR_1(\xi)}}{(v_1 \cdot v_2^{T_i})^{-\hat{r}_2}} \\
&= \frac{(v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho} \cdot (v_1 \cdot v_2^{T_{i'}})^{-TCR_1(\xi)}}{(v_1 \cdot v_2^{T_i})^{\hat{r}_2} \cdot (v_1 \cdot v_2^{T_i})^{-\hat{r}_2}} \\
&= (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho}.
\end{aligned}$$

Since $\rho = \rho' - a/F(id)$, then $\mathcal{B}$ has

$$\begin{aligned}
\sigma_3 &= (\sigma_2 \cdot (g^{J(id)} \cdot B^{F(id)})^{-\rho'} \cdot A^{J(id)/F(id)})^{-1} \\
&= ((g^{J(id)} \cdot B^{F(id)})^{\rho' - a/F(id)} \cdot (g^{J(id)} \cdot B^{F(id)})^{-\rho'} \cdot A^{J(id)/F(id)})^{-1} \\
&= B^a = g^{ab}.
\end{aligned}$$

Therefore $\mathcal{B}$ can solve the CDH problem.

This completes the proof of Theorem 2. □