A CCA-secure collusion-resistant Identity-based Proxy Re-encryption Scheme

Arinjita Paul[2],Varshika Srinivasavaradhan[1], S. Sharmila Deva Selvi[2] and C. Pandu Rangan[2]

[1] Thiagarajar College of Engineering, Madurai, India.
varshikavaradhan@gmail.com
[2] Theoretical Computer Science Lab,
Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai, India.
{arinjita,sharmila,prangan}@cse.iitm.ac.in

Abstract. Cloud storage enables its users to store confidential information as encrypted files in the cloud. A cloud user (say Alice) can share her encrypted files with another user (say Bob) by availing proxy re-encryption services of the cloud. Proxy Re-Encryption (PRE) is a cryptographic primitive that allows transformation of ciphertexts from Alice to Bob via a semi-trusted proxy, who should not learn anything about the shared message. Typically, the re-encryption rights are enabled only for a bounded, fixed time and malicious parties may want to decrypt or learn messages encrypted for Alice, even beyond that time. The basic security notion of PRE assumes the proxy (cloud) is semi-trusted, which is seemingly insufficient in practical applications. The proxy may want to collude with Bob to obtain the private keys of Alice for later use. Such an attack is called *collusion attack,* allowing colluders to illegally access all encrypted information of Alice in the cloud. Hence, achieving collusion resistance is indispensable to real-world scenarios. Realizing collusion-resistant PRE has been an interesting problem in the ID-based setting. To this end, several attempts have been made to construct a collusion-resistant IB-PRE scheme and we discuss their properties and weaknesses in this paper. We also present a new collusion-resistant IB-PRE scheme that meets the adaptive CCA security under the decisional bilinear Diffie-Hellman hardness assumption and its variant in the random oracle model.

Keywords: Identity-Based Proxy Re-Encryption, Collusion-resistance, Random Oracle, Unidirectional, CCA-secure.

1 Introduction

Cloud security is imperative in recent years owing to the popularity of cloud data storage and transmission. In order to preserve data privacy, users rely on standard encryption mechanisms that encrypt data using their public keys prior to

cloud storage. Enabling secure data sharing in the cloud calls for fast and secure re-encryption techniques for managing encrypted file systems. Blaze, Bleumer and Strauss [3] introduced the concept of Proxy Re-encryption (PRE) towards an efficient solution that offers delegation of decryption rights without compromising privacy. PRE allows a semi-trusted third party termed proxy to securely divert encrypted files of user A (delegator) to user B (delegatee) without revealing any information about the underlying files to the proxy. In the cloud scenario, the file owner shares a re-encryption key with the proxy (designated server in the cloud) who is assumed semi-trusted. PRE systems are classified into unidirectional and bidirectional schemes based on the direction of delegation. They are also classified into single-hop and multi-hop schemes based on the number of re-encryptions permitted. In this work, we focus on unidirectional and single-hop PRE schemes.

In a single-hop environment, a user A uses the cloud to store encrypted information and further sets the cloud as a proxy to allow re-encryption, thereby maintaining two kinds of encrypted data. The first kind called *first-level ciphertext* is the encrypted data that A would like to share with others. These kinds of data are subject to re-encryption and the cloud performs the conversion as a service upon getting the re-encryption key (re-key) from user A, re-encrypting towards user B. The second kind called *second-level ciphertext*, is the encrypted data re-encrypted towards A by a user C. Note that the second-level ciphertexts of A cannot be re-encrypted again with the re-key of A, as the PRE scheme is single hop. The re-key is created as a function of the private key of A, the public key of B and possibly some keys associated with the cloud itself. Hence, it is natural to ask if user B and the cloud can collude and acquire the private key of A. To motivate such a collusion, we observe the following two scenarios. Firstly, a malicious user B and a colluding cloud with a re-key may want to obtain the hidden messages in the second-level ciphertexts of A, which can be realized only using the private key of A. Again, the re-encryption rights are enabled for a bounded, fixed period and malicious parties may want to decrypt ciphertexts of A even beyond that period. Such an attack where a colluding cloud and a delegatee B obtains the private key of A is termed *collusion attack*. Preventing collusion attack is one of the major important problems in the context of cloud storage and computing. When the private key of A is obtained, the cloud and user B can cause total damage to user A in every possible way. Such a disclosure could be misused to the detriment of the delegator A such as unauthorized sharing of his confidential files, financial losses and identity theft. This marks collusion-resistance as a crucial property in proxy re-encryption; it achieves re-encryption by placing minimal trust on the proxy. Besides cloud storage, PRE can be applied to secure encrypted electronic mail forwarding, distributed system storage, outsourced filtering of encrypted spam, access control, DRM of apple iTunes among others [1,2,14].

Identity-based PRE was introduced by Green and Ateniese [8] as a solution to the certificate management problem in the PKI based PRE schemes. Of all the properties offered by identity-based proxy re-encryption (IB-PRE), collusion

resistance is the most desirable as it preserves the private key of the delegator even during an event of collusion between the proxy and delegatees. This would enable re-encryption in several real-time scenarios, such as secure sharing of files in a cloud with an untrusted server. In this paper, we study IB-PRE in the light of collusion resistance and propose a CCA-secure IB-PRE scheme that achieves the same based on Decisional Bilinear Diffie Hellman (DBDH) assumption and its variants in the random oracle model.

1.1 Related Works and Contribution

In ACNS 2007, Green and Ateniese [8] presented the first two constructions of IB-PRE, one being CPA-secure and the other being CCA-secure using bilinear pairing based on the Decisional Bilinear Diffie-Hellman assumption in the random oracle model. Their scheme is unidirectional, non-interactive, permits multiple re-encryptions but does not offer security against collusion attacks.

In this paper, we address the open problem proposed in [10] to design a non-interactive collusion-resistant IB-PRE scheme. Although several attempts have been made to achieve collusion-resistance in the identity based setting, all existing results are shown to either have some weaknesses or be insecure. In the collusion-resistant IB-PRE scheme given by Wang et al. [17] in the random oracle model, the re-keys are constructed using the master secret key which involves the PKG, making the scheme highly infeasible. Since the PKG is responsible for the generation of private keys, achieving delegation with the involvement of the PKG is trivial but undesirable. In [13], a generic construction for collusion resistant IB-PRE has been given based on threshold cryptosystem and key-management in IBE. However, their encryption algorithm involves splitting the private keys of the delegator into two components and publishing two public keys corresponding to the private keys. This is equivalent to the PKI setting, as the public keys require certification. Wang *et al.* [18] proposed a collusion-resistant IB-PRE scheme which is CPA-secure for the first-level ciphertext and CCA-secure for the second level ciphertext in the standard model based on the eDBDH assumption. In 2013, Han *et al.* [9] presented a CPA secure collusion-resistant IB-PRE scheme in the standard model based on the DBDH assumption. In 2015, Qiu et al [12] proposed a collusion-resistant IB-PRE scheme in the standard model. However, in 2016, Zhang et al. [19] showed that the scheme presented in [12] is vulnerable to collusion attacks. They also proposed a new identity-based proxy re-encryption scheme withstanding collusion attack and chosen ciphertext attack in the standard model. Note that both the collusion-resistant PRE schemes [9,19] make use of the information from the ciphertext components in the process of re-key generation. This clearly forces the user to create a separate delegation key for every ciphertext being translated. In the standard definition of PRE, a re-encryption key is generated only once between two parties (delegator A and delegatee B), irrespective of the number of ciphertexts being translated. But in [9,19], for every delegation between A and B, A needs to generate a new re-encryption key being delegated from A to B. This enforces the fact that user A needs to be online along with the proxy for converting every ciphertext towards

user B. In fact, this is equivalent to the *decrypt-and-then-encrypt* functionality. Note that making use of the knowledge of information from the ciphertext to generate re-keys is simple and trivial but makes the scheme highly impractical.

In our work, we address the open problem on collusion-resistance in the ID-based setting affirmatively adhering to the standard definition of PRE. Ever since the problem is proposed, it has remained as a challenging problem and no solution was emerging for several years. In the recent past, certain attempts have been made but most of them have either major drawbacks or serious flaws as discussed. A summary of the IB-PRE schemes have been provided in Table 1 in the context of collusion-resistance, alongside our scheme. Our collusion-resistant IB-PRE scheme is based on the IBE scheme due to Boneh and Franklin [4] and BLS short signature [5] and satisfies adaptive CCA security based on standard assumptions called the Decisional Diffie-Hellman assumption (DBDH) and its variant (m-DBDH). The proof of CCA security is considered in the random oracle model. The proof of collusion resistance of our scheme is based on modified Computational Diffie Hellman assumption(m-CDH).

Scheme	Security	Proof model	Delegation process involves	Underlying assumption	Remarks
Wang *et al.* [17]	CCA	RO	PKG, Delegator	DBDH	PKG involvement for collusion-resistance makes scheme infeasible.
Wang *et al.* [18]	CPA*	Standard	Delegator	eDBDH	Questionable or unproven claims.*
Han *et al.*[9]	CPA	Standard	Ciphertext Components and Delegator	DBDH	Re-key generation involves delegator and ciphertext components for collusion -resistance, standard PRE definition not satisfied.
Qiu *et al.*[12]	CCA	Standard	Delegator	DBDH	Collusion attack reported in [19]
Zhang *et al.*[19]	CCA	Standard	Ciphertext Components and Delegator	DBDH	Re-key generation involves delegator and ciphertext components for collusion -resistance, standard PRE definition not satisfied.
Our Scheme	CCA	RO	Delegator	DBDH	Collusion-resistant, adheres to standard PRE definition.

Table 1: A summary of IB-PRE schemes in the context of collusion-resistance. *The proof of security in [18] is questionable as simulating the challenge ciphertext solves the discrete log problem, discussed in details in Section 5.

2 Preliminaries

2.1 Bilinear Maps

A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ is a bilinear map if it satisfies the following conditions:

1. $\mathbb{G}_1$, $\mathbb{G}_T$ are of the same prime order q.

2. For all a, $b \epsilon \mathbb{Z}_q^*$, $g \epsilon \mathbb{G}_1$, $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$.
3. The map is non-degenerate, i.e., if $\mathbb{G}_1 = \langle g \rangle$, then $\mathbb{G}_T = \langle \hat{e}(g, g) \rangle$.
4. $\hat{e}$ is efficiently computable.

2.2 Hardness assumption

We state the computational hardness assumptions that we use to prove the security of our scheme. Let $\mathbb{G}_1$, $\mathbb{G}_\mathbb{T}$ be cyclic groups with prime order q and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be an admissible bilinear map.

m-Computational Diffie-Hellman (m-CDH) assumption [15] : The modified Computational Diffie-Hellman (m-CDH) assumption in $\mathbb{G}_1$ is, given a tuple of elements $(g, g^a, g^b, g^{\frac{1}{b}}, g^{\frac{a}{b}}) \in {\mathbb{G}_1}^5$, where $a,b \in_R \mathbb{Z}_q^*$, there exists no PPT adversary which can compute g^{ab} in $\mathbb{G}_1$, with a non-negligible advantage.

Decisional Bilinear Diffie-Hellman (DBDH) assumption : The Decisional Bilinear Diffie-Hellman (DBDH) assumption in $\mathbb{G}_1$,$\mathbb{G}_T$ is, given a tuple of elements $(g, g^a, g^b, g^c, T) \in {\mathbb{G}_1}^4 \times \mathbb{G}_T$, where $a,b,c \in_R \mathbb{Z}_q^*$, there exists no PPT adversary which can decide whether $T = \hat{e}(g, g)^{abc}$ or T is a random element in $\mathbb{G}_T$, with a non-negligible advantage.

m-Decisional Bilinear Diffie-Hellman (m-DBDH) assumption [16] : The modified-Decisional Bilinear Diffie-Hellman (m-DBDH) assumption in $\mathbb{G}_1$,$\mathbb{G}_T$ is, given a tuple of elements $(g, g^a, g^{\frac{1}{a}}, g^{\frac{1}{b}}, g^{\frac{a}{b}}, g^b, g^c, T) \in {\mathbb{G}_1}^7 \times \mathbb{G}_T$, where $a,b,c \in_R \mathbb{Z}_q^*$, there exists no PPT adversary which can decide whether $T = \hat{e}(g, g)^{abc}$ or T is a random element in $\mathbb{G}_T$, with a non-negligible advantage.

3 Definition and Security Model

3.1 Definition

In this section, we describe the syntactical definition of our single-hop unidirectional IB-PRE scheme. An IB-PRE scheme consists of the following algorithms.

- **Setup**(λ) : The PKG runs this probabilistic algorithm that takes a security parameter λ as input and outputs the public parameters $params$, which is shared with all the users, and the master secret key msk is kept private.
- **KeyGen**(msk, id_i, $params$) : This is a probabilistic algorithm run by the PKG which on input of a user identity $id_i \in \{0, 1\}^*$ and the master secret key msk, outputs the private key sk_{id_i} of the user identity id_i, which is securely communicated to the user.
- **ReKeyGen**(sk_{id_i}, id_j, $params$) : The delegator runs this probabilistic algorithm and takes as input its private key sk_{id_i} and the public key of the delegatee id_j to generate a re-encryption key $RK_{i \rightarrow j}$ from id_i to id_j. The delegator then sends the re-encryption key to the proxy via a secure channel.

- **Encrypt**(m, id_i, $params$) : The sender runs the encryption algorithm which takes as input a message $m \in \mathcal{M}$, and an identity id_i under which m is encrypted. It outputs the ciphertext $\mathbb{C}$, which is termed as first-level ciphertext.
- **Decrypt**($\mathbb{C}$, sk_{id_i}, $params$) : The decryption algorithm is a deterministic algorithm run by the delegator. On input of a first-level ciphertext $\mathbb{C}$ and the delegator's private key sk_{id_i}, the algorithm outputs message $m \in \mathcal{M}$ or the error message *"INVALID CIPHERTEXT"*.
- **Re-Encrypt**($\mathbb{C}$, $RK_{i \to j}$, $params$) : This is a probabilistic algorithm run by the proxy which takes in as input the first-level ciphertext $\mathbb{C}$ and the re-encryption key $RK_{i \to j}$ and outputs the re-encrypted ciphertext $\mathbb{D}$, termed as second-level ciphertext.
- **Re-Decrypt**($\mathbb{D}$, sk_{id_j}, $params$) : This is a deterministic algorithm run by the delegatee. On input of the second-level ciphertext $\mathbb{D}$ and the delegatee's private key sk_{id_j}, the algorithm outputs the original message $m \in \mathcal{M}$ or the error message *"INVALID CIPHERTEXT"*.

The consistency of an IB-PRE scheme for any given public parameters $params$ and a key pair (id_i, sk_{id_i}),(id_j, sk_{id_j}) is defined as follows:

1. Consistency between encryption and decryption:

$$Decrypt(Encrypt(m, id_i, params), sk_{id_i}, params) = m, \forall m \in \mathcal{M}.$$

2. Consistency between re-encryption and re-decryption:

$$Re - Decrypt(\mathbb{D}, sk_{id_j}, params) = m, \forall m \in \mathcal{M},$$

where $\mathbb{D} \leftarrow Re\text{-}Encrypt(\mathbb{C}, \mathbb{RK}_{i \to j}, params)$ and $\mathbb{C} \leftarrow Encrypt(m, id_i, params)$.

3.2 Security Model

In this subsection, we define the security notions of our IB-PRE scheme. In IB-PRE, there are two levels of ciphertexts, the first-level and the second-level ciphertext, and it is crucial to prove the security for both levels [11]. We consider the CK model wherein the adversary $\mathcal{A}$ can adaptively choose public keys for malicious users. $\mathcal{A}$ adaptively queries the oracles listed below, and the challenger $\mathcal{C}$ responds to the queries and simulates an environment running IB-PRE for $\mathcal{A}$.

- **Private Key Extraction Oracle**($\mathcal{O}_{KE}(id_i)$): Given as input an identity id_i, return the corresponding private key sk_{id_i}.
- **Re-Key Generation Oracle**($\mathcal{O}_{RK}(id_i, id_j)$): Given as input ($id_i$,$id_j$), return the re-encryption key $RK_{i \to j}$.
- **Re-Encryption Oracle**($\mathcal{O}_{RE}(id_i, id_j, \mathbb{C})$): Given as inputs two identities (id_i, id_j) and a first-level ciphertext $\mathbb{C}$, return the second level ciphertext $\mathbb{D}$.
- **Decryption Oracle**($\mathcal{O}_{DEC}(id_i, \mathbb{C})$): Given as input an identity id_i and a first level ciphertext $\mathbb{C}$ encrypted under id_i, return the message m or *"INVALID CIPHERTEXT"* if the ciphertext is invalid.
- **Re-Decryption Oracle**($\mathcal{O}_{REDEC}(id_j, \mathbb{D})$): Given as input an identity id_j and a second level ciphertext $\mathbb{D}$ re-encrypted under id_j, return the message m or *"INVALID CIPHERTEXT"* if the ciphertext is invalid.

First Level Ciphertext Security: In the first level ciphertext security, an adversary $\mathcal{A}$ is challenged with a first level ciphertext $\mathbb{C}$ encrypted under the target identity id^*. Following is the description of the game template for Chosen Ciphertext Security:

1. **Setup:** The challenger $\mathcal{C}$ takes a security parameter λ and executes the *Setup* algorithm to get the system parameters *params* and returns it to $\mathcal{A}$.
2. **Phase 1:** $\mathcal{A}$ adaptively queries the Private Key Extraction, Re-Key Generation, Re-Encryption, Decryption and Re-Decryption oracles and $\mathcal{C}$ responds to the queries.
3. **Challenge:** When $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length plaintexts m_0,$m_1 \in \mathcal{M}$ and a target identity id^* with the following adversarial constraints:
   - The private key of the target identity id^* must not be queried previously.
   - $\mathcal{A}$ must not have queried $\mathcal{O}_{RK}(id^*, d_j)$, such that the private key of id_j is already queried upon.

   On receiving $\{m_0, m_1\}$, $\mathcal{C}$ obtains a random bit $\psi \in \{0, 1\}$ and computes a challenge ciphertext $\mathbb{C}^* = Encrypt(id^*, m_\psi, params)$ and returns $\mathbb{C}^*$ to $\mathcal{A}$.
4. **Phase 2:** $\mathcal{A}$ issues queries as in Phase 1 with the following constraints.
   - The ReKey Generation query $\mathcal{O}_{RK}(id,id_j)$ is only allowed if the private key of id_j has not been queried previously.
   - If $\mathcal{A}$ issues a Re-Encryption query $\mathcal{O}_{RE}(id_i,id_j,\mathbb{C})$ such that the private key of id_j has been queried upon, $(id_i,\mathbb{C})$ cannot be a challenge derivative (defined next) of $(id^*,\mathbb{C}^*)$.
   - $\mathcal{A}$ can issue a Decryption query $\mathcal{O}_{DEC}(id_i,\mathbb{C})$ or a Re-Decryption query $\mathcal{O}_{REDEC}(id_j,\mathbb{D})$ only if $(id_i,\mathbb{C})$ or $(id_j,\mathbb{D})$ respectively is not a *derivative* of $(id^*,\mathbb{C}^*)$.

   Definition 1. (Challenge Derivative). *The challenge derivatives of* $(id_i, \mathbb{C})$ *in the CCA setting as adopted from [7] are as shown below:*
   - *Reflexitivity:* $(id_i, \mathbb{C})$ *is a challenge derivative of its own.*
   - *Derivative by re-encryption: if* $\mathbb{D} \leftarrow \mathcal{O}_{RE}(id_i, id_j, \mathbb{C})$*, then* $(id_j, \mathbb{D})$ *is a challenge derivative of* $(id_i, \mathbb{C})$*.*
   - *Derivative by re-key: if* $\mathbb{D} \leftarrow Re\text{-}Encrypt(\mathbb{C}, RK_{i\rightarrow j}, params)$*, where the re-key* $RK_{i\rightarrow j} \leftarrow \mathcal{O}_{RK}(ID_i, ID_j)$*,* $(id_j, \mathbb{D})$ *is a challenge derivative of* $(id_i, \mathbb{C})$*.*
5. **Guess:** Finally, $\mathcal{A}$ outputs a guess $\psi' \in \{0, 1\}$.

The advantage of the adversary $\mathcal{A}$ in winning the game is defined as:

$$Adv_{A,first}^{IND-IBPRE-CCA} = 2|Pr[\psi' = \psi] - \tfrac{1}{2}|$$

where the probability is taken over the coin tosses of the challenger $\mathcal{C}$ and adversary $\mathcal{A}$. The scheme is $IND\text{-}IBPRE\text{-}CCA$ secure for the first level ciphertext against any t-time adversary $\mathcal{A}$ making q_{KE} queries to key extraction oracle, q_{RK} queries to the re-key generation oracle, q_{RE} queries to re-encryption oracle, q_{DEC} queries to decryption oracle, q_{REDEC} queries to re-decryption oracle, if the advantage of $\mathcal{A}$ is: $Adv_{A,first}^{IND-IBPRE-CCA} \leq \epsilon$.

Second Level Ciphertext Security: In the second-level ciphertext security, the adversary $\mathcal{A}$ is challenged with a second level ciphertext $\mathbb{D}^*$ which is a re-encryption of the ciphertext $\mathbb{C}$ under the delegator identity id_i towards target identity id^*, re-encrypted using the re-key $RK_{i\to *}$. $\mathcal{A}$ does not have access to the corresponding first level ciphertext $\mathbb{C}$. The security for the second level ciphertext is unaffected whether the delegator identity id_i is a corrupt user or not. Note that, since a second-level ciphertext cannot be further re-encrypted, $\mathcal{A}$ is allowed to obtain all the re-encryption keys in our security model. This also justifies the removal of the re-encryption oracle from the security model. Following is the description of the game template for Chosen Ciphertext Security:

1. **Setup :** The challenger $\mathcal{C}$ takes a security parameter λ and executes the *Setup* algorithm to get the system parameters *params* and return it to $\mathcal{A}$.
2. **Phase 1:** $\mathcal{A}$ adaptively queries to Private Key Extraction, Re-Key Generation, Decryption and Re-Decryption oracles and the challenger $\mathcal{C}$ responds to the queries. Note that since the challenger $\mathcal{C}$ provides $\mathcal{A}$ with all possible re-encryption keys, $\mathcal{C}$ need not simulate the re-encryption oracle.
3. **Challenge:** When $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal-length plaintexts m_0,$m_1 \in \mathcal{M}$, a delegator identity id_i and an honest target delegatee identity id^* with the following adversarial constraints:
   - The adversary $\mathcal{A}$ must not have queried $\mathcal{O}_{KE}(id^*)$ at any point in time.
   - The $\mathcal{A}$ must not have queried $\mathcal{O}_{RK}(id_i, id^*)$.
   - $\mathcal{A}$ cannot choose id_i as the delegator if it has already obtained $RK_{i\to *}$.

   On receiving $\{m_0, m_1\}$, $\mathcal{C}$ obtains a random bit $\psi \in \{0, 1\}$ and computes a challenge ciphertext $\mathbb{D}^* = \textit{Re-Encrypt}(\textit{Encrypt}(m_\psi, id_i, params), RK_{i\to *}, params)$ and returns $\mathbb{D}^*$ to $\mathcal{A}$.
4. **Phase 2:** $\mathcal{A}$ issues queries as in Phase 1 with the following constraints:
   - $\mathcal{A}$ cannot issue a Re-Decryption query $\mathcal{O}_{REDEC}(id^*,\mathbb{D}^*)$.
   - If sk_{id_i} has been queried previously, $\mathcal{A}$ cannot query $RK_{i\to *}$ to $\mathcal{C}$.
5. **Guess:** Finally, $\mathcal{A}$ outputs a guess $\psi' \in \{0, 1\}$.

The advantage of the adversary $\mathcal{A}$ in winning the game is defined as:

$$Adv_{A,second}^{IND-IBPRE-CCA} = 2|Pr[\psi' = \psi] - \tfrac{1}{2}|$$

where the probability is over the coin tosses of the challenger $\mathcal{C}$ and adversary $\mathcal{A}$. The scheme is $IND\text{-}IBPRE\text{-}CCA$ secure for the second level ciphertext against a t-time adversary $\mathcal{A}$ making q_{KE} queries to key extraction oracle, q_{RK} queries to the re-key generation oracle, q_{DEC} queries to decryption oracle, q_{REDEC} queries to re-decryption oracle, if the advantage of $\mathcal{A}$ is: $Adv_{A,second}^{IND-IBPRE-CCA} \leq \epsilon$.

Collusion Resistance: Collusion-resistance or delegator secret key (DSK) security prevents a colluding proxy and delegatee to recover the delegator's private key in full [7]. Following is the game template of the security model for collusion resistance as in [6].

- **Setup**: $\mathcal{C}$ takes as input the security parameter λ and runs the *Setup* algorithm to generate and return the system parameters *params* to $\mathcal{A}$.

– **Queries**: $\mathcal{A}$ issues the following queries adaptively to $\mathcal{C}$:
  - Private-Key Extraction Oracle $\mathcal{O}_{KE}(id_i)$: $\mathcal{C}$ runs the KeyGen(msk, id_i, $params$) algorithm to generate the private key sk_{id_i} of identity id_i and returns sk_{id_i} to $\mathcal{A}$.
  - Re-encryption Key Generation Oracle $\mathcal{O}_{RK}(id_i, id_j)$: $\mathcal{C}$ generates and returns the re-encryption key $RK_{i\rightarrow j}$ from identity id_i to id_j.
– **Output**: $\mathcal{A}$ returns sk_{i^*} as the private key of an identity id_i^*. $\mathcal{A}$ wins the game if sk_i^* is a valid private key of an identity id_i^* whose private key has not been queried for.

The advantage of $\mathcal{A}$ in attacking the collusion-resistance or delegator secret security of the scheme is defined as $Adv_{\mathcal{A}}^{DSK} = Pr[\mathcal{A}\, wins]$, where the probability is over the random coin tosses of the challenger $\mathcal{C}$ and adversary $\mathcal{A}$. A scheme is defined as (t, ϵ)-DSK secure against a t-time adversary $\mathcal{A}$ making at most q_{RK} re-encryption key generation queries if the advantage of $\mathcal{A}$ is: $Adv_{\mathcal{A}}^{DSK} \leq \epsilon$.

4 Our Proposed Collusion Resistant IB-PRE Scheme

4.1 Overview of Construction

The starting points of our contruction are the the IB-PRE scheme of Green and Ateniese [8] which is based on Boneh and Franklin's IBE scheme [4] and BLS short signature [5]. In the system by Green and Ateniese, the PKG chooses a random element $s \leftarrow \mathbb{Z}_q^*$ as the master secret key and sets the private key for an identity id_i as $H_1(id_i)^s$. Note that the hash functions are used as defined in the IB-PRE scheme in [8]. The original ciphertext of a message $m \in \{0,1\}^n$ is computed by choosing a random element $\sigma \leftarrow \mathbb{G}_T$ and computing $r = H_4(\sigma, m)$. Then the ciphertext components $\mathbb{C}_1 = g^r$, $\mathbb{C}_2 = \sigma \cdot \hat{e}(g^s, H_1(id_i)^r)$ and $\mathbb{C}_3 = m \oplus H_5(\sigma)$ are computed. $S = H_3(id_i||\mathbb{C}_1||\mathbb{C}_2||\mathbb{C}_3)^r$ is computed as a BLS signature used during re-encryption/decryption to confirm well-formedness of the ciphertext. Finally $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, S)$ is output as the ciphertext of message m. The re-key is generated by picking a random element $N \leftarrow \{0,1\}^n$ and computing $K = \hat{e}(sk_{id_i}, H_1(id_j))$. The re-key $RK_{i\rightarrow j} = \langle RK_{i\rightarrow j}^1, RK_{i\rightarrow j}^2 \rangle = \langle N, H_2(K, id_i, id_j, N) \cdot sk_{id_i} \rangle$ is computed. Now, if the proxy and delegatee collude, $K = \hat{e}(H_1(id_i), sk_{id_j})$ is computed and the private key of the delegator can be recovered by computing $sk_{id_i} = \frac{RK_{i\rightarrow j}^2}{H_2(K, id_i, id_j, N)}$.

In order to extend the system proposed by Green and Ateniese to the collusion-resistant setting, we introduce another generator h and two group elements $g_1 = g^\delta$, $h_1 = h^\delta$ to the public parameters. In our attempt, the PKG chooses $s \leftarrow \mathbb{Z}_q^*$ as the master secret key. The re-key is generated by picking $s_1, s_2 \leftarrow \mathbb{Z}_q^*$ and computing $x_{ij} = H_5(\hat{e}(P_{pub_1}, H_1(id_j)_1^s), id_i, id_j)$, where $x_{ij} \in \mathbb{Z}_q^*$ and $P_{pub_1} = g_1^s$. The re-key is computed as $RK_{i\rightarrow j} = \langle \mathrm{RK}_{i\rightarrow j}^1, \mathrm{RK}_{i\rightarrow j}^2, \mathrm{RK}_{i\rightarrow j}^3 \rangle = \langle (sk_{id_i})^{-1} \cdot h^{x_{ij}\, s_2}, h_1{}^{s_2}, g^{s_1} \rangle$. We construct our re-key in such a way that the private key of the delegator ($sk_{id_i} \in \mathbb{G}_1$) is blinded with a random salt and can only

be removed in the target group G_T during decryption of the re-encrypted ciphertexts. The private key sk_{id_i} of delegator can be retrieved from the re-key component $\mathrm{RK}^1_{i\to j}$ only by users with the knowledge of both sk_{id_j} (the delegatee's secret key) and random element s_2 (chosen by delegator). This clearly makes it infeasible to retrieve the private key sk_{id_i} in $\mathbb{G}_1$ from the re-key, which prevents the colluders to recover the delegator's private key and provides collusion-resistance.

4.2 Construction

In this sub-section, we present the construction of our collusion-resistant IB-PRE scheme followed by its correctness and security proof. Our IB-PRE scheme consists of the following algorithms:

1. **Setup**(λ) : The PKG takes the security parameter λ as input. Let $\mathbb{G}_1$, $\mathbb{G}_T$ be groups of prime order q and let g, h be the generators of $\mathbb{G}_1$. The PKG picks $\delta \leftarrow \mathbb{Z}_q^*$ and computes $g_1 = g^\delta$ and $h_1 = h^\delta$. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be an admissible bilinear map.
   Five cryptographic hash functions are chosen by PKG as below, which are modelled as random oracles in our security proof:

$$H_1 : \{0,1\}^* \to \mathbb{G}_1$$
$$H_2 : \mathbb{G}_T \times \{0,1\}^n \to \mathbb{Z}_q^*$$
$$H_3 : \mathbb{G}_T \to \{0,1\}^n$$
$$H_4 : \{0,1\}^* \to \mathbb{G}_1$$
$$H_5 : \mathbb{G}_T \times \{0,1\}^* \to \mathbb{Z}_q^*$$

   The PKG selects $s \xleftarrow{\$} \mathbb{Z}_q^*$, computes $P_{pub_1} = g^s$, $P_{pub_2} = g_1^s$ and the master secret key $msk = s$. The message space $\mathcal{M}$ is $\{0,1\}^n$. PKG returns the public parameters $params = (\mathbb{G}_1, \mathbb{G}_T, g, h, g_1, h_1, P_{pub_1}, P_{pub_2}, \hat{e}, H_1, H_2, H_3, H_4, H_5, n)$.

2. **KeyGen**(msk, id_i, $params$) : For each user with identity $id_i \in \{0,1\}^*$, the PKG computes the private key $sk_{id_i} = H_1(id_i)^s$ and sends sk_{id_i} to user id_i in a secure way.

3. **ReKeyGen**($sk_{id_i}, id_j, params$) : The user with identity id_i generates a re-encryption key from id_i to user id_j as below:
   - Select s_1, $s_2 \xleftarrow{\$} \mathbb{Z}_q^*$.
   - Compute $x_{ij} = H_5(\hat{e}(P_{pub_1}, H_1(id_j)^{s_1}), id_i, id_j) \in \mathbb{Z}_q^*$.
   - Compute $\mathrm{RK}^1_{i\to j} = (sk_{id_i})^{-1} \cdot h^{x_{ij}\, s_2} = H_1(id_i)^{-s} \cdot h^{x_{ij}\, s_2} \in \mathbb{G}_1$.
   - Compute $\mathrm{RK}^2_{i\to j} = {h_1}^{s_2} \in \mathbb{G}_1$.
   - Compute $\mathrm{RK}^3_{i\to j} = g^{s_1} \in \mathbb{G}_1$.

   The delegator id_i sends the re-encryption key $RK_{i\to j} = (RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j})$ to the proxy via a secure channel.

4. **Encrypt**(m,id_i, $params$) : To encrypt the message $m \epsilon \mathcal{M}$ for the user with identity id_i, the sender chooses $\sigma \xleftarrow{\$} \mathbb{G}_T$ and computes $r = H_2(\sigma, m)$. The sender then computes the ciphertext $\mathbb{C}$ as below:
   - Compute $\mathbb{C}_1 = g^r \in \mathbb{G}_1$.
   - Compute $\mathbb{C}_2 = g_1^r \in \mathbb{G}_1$.
   - Compute $\mathbb{C}_3 = \sigma \cdot \hat{e}(P_{pub_2}, H_1(id_i)^r) \in \mathbb{G}_T$.
   - Compute $\mathbb{C}_4 = m \oplus H_3(\sigma) \in \{0,1\}^n$.
   - Compute $\mathbb{C}_5 = H_4(id_i \| \mathbb{C}_1 \| \mathbb{C}_2 \| \mathbb{C}_3 \| \mathbb{C}_4)^r \in \mathbb{G}_1$.

   The sender returns the first-level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$.

5. **Decrypt**($\mathbb{C}$, sk_{id_i}, $params$) : With input a first level ciphertext $\mathbb{C} = \langle \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5 \rangle$, user id_i with his private key sk_{id_i} decrypts as below:
   - Check if both the following conditions hold:

$$\hat{e}(\mathbb{C}_1, g_1) \stackrel{?}{=} \hat{e}(g, \mathbb{C}_2) \tag{1}$$

$$\hat{e}(\mathbb{C}_1, H_4(id_i \| \mathbb{C}_1 \| \mathbb{C}_2 \| \mathbb{C}_3 \| \mathbb{C}_4)) \stackrel{?}{=} \hat{e}(g, \mathbb{C}_5) \tag{2}$$

     If either of the conditions fail, it returns *"INVALID CIPHERTEXT"*.
   - Otherwise, compute σ as below:

$$\sigma = \frac{\mathbb{C}_3}{\hat{e}(\mathbb{C}_2, sk_{id_i})}. \tag{3}$$

   - Compute the message:

$$m = \mathbb{C}_4 \oplus H_3(\sigma), \tag{4}$$

   - Check the following condition:

$$\mathbb{C}_2 \stackrel{?}{=} g_1{}^{H_2(\sigma, m)}. \tag{5}$$

     If satisfied, it outputs m, else outputs *"INVALID CIPHERTEXT"*.

6. **Re-Encrypt**($\mathbb{C}$, $\mathbb{RK}_{i \to j}$, $params$) : The proxy re-encrypts the first-level ciphertext $\mathbb{C}$ to second-level ciphertext $\mathbb{D}$ as below.
   - Check if the following condition holds:

$$\hat{e}(\mathbb{C}_1, H_4(id_i || \mathbb{C}_1 || \mathbb{C}_2 || \mathbb{C}_3 || \mathbb{C}_4)) \stackrel{?}{=} \hat{e}(g, \mathbb{C}_5). \tag{6}$$

   - If the check fails, return *"INVALID CIPHERTEXT"*.
   - Set $\mathbb{D}_1 = \mathbb{C}_1 = g^r \in \mathbb{G}_1$
   - Set $\mathbb{D}_2 = \mathrm{RK}^3_{i \to j} = g^{s_1} \in \mathbb{G}_1$,
   - Compute $\mathbb{D}_3 = \mathbb{C}_3 . \hat{e}(\mathbb{C}_2, \mathrm{RK}^1_{i \to j}) = \sigma . \hat{e}(g_1^r, h^{x_{ij} s_2}) \in \mathbb{G}_T$,
   - Set $\mathbb{D}_4 = \mathbb{C}_4 = m \oplus H_3(\sigma) \in \{0,1\}^n$,
   - Set $\mathbb{D}_5 = \mathrm{RK}^2_{i \to j} = h_1^{s_2} \in \mathbb{G}_1$.

   The proxy returns the second-level ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3, \mathbb{D}_4, \mathbb{D}_5)$.

7. **Re-Decrypt**($\mathbb{D}$,sk_{id_j}, $params$) : With input a second-level ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3, \mathbb{D}_4, \mathbb{D}_5)$, the delegatee id_j performs the following computations:
   - Compute $x_{ij} = H_5(\hat{e}(D_2, sk_{id_j}), id_i, id_j)$ using private key sk_{id_j}.
   - Compute σ as below:

$$\sigma = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_1, \mathbb{D}_5)^{x_{ij}}}. \tag{7}$$

   - Compute the message m as:

$$m = \mathbb{D}_4 \oplus H_3(\sigma). \tag{8}$$

   - Check if the following condition holds:

$$\mathbb{D}_1 \stackrel{?}{=} g^{H_2(\sigma, m)}. \tag{9}$$

     If the check holds, it returns m else returns *"INVALID CIPHERTEXT"*.

4.3 Correctness

Our collusion resistant unidirectional IB-PRE scheme is consistent and correct, which can be verified using the following computations:

- Correctness of first-level ciphertext verification from equation 6:

$$\begin{aligned} RHS &= \hat{e}(\mathbb{C}_1, H_4(id_i||\mathbb{C}_1||\mathbb{C}_2||\mathbb{C}_3||\mathbb{C}_4)) \\ &= \hat{e}(g^r, H_4(id_i||\mathbb{C}_1||\mathbb{C}_2||\mathbb{C}_3||\mathbb{C}_4)) \\ &= \hat{e}(g, \mathbb{C}_5) \\ &= LHS. \end{aligned}$$

- Consistency between encryption and decryption from equation 3:

$$\begin{aligned} RHS &= \frac{\mathbb{C}_3}{\hat{e}(\mathbb{C}_2, sk_{id_i})} \\ &= \frac{\sigma \cdot \hat{e}(P_{pub_2}, H_1(id_i)^r)}{\hat{e}(\mathbb{C}_2, sk_{id_i})} \\ &= \frac{\sigma \cdot \hat{e}({g_1}^s, H_1(id_i)^r)}{\hat{e}({g_1}^r, H_1(id_i)^s)} \\ &= \sigma \\ &= LHS. \end{aligned}$$

  Using σ in equation 4, we get:

$$\begin{aligned} RHS &= \mathbb{C}_4 \oplus H_3(\sigma) \\ &= m \oplus H_3(\sigma) \oplus H_3(\sigma) \\ &= m \\ &= LHS. \end{aligned}$$

– Consistency between re-encryption and re-decryption from equation 7:

$$\begin{aligned} RHS &= \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_1, \mathbb{D}_5)^{x_{ij}}} \\ &= \frac{\mathbb{C}_3.\hat{e}(\mathbb{C}_1, \mathrm{RK}^1_{i\to j})}{\hat{e}(\mathbb{C}_1, \mathrm{RK}^2_{i\to j})} \\ &= \frac{\sigma.\hat{e}(g_1^r, h^{x_{ij}\, s_2})}{\hat{e}(g^r, h_1^{s_2})^{x_{ij}}} \\ &= \sigma \\ &= LHS. \end{aligned}$$

Using σ in equation 8, we get:

$$\begin{aligned} RHS &= \mathbb{D}_4 \oplus H_3(\sigma) \\ &= m \oplus H_3(\sigma) \oplus H_3(\sigma) \\ &= m \\ &= LHS. \end{aligned}$$

Remark 1. Our IB-PRE scheme defines the two levels of ciphertexts as follows. The first level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ is generated by the **Encrypt** algorithm, which includes the delegatable ciphertexts encrypted towards the delegator. The second level ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3, \mathbb{D}_4, \mathbb{D}_5)$ is generated by the **Re-Encrypt** algorithm, comprising the re-encrypted ciphertexts that cannot be delegated further in encrypted form. When a user stores her data encrypted in the cloud, the data may belong to any of the two categories.

4.4 Security Proof

Collusion Resistance:

Theorem 1. *Our proposed scheme is DSK-secure under the m-CDH assumption. If a DSK adversary $\mathcal{A}$ breaks the DSK security of the given scheme with an advantage ϵ in time t, then there exists a challenger $\mathcal{C}$ who solves the m-CDH problem with advantage ϵ' within time t' where:*

$$\epsilon' \geq \frac{\epsilon}{e(1 + q_{KE})},$$

$$t^* \leq t + O(q_{H_1} + 3q_{RK} + q_{KE})t_{et} + O(q_{RK})t_{bp},$$

where e is the base of natural logarithm, t_{et} denotes the time taken for exponentiation in group $\mathbb{G}_1$ and t_{bp} is the time taken for one bilinear pairing operation.

Proof. Let $\mathcal{A}$ be a p.p.t adversary that has a non-negligible advantage ϵ in breaking the $(t, \epsilon)DSK$ security of the scheme with access to the random oracle H_1. Then, we can construct a polynomial time algorithm $\mathcal{C}$ to solve the m-CDH assumption in $\mathcal{G}_1$ with a non-negligible advantage. Note that the hash functions $H_2,\ H_3,\ H_4$ and H_5 are not modeled as random oracles in the proof. Algorithm $\mathcal{C}$ accepts as input a properly-distributed tuple $\langle \mathbb{G}_1 = \langle g \rangle, g^a, g^b, g^{\frac{1}{b}}, g^{\frac{a}{b}} \rangle$ and outputs the value of g^{ab}. $\mathcal{C}$ plays the DSK game with $\mathcal{A}$ in the following way:

- **Setup**: $\mathcal{C}$ implicitly defines the master secret key $msk = a$ and $\delta = \frac{1}{b}$. It sets $P_{pub_1} = g^a$, $g_1 = g^{\frac{1}{b}}$ and $P_{pub_2} = g^{\frac{a}{b}}$. It picks $\nu \leftarrow \mathbb{Z}_q^*$, computes $h = (g^a)^\nu$ and $h_1 = (g^{\frac{a}{b}})^\nu$ and returns the resulting system parameters $params$ to $\mathcal{A}$.

- **Queries**: $\mathcal{C}$ interacts with $\mathcal{A}$ in the following ways:
  - $H_1(id_i)$ Oracle: $\mathcal{C}$ responds to the queries of $\mathcal{A}$ by maintaining a list L_{H_1} with tuples of the form $\langle id_i \in \{0,1\}^*,\ y_i \in \mathbb{Z}_q^*,\ k_i \in \mathbb{Z}_q^*, \alpha_i \in \{0,1\} \rangle$. If the tuple $\langle id_i, y_i, k_i, \alpha_i \rangle$ already exists in L_{H_1}, retrieve and return the value y_i. Else randomly set $\alpha_i \in \{0,1\}$ such that $Pr[\alpha_i = 0] = \gamma$ which is defined as in the first level ciphertext security. Set the hash value according to the following cases:
    * If $\alpha_i = 0$, select $z_i \leftarrow \mathbb{Z}_q^*$, compute $y_i = g^{z_i}$ and set $H_1(id_i) = y_i$.
    * If $\alpha_i = 1$, select $z_i \leftarrow \mathbb{Z}_q^*$, compute $y_i = (g^b)^{z_i}$. Set $H_1(id_i) = y_i$.

    Store tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ in list L_{H_1} and return y_i.

  - Private Key Extraction Oracle $\mathcal{O}_{KE}(id_i)$: $\mathcal{C}$ responds to the key-extraction query of an identity id_i by first checking for a tuple $\langle id_i, y_i, k_i, \alpha_i \rangle$ already exists in L_{H_1}. If $\alpha_i = 1$, abort and return $failure$. Otherwise, return $sk_{id_i} = (g^a)^{z_i}$ as the private key of identity id_i.

  - Re-encryption Key Generation Oracle $\mathcal{O}_{RK}(id_i, id_j)$: $\mathcal{C}$ maintains a list L_{RK} that contains tuples of the form $\langle id_i \in \{0,1\}^*, id_j \in \{0,1\}^*, x_{ij} \in \mathbb{Z}_q^*, RK_{i\to j}^1 \in \mathbb{G}_1, RK_{i\to j}^2 \in \mathbb{G}_1, RK_{i\to j}^3 \in \mathbb{G}_1, s_1 \in \mathbb{Z}_q^*, \bar{s}_2 \in \mathbb{Z}_q^* \rangle$. $\mathcal{C}$ responds to the re-encryption key-generation queries of $\mathcal{A}$ from user id_i to id_j by searching list L_{H_1} for tuples corresponding to id_i and id_j respectively and computing the re-keys as per the following cases:
    * Check if the re-key $RK_{i\to j}$ exists in L_{RK} by searching for a tuple $\langle id_i, id_j, x_{ij}, RK_{i\to j}^1, RK_{i\to j}^2, RK_{i\to j}^3, s_1, \bar{s}_2 \rangle$. If present, return $RK_{i\to j}$.
    * If $\alpha_i = 0 \ \wedge \ \alpha_j = 0$: Generate the re-keys as per **ReKeyGen** protocol.
    * If $\alpha_i = 0 \ \wedge \ \alpha_j = 1$: Generate the re-keys as per **ReKeyGen** protocol.
    * If $\alpha_i = 1 \ \wedge \ \alpha_j = 0$: Pick $x_{ij}, s_1, \bar{s}_2 \leftarrow \mathbb{Z}_q^*$. Update list L_{H_5} with the tuple $\langle \hat{e}(g^a, g^{z_j s_1}), id_i, id_j, x_{ij} \rangle$. Implicitly define $s_2 = z_i x_{ij}^{-1} \nu^{-1} b + \bar{s}_2$ and compute the re-key $RK_{i\to j}$ as follows:
      · Compute $RK_{i\to j}^1 = (g^a)^{\nu x_{ij} \bar{s}_2}$.
      · Compute $RK_{i\to j}^2 = (g^a)^{x_{ij}^{-1} z_i} \cdot (g^{\frac{a}{b}})^{\nu \bar{s}_2}$.

· Compute $RK^3_{i\to j} = g^{s_1}$.

Observe that the re-encryption key $RK_{i\to j}$ computed is identically distributed as the keys generated by the **ReKeyGen** algorithm in the construction. Infact, we have:

· $RK^1_{i\to j} = (g^a)^{\nu x_{ij}\bar{s}_2} = (g^{bz_i})^{-a} \cdot g^{a\nu x_{ij}(x_{ij}^{-1}\nu^{-1}bz_i+\bar{s}_2)} = H_1(id_i)^{-s} \cdot h^{x_{ij}s_2}$.

· $RK^2_{i\to j} = (g^a)^{x_{ij}^{-1}z_i} \cdot (g^{\frac{a}{b}})^{\nu\bar{s}_2} = (g^{\frac{a}{b}})^{\nu(z_i x_{ij}^{-1}\nu^{-1}b+\bar{s}_2)} = h_1^{s_2}$.

∗ If $\alpha_i = 1 \wedge \alpha_j = 1$: Generate the re-keys as below:

· Pick $s_1, \bar{s}_2, x_{ij} \leftarrow \mathbb{Z}_q^*$.

· Compute $RK^1_{i\to j} = (g)^{x_{ij}}$.

· Compute $RK^2_{i\to j} = g^{\frac{a}{b}\nu s_2} = h_1^{\bar{s}_2}$.

· Compute $RK^3_{i\to j} = g^{s_1}$.

Update list L_{RK} with the tuple $\langle id_i, id_j, x_{ij}, RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j}, s_1, \bar{s}_2\rangle$.

Return the re-keys $RK_{i\to j} = (RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j})$ to $\mathcal{A}$.

– **Output**: Eventually, $\mathcal{A}$ returns sk_{id^*} as the private key corresponding to the identity id^*, where id^* is honest ($\alpha^* = 1$). $\mathcal{C}$ recovers the tuple $\langle id^*, y^*, z^*, \alpha^*\rangle$ from list L_{H_1} and returns $(sk_{id^*})^{z^{*-1}}$ as a solution to the hard problem. Note that for an honest identity id^*, $(sk_{id^*})^{z^{*-1}} = (g^{abz^*})^{z^{*-1}} = g^{ab}$ is the solution to the m-CDH problem.

– **Probability Analysis**: We calculate the probability that $\mathcal{C}$ aborts during the simulation. Let *Abort* denote the event that $\mathcal{C}$ aborts during the game and q_{KE} denote the number of queries made to the key extraction oracle. We note that $\mathcal{C}$ does not abort in the following events:

– E_1: $\alpha^* = 0$ in the *Private Key Extraction* phase.

– E_1: $\alpha^* = 1$ in the *Output* phase.

We have $Pr[\neg Abort] \geq \gamma^{q_{KE}}(1-\gamma)$, which has a maximum value at $\gamma_{OPT} = \frac{q_{KE}}{1+q_{KE}}$. Using γ_{OPT}, we obtain:

$$Pr[\neg Abort] \geq \frac{1}{e(1+q_{KE})}.$$

Therefore, the advantage of $\mathcal{C}$ in solving the m-CDH problem is:

$$\begin{aligned}\epsilon' &\geq \epsilon \cdot Pr[\neg Abort] \\ &\geq \frac{\epsilon}{e(1+q_{KE})},\end{aligned}$$

where, e is the base of the natural logarithm. The running time of $\mathcal{C}$ is:

$$t^* \leq t + O(q_{H_1} + 3q_{RK} + q_{KE})t_{et} + O(q_{RK})t_{bp}.$$

This completes the proof of the theorem. □

First-level Ciphertext Security:

Theorem 2. *Our proposed scheme is CCA-secure for the first level ciphertext under the DBDH assumption. If an IND-IBPRE-CCA adversary $\mathcal{A}$ breaks the IND-IBPRE-CCA security of the given scheme with an advantage ϵ within time t, then there exists an adversary $\mathcal{C}$ that solves the DBDH problem with an advantage ϵ' within time t' where,*

$$\epsilon' \geq \frac{\epsilon}{e(1+q_{RK}+q_{KE})},$$

$$t' \leq t + (q_{H_1}+q_{H_2}+q_{H_3}+q_{H_4}+q_{H_5}+q_{KE}+q_{RK}+q_{RE}+q_{DEC} + q_{REDEC})O(1) + (q_{H_1}+q_{H_4}+q_{KE}+4q_{RK}+7q_{RE}+6q_{DEC}+3q_{REDEC})t_{et} + (q_{RK}+5q_{RE}+q_{DEC}+2q_{REDEC})t_{bp},$$

where e is the base of natural logarithm, t_{et} denotes the time taken for exponentiation in group $\mathbb{G}_1$ and t_{bp} denotes the time taken for bilinear pairing operation.

Proof. Let $\mathcal{A}$ be a p.p.t algorithm that has a non-negligible advantage ϵ in breaking the scheme with access to the random oracles H_1, H_2, H_3, H_4, H_5. Then, we can use $\mathcal{A}$ to construct an algorithm $\mathcal{C}$ to solve the DBDH problem in $\mathbb{G}_1$, $\mathbb{G}_T$ with a non-negligible advantage. Algorithm $\mathcal{C}$ accepts as input a properly-distributed tuple $\langle \mathbb{G}_1 = \langle g \rangle, g^a, g^b, g^c, T \in \mathbb{G}_T \rangle$ and outputs 1 if $T = \hat{e}(g,g)^{abc}$. We now describe the algorithm $\mathcal{C}$.

- **Initialisation:** $\mathcal{C}$ implicitly defines the master secret key $msk = a$ and sets $P_{pub_1} = g^a$. $\mathcal{C}$ picks $\delta, \nu \leftarrow \mathbb{Z}_q^*$, computes $g_1 = g^\delta$, $h = g^\nu$, $h_1 = g^{\nu\delta}$ and $P_{pub_2} = (g^a)^\delta$.

- **Phase 1:** $\mathcal{C}$ interacts with $\mathcal{A}$ in the following ways:
  - *Oracle Queries*:
    $\mathcal{C}$ responds to the hash function queries of $\mathcal{A}$ in the following ways:
    * *$H_1(id_i)$ Oracle*: $\mathcal{C}$ responds to the queries of $\mathcal{A}$ by maintaining a list L_{H_1} with tuples of the form $\langle id_i \in \{0,1\}^*,\ y_i \in \mathbb{G}_1,\ z_i \in \mathbb{Z}_q^*, \alpha_i \in \{0,1\} \rangle$. If the tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ already exists in L_{H_1}, retrieve and return the value y_i. Else randomly set $\alpha_i \in \{0,1\}$ such that $Pr[\alpha_i = 0] = \gamma$ which is defined later. Set the hash value according to the following cases:
      · If $\alpha_i = 0$, select $z_i \leftarrow \mathbb{Z}_q^*$, compute $y_i = g^{z_i}$ and set $H_1(id_i) = y_i$.
      · If $\alpha_i = 1$, select $z_i \leftarrow \mathbb{Z}_q^*$, compute $y_i = (g^b)^{z_i}$. Set $H_1(id_i) = y_i$.
      Store tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ in list L_{H_1} and return y_i.

    * *$H_2(\sigma, m)$ Oracle*: $\mathcal{C}$ maintains a list L_{H_2} with tuples of the form $\langle \sigma \in \mathbb{G}_T,\ m \in \{0,1\}^n, r \in_R \mathbb{Z}_q^* \rangle$. If the tuple $\langle \sigma, m, r \rangle$ already exists in L_{H_2}, retrieve and return the value r. Else, choose $r \leftarrow \mathbb{Z}_q^*$, set $H_2(\sigma, m) = r$, store tuple $\langle \sigma, m, r \rangle$ to L_{H_2} and return r.

    * *$H_3(\sigma)$ Oracle*: $\mathcal{C}$ maintains a list L_{H_3} with tuples of the form $\langle \sigma \in \mathbb{G}_T,\ p \in \{0,1\}^n \rangle$. If the tuple $\langle \sigma, p \rangle$ already exists in L_{H_3}, retrieve

and return the value p. Else, choose $p \leftarrow \{0,1\}^n$, set $H_3(\sigma) = p$, store the tuple $\langle \sigma, p \rangle$ to L_{H_3} and return p.

* *$H_4(id_i||\{0,1\}^*)$ Oracle*: $\mathcal{C}$ maintains a list L_{H_4} with tuples of the form $\langle id_i \in \{0,1\}^n,\ \mathbb{C}, z_i \in \mathbb{Z}_q^*, k_i \in \mathbb{Z}_q^*, l_i \in \mathbb{G}_1, \alpha_i \in \{0,1\} \rangle$. If the tuple $\langle id_i, \mathbb{C}, z_i, k_i, l_i, \alpha_i \rangle$ already exists in L_{H_4}, retrieve and return the value l. Else, check if the tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ already exists in L_{H_1}. If the tuple exists, compute the hash value as shown next. If does not exist, randomly set $\alpha_i \in \{0,1\}$ such that $Pr[\alpha_i = 0] = \gamma$, compute the value of $H_1(id_i)$ as shown above and store tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ in L_{H_1}. Compute the hash value according to the following cases:
  - If $\alpha_i = 0$, select $k_i \leftarrow \mathbb{Z}_q^*$, compute $l_i = (g^{z_i})^{k_i}$.
  - If $\alpha_i = 1$, select $k_i \leftarrow \mathbb{Z}_q^*$, compute $l_i = (g^{bz_i})^{k_i}$.

  Store the tuple $\langle id_i, \mathbb{C}, z_i, k_i, l_i, \alpha_i \rangle$ to L_{H_4} and return l_i.

* *$H_5(X, id_i, id_j)$ Oracle*: $\mathcal{C}$ maintains a list L_{H_5} with tuples of the form $\langle X \in \mathbb{G}_T,\ id_i \in \{0,1\}^*,\ id_j \in \{0,1\}^*,\ x_{ij} \in \mathbb{Z}_q^* \rangle$. If the tuple $\langle X, id_i, id_j, x_{ij} \rangle$ already exists in L_{H_5}, retrieve and return the value x_{ij}. Else, choose $x_{ij} \leftarrow \mathbb{Z}_q^*$, set $H_5(X, id_i, id_j) = x_{ij}$, store the tuple $\langle X, id_i, id_j, x_{ij} \rangle$ to L_{H_5} and return x_{ij}. If $\mathcal{A}$ queries H_5 oracle with $(T^{\bar{s}_1 z_i}, id_i, id_j)$ where $\bar{s}_1 \in \mathbb{Z}_q^*$, abort and solve the m-DBDH problem.

- *Private Key Extraction Query* $\mathcal{O}_{KE}(id_i)$: $\mathcal{C}$ responds to a private key-extraction query for user id_i by searching list L_{H_1} for a tuple $\langle id_i, y_i, z_i, \alpha_i \rangle$ corresponding to id_i and computing the private key as below:
  * If $\alpha_i = 1$, return $failure$ and abort.
  * Otherwise, compute $sk_{id_i} = (g^a)^{z_i} = H_1(id_i)^s$ and return sk_{id_i}.

- *Re − Encryption Key Generation Query* $\mathcal{O}_{RK}(id_i, id_j)$: $\mathcal{C}$ maintains a list L_{RK} with tuples $\langle id_i \in \{0,1\}^*, id_j \in \{0,1\}^*, x_{ij} \in \mathbb{Z}_q^*, RK_{i\to j}^1 \in \mathbb{G}_1, RK_{i\to j}^2 \in \mathbb{G}_1, RK_{i\to j}^3 \in \mathbb{G}_1, s_1 \in \mathbb{Z}_q^*, \bar{s}_2 \in \mathbb{Z}_q^* \rangle$ to store the re-encryption keys. When $\mathcal{A}$ sends a re-key generation query from user id_i to id_j, $\mathcal{C}$ computes the re-key according to the following cases:
  * Check if the re-encryption key $RK_{i\to j}$ exists in L_{RK} by searching for a tuple $\langle id_i, id_j, x_{ij}, RK_{i\to j}^1, RK_{i\to j}^2, RK_{i\to j}^3, s_1, \bar{s}_2 \rangle$. If present, return $RK_{i\to j}$.
  * Check for a tuple and $\langle id_i, y_i, z_i, \alpha_i \rangle$ and $\langle id_j, y_j, z_j, \alpha_j \rangle \in L_{H_1}$ and compute the re-keys according to the following cases.
  * If $\alpha_i = 1 \wedge \alpha_j = 0$, return $failure$ and abort.
  * If $\alpha_i = 1 \wedge \alpha_j = 1$, pick $x_{ij}, \bar{s}_1, s_2 \leftarrow \mathbb{Z}_q^*$ and implicitly define $s_1 = c\bar{s_1}$. Compute $RK_{i\to j}^1 = g^{x_{ij}}$, $RK_{i\to j}^2 = g^{\nu\delta s_2} = h_1^{\bar{s}_2}$ and $RK_{i\to j}^3 = g^{c\bar{s}_1}$. Store tuple $\langle id_i, id_j, x_{ij}, RK_{i\to j}^1, RK_{i\to j}^2, RK_{i\to j}^3, \bar{s}_1, \bar{s}_2 \rangle$ in L_{RK}.
  * If $\alpha_i = 0 \ \wedge \ \alpha_j = 0$, generate re-keys as per the re-encryption key-generation protocol.

* If $\alpha_i = 0 \ \wedge \ \alpha_j = 1$, generate re-keys as per the re-encryption key-generation protocol.

Update list L_{RK} with the tuple $\langle id_i, id_j, x_{ij}, RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j}, s_1, \bar{s}_2\rangle$ and return $RK_{i\to j} = (RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j})$ to $\mathcal{A}$.

- *Re − Encryption Query* $\mathcal{O}_{RE}(id_i, id_j, \mathbb{C})$: Given a re-encryption query for an original ciphertext $\mathbb{C}$ from user id_i to id_j, the challenger $\mathcal{C}$ validates the well-formedness of the ciphertext by checking if condition 6 holds. If it does not hold, $\mathcal{C}$ returns *"INVALID CIPHERTEXT"*, otherwise it re-encrypts as below:
  * If $\alpha_i = 1 \ \wedge \ \alpha_j = 0$, retrieve tuple $\langle id_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, z_i, k_i, l_i, \alpha_i\rangle$ from L_{H_4}.
  * Compute $Q = \mathbb{C}_5^{k_i^{-1}}$.
  * Compute $\sigma = \mathbb{C}_3 / \hat{e}(P_{pub_2}, Q)$.
  * Check the list L_{RK} for tuple of the form $\langle id_i, id_j, x_{ij}, RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j}, s_1, \bar{s}_2\rangle$. If no such tuple exists, pick $s_1, \bar{s}_2 \leftarrow \mathbb{Z}_q^*$ and compute $x_{ij} = H_5(\hat{e}(P_{pub_1}, g^{bz_j})^{s_1})$. Otherwise compute the re-encrypted ciphertext as follows:
  * Set $\mathbb{D}_1 = \mathbb{C}_1$.
  * Compute $\mathbb{D}_2 = g^{s_1}$.
  * Compute $\mathbb{D}_3 = \sigma \cdot \hat{e}(\mathbb{C}_2, h^{x_{ij}\bar{s}_2})$.
  * Set $\mathbb{D}_4 = \mathbb{C}_4$.
  * Compute $\mathbb{D}_5 = h_1^{\bar{s}_2}$.
  * Return the re-encrypted ciphertext as $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3, \mathbb{D}_4, \mathbb{D}_5)$.

  For all other cases, generate the re-encryption key $RK_{i\to j}$ using the re-encryption key generation oracle and run the $Re - Encrypt$ algorithm to generate the re-encrypted ciphertext $\mathbb{D}$.

- *Decryption Query* $\mathcal{O}_{DEC}(id_i, \mathbb{C})$: Given a decryption query for a first level ciphertext $\mathbb{C}$, the challenger $\mathcal{C}$ validates the well-formedness of the ciphertext by checking if conditions 1 and 2 hold. If they do not hold, $\mathcal{C}$ returns *"INVALID CIPHERTEXT"*. Otherwise, if $\alpha_i = 0$ for identity id_i, $\mathcal{C}$ runs the **Decrypt** algorithm to retrieve the plaintext m. For an id_i such that $\alpha_i = 1$, decrypt as below:
  * Check list L_{H_4} for a tuple of the form $\langle id_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, z_i, k_i, l_i, \alpha_i\rangle$. If it does not exist, return *"INVALID CIPHERTEXT"*.
  * Otherwise, compute $Q = \mathbb{C}_5^{k_i^{-1}}$.
  * Compute $\sigma = \mathbb{C}_3 / \hat{e}(P_{pub_2}, Q)$.
  * Compute $m = \mathbb{C}_4 \oplus H_3(\sigma)$.
  * If $\mathbb{C}_1 \stackrel{?}{=} g^{H_2(\sigma, m)}$ and $\mathbb{C}_2 \stackrel{?}{=} h^{H_2(\sigma, m)}$, output m, otherwise output *"INVALID CIPHERTEXT"*.

- *Re − Decryption Query* $\mathcal{O}_{REDEC}(id_i, \mathbb{D})$: Given a decryption query for a user with id_i where $\alpha = 0$, $\mathcal{C}$ runs the **Re-Decrypt** algorithm to retrieve the plaintext m. For an id_i such that $\alpha_i = 1$, decrypt as below:

* Check L_{RK} for the existence of a tuple $\langle id_j, id_i, x_{ji}, RK^1_{j\to i}, RK^2_{j\to i}, RK^3_{j\to i}, s_1, \bar{s}_2\rangle$ such that $D_2 = g^{s_1}$. If no such tuple exists, return *"INVALID CIPHERTEXT"*.
* Check list L_{H_1} if $\alpha_i = \alpha_j = 1$. If true, compute $\overline{\mathbb{D}}_3 = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_1^{\delta}, RK^1_{i\to j})}$. Check list L_{H_2} for a tuple of the form $\langle \sigma, m, r\rangle$ such that the following conditions are satisfied:

$$\mathbb{D}_1 \stackrel{?}{=} g^r,$$
$$\mathbb{D}_2 \stackrel{?}{=} RK^3_{i\to j},$$
$$\overline{\mathbb{D}}_3 \stackrel{?}{=} \sigma \cdot \hat{e}(P_{Pub_2}, g^{bz_j r}),$$
$$\mathbb{D}_5 \stackrel{?}{=} RK^2_{i\to j}.$$

  If all the conditions hold, return m, otherwise return *"INVALID CIPHERTEXT"*.
* Otherwise, compute $X = \hat{e}(P_{pub}, H_1(id_i)^{s_1})$. Search list L_{H_5} for a tuple $\langle X, id_j, id_i, x_{ij}\rangle$ and compute $\sigma = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_5^{x_{ij}}, \mathbb{D}_1)}$. Compute $m = \mathbb{D}_4 \oplus H_5(\sigma)$. If $\mathbb{D}_1 = h^{H_2(\sigma,m)}$, return m, else return *"INVALID CIPHERTEXT"*.

– **Challenge:** Once $\mathcal{A}$ decides that Phase-1 is over, it outputs two messages m_0, m_1, the target identity id^* on which it wishes to be challenged. $\mathcal{C}$ checks list L_{H_1} if $\alpha^* = 1$, and if it holds, it aborts. It tosses a coin and chooses $\psi \in \{0,1\}$ uniformly at random and simulates the challenge ciphertext for m_ψ as shown stepwise:
  1. Pick $\sigma \leftarrow \mathbb{G}_T,\ p \in \{0,1\}^n$, and store tuple $\langle \sigma, p\rangle$ in list L_{H_3}.
  2. Compute $\mathbb{C}_4^* = m_\psi \oplus p$.
  3. Implicitly define $r = c$, where c is not known to the challenger $\mathcal{C}$. Set $\mathbb{C}_1 = g^c,\ \mathbb{C}_2 = (g^c)^{\delta}$.
  4. Pick $z^* \in \mathbb{Z}_q^*$, define $H_1(id^*)$ as $(g^b)^{z^*}$ and update list L_{H_1}.
  5. Compute $\mathbb{C}_3^* = \sigma \cdot T^{\delta z^*}$.
  6. Pick $k^* \in \mathbb{Z}_q^*$. Update list L_{H_4} with tuple $\langle id^*, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, z, k, g^{z^*k^*}, \alpha^* = 1\rangle$ and compute $\mathbb{C}_5^* = (g^c)^{z^*k^*}$.
  7. Return the first-level challenge ciphertext $\mathbb{C}^* = (\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_4^*, \mathbb{C}_5^*)$.

  Note that the challenge ciphertext $\mathbb{C}^* = (\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_4^*, \mathbb{C}_5^*)$ is identically distributed to the real original ciphertext generated by the encryption algorithm. Hence, the challenge ciphertext is a valid ciphertext.

– **Phase-2:** The adversary $\mathcal{A}$ continues to query the oracles maintained by $\mathcal{C}$ with the restrictions stated in the security model.

– **Guess:** The adversary $\mathcal{A}$ eventually produces its guess $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathcal{A}$ wins the game and $\mathcal{C}$ decides $\hat{e}(g,g)^{abc} = T$, else T is random.

– **Probability Analysis:** We calculate the probability that $\mathcal{C}$ aborts during the simulation. Let $\mathcal{A}$ issue q_{KE} key-extraction and q_{RK} re-encryption key generation queries. Let *Abort* denote the event that $\mathcal{C}$ aborts during the game. We note that $\mathcal{C}$ does not abort in the following events:
  – E_1: $\alpha_i = 0$ in the *re-encryption key-generation* query.
  – E_2: $\alpha_i = 0$ in the *key-generation* query.
  – E_3: $\alpha^* = 1$ in the *Challenge* phase.

  We have $Pr[\neg Abort] \geq \gamma^{q_{RK}+q_{KE}}(1-\gamma)$, which has a maximum value at $\gamma_{OPT} = \frac{q_{RK}+q_{KE}}{1+q_{RK}+q_{KE}}$. Using γ_{OPT}, we obtain:

$$Pr[\neg Abort] \geq \frac{1}{e(1+q_{RK}+q_{KE})}.$$

  Therefore, the advantage of $\mathcal{C}$ in solving the DBDH problem is:

$$\begin{aligned}\epsilon' &\geq \epsilon \cdot Pr[\neg Abort] \\ &\geq \frac{\epsilon}{e(1+q_{RK}+q_{KE})},\end{aligned}$$

  where, e is the base of the natural logarithm. We also bound the running time of $\mathcal{C}$ to solve DCDH instance by:

$$\begin{aligned}t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{KE} + q_{RK} + q_{RE} + q_{DEC} \\ & + q_{REDEC})O(1) + (q_{H_1} + q_{H_4} + q_{KE} + 4q_{RK} + 7q_{RE} + 6q_{DEC} + 3q_{REDEC})t_{et} \\ & + (q_{RK} + 5q_{RE} + q_{DEC} + 2q_{REDEC})t_{bp}.\end{aligned}$$

  This completes the proof of the theorem. □

Second-level Ciphertext Security:

Theorem 3. *Our proposed scheme is CCA-secure for the second-level ciphertext under the m-DBDH assumption. If an IND-IBPRE-CCA adversary $\mathcal{A}$ breaks the IND-IBPRE-CCA security of the given scheme with a non-negligible advantage, then there exists an adversary $\mathcal{C}$ who solves the m-DBDH problem with an advantage ϵ' within time t' where,*

$$\epsilon' \geq \frac{\epsilon}{e(1+q_{KE})} - \frac{q_{H_5}}{2^{2n}},$$

$$\begin{aligned}t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{KE} + q_{RK} + q_{DEC} + q_{REDEC})O(1) \\ & + (q_{H_1} + q_{H_4} + 5q_{RK} + 6q_{DEC} + 3q_{REDEC})t_{et} \\ & + (q_{RK} + q_{DEC} + 2q_{REDEC})t_{bp},\end{aligned}$$

where e is the base of natural logarithm, t_{et} denotes the time taken for exponentiation in group $\mathbb{G}_1$ and t_{bp} denotes the time taken for bilinear pairing operation.

Proof. Let $\mathcal{A}$ be a p.p.t algorithm that has a non-negligible advantage ϵ in breaking the given scheme with access to the random oracles H_1, H_2, H_3, H_4, H_5. Then, we can use $\mathcal{A}$ to construct an algorithm $\mathcal{C}$ to solve the m-DBDH problem in $\mathbb{G}$, $\mathbb{G}_T$ with a non-negligible advantage. Algorithm $\mathcal{C}$ accepts as input a properly-distributed tuple $\langle \mathbb{G}_1 = \langle g \rangle, g^a, g^b, g^c, g^{\frac{1}{a}}, g^{\frac{1}{b}}, g^{\frac{a}{b}}, T \in \mathbb{G}_T \rangle$ and outputs 1 if $T = \hat{e}(g,g)^{abc}$. We now describe the algorithm $\mathcal{C}$.

– **Initialisation:** $\mathcal{C}$ implicitly defines the master secret key $msk = a$ and $\delta = \frac{1}{b}$. $\mathcal{C}$ sets $P_{pub_1} = g^a$, $g_1 = g^{\frac{1}{b}}$ and $P_{pub_2} = g^{\frac{a}{b}}$. $\mathcal{C}$ picks $\nu \leftarrow \mathbb{Z}_q^*$ and sets $h = (g^a)^\nu$ and $h_1 = (g^{\frac{a}{b}})^\nu$.

– **Phase 1:** $\mathcal{C}$ interacts with $\mathcal{A}$ in the following ways:

- *Oracle Queries*: $\mathcal{C}$ responds to the H_1, H_2, H_3, H_4 and H_5 hash function queries of $\mathcal{A}$ in the same way as it does for the security of original ciphertext.

• *Private Key Extraction Query* ($\mathcal{O}_{KE}$): $\mathcal{C}$ responds to the key extraction queries of $\mathcal{A}$ in the same way as it does for the security of original ciphertext.

• *Re − Encryption Key Generation Query* ($\mathcal{O}_{RK}$): $\mathcal{C}$ maintains a list L_{RK} that contains tuples of the form $\langle id_i \in \{0,1\}^*, id_j \in \{0,1\}^*, x_{ij} \in \mathbb{Z}_q^*, RK^1_{i\to j} \in \mathbb{G}_1, RK^2_{i\to j} \in \mathbb{G}_1, RK^3_{i\to j} \in \mathbb{G}_1, s_1 \in \mathbb{Z}_q^*, \bar{s}_2 \in \mathbb{Z}_q^* \rangle$. $\mathcal{C}$ responds to the re-encryption key-generation queries of $\mathcal{A}$ from user id_i to id_j by searching list L_{H_1} for tuples corresponding to id_i and id_j respectively and computing the re-keys as per the following cases:

  * Check if the re-encryption key $RK_{i\to j}$ exists in L_{RK} by searching for a tuple $\langle id_i, id_j, x_{ij}, RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j}, s_1, \bar{s}_2 \rangle$. If present, return $RK_{i\to j}$.
  * If $\alpha_i = 0 \wedge \alpha_j = 0$: Generate the re-keys as per **ReKeyGen** protocol.
  * If $\alpha_i = 0 \wedge \alpha_j = 1$: Generate the re-keys as per **ReKeyGen** protocol.
  * If $\alpha_i = 1 \wedge \alpha_j = 0$: Pick $x_{ij}, s_1, \bar{s}_2 \leftarrow \mathbb{Z}_q^*$. Update list L_{H_5} with the tuple $\langle \hat{e}(g^a, g^{z_j s_1}), id_i, id_j, x_{ij} \rangle$. Implicitly define $s_2 = z_i x_{ij}^{-1} \nu^{-1} b + \bar{s}_2$ and compute the re-keys as follows:
    · Compute $RK^1_{i\to j} = (g^a)^{\nu x_{ij} \bar{s}_2}$.
    · Compute $RK^2_{i\to j} = (g^a)^{x_{ij}^{-1} z_i} \cdot (g^{\frac{a}{b}})^{\nu \bar{s}_2}$.
    · Compute $RK^3_{i\to j} = g^{s_1}$.
  * If $\alpha_i = 1 \wedge \alpha_j = 1$: Compute the re-keys as follows:
    · Pick $s_1, \bar{s}_2, x_{ij} \leftarrow \mathbb{Z}_q^*$.
    · Compute $RK^1_{i\to j} = (g)^{x_{ij}}$.
    · Compute $RK^2_{i\to j} = g^{\frac{a}{b} \nu s_2} = h_1^{\bar{s}_2}$.
    · Compute $RK^3_{i\to j} = g^{s_1}$.

  The correctness of re-key generation given above is shown in the re-key generation oracle of Collusion Resistance proof in Section 4.4. Update list L_{RK} with the tuple $\langle id_i, id_j, x_{ij}, RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j}, s_1, \bar{s}_2 \rangle$. Return the re-keys $RK = (RK^1_{i\to j}, RK^2_{i\to j}, RK^3_{i\to j})$ to $\mathcal{A}$.

• *Decryption Query* ($\mathcal{O}_{DEC}$): $\mathcal{C}$ responds to the decryption queries of $\mathcal{A}$ in the same way as it does for the security of original ciphertext.

• *Re − Decryption Query* ($\mathcal{O}_{REDEC}$): On a re-decryption queries of $\mathcal{A}$ for user id_i, $\mathcal{C}$ responds as per the following cases:

* Check list L_{H_1} for a tuple corresponding to id_i. If $\alpha_i = 0$, $\mathcal{C}$ runs the *Re-Decrypt* algorithm to obtain the plaintext m or *"INVALID CIPHERTEXT"*.
* Otherwise, check L_{RK} for the existence of a tuple $\langle id_j, id_i, x_{ji}, RK^1_{j\to i}, RK^2_{j\to i}, RK^3_{j\to i}, s_1, s_2\rangle$ such that $\alpha_j = 0$, $\mathbb{D}_2 = RK^3_{j\to i}$ and $\mathbb{D}_5 = RK^2_{j\to i}$. Compute $X = e(P_{pub}, H_1(id_i)^{s_1})$. Search list L_{H_5} for a tuple of the form $\langle X, id_i, id_j, s_{ij}\rangle$ and compute $\sigma = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_5^{x_{ij}}, \mathbb{D}_1)}$ and $m = \mathbb{D}_4 \oplus H_3(\sigma)$. If $\mathbb{D}_1 = g^{H_2(\sigma,m)}$, return m, else return *"INVALID CIPHERTEXT"*.
* Check list L_{H_1} if $\alpha_i = \alpha_j = 1$. If true, compute $\overline{\mathbb{D}}_3 = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_1^{\delta}, RK^1_{i\to j})}$. Check list L_{H_2} for a tuple of the form $\langle \sigma, m, r\rangle$ such that the following conditions are satisfied:

$$\mathbb{D}_1 \stackrel{?}{=} g^r,$$
$$\mathbb{D}_2 \stackrel{?}{=} RK^3_{i\to j},$$
$$\overline{\mathbb{D}}_3 \stackrel{?}{=} \sigma \cdot \hat{e}(P_{pub_2}, g^{bz_j r}),$$
$$\mathbb{D}_5 \stackrel{?}{=} RK^2_{i\to j}.$$

  If all the conditions hold, return m, otherwise return *"INVALID CIPHERTEXT"*.
* Otherwise, compute $X = \hat{e}(P_{pub}, H_1(id_i)^{s_1})$. Search list L_{H_5} for a tuple $\langle X, id_j, id_i, x_{ij}\rangle$ and compute $\sigma = \frac{\mathbb{D}_3}{\hat{e}(\mathbb{D}_5^{x_{ij}}, \mathbb{D}_1)}$. Compute $m = \mathbb{D}_4 \oplus H_5(\sigma)$. If $\mathbb{D}_1 = h^{H_2(\sigma,m)}$, return m, else return *"INVALID CIPHERTEXT"*.

– **Challenge:** Once $\mathcal{A}$ decides that Phase-1 is over, it outputs two messages m_0, m_1, the delegator (id_i) and the target identity id^* (delegatee) on which it wishes to be challenged. $\mathcal{C}$ checks list L_{H_1} if $\alpha^* = 1$ for identity id^* and if $\alpha_1 = 0$ for id_i. If either of the checks hold, it aborts. It tosses a coin and chooses $\psi \in \{0, 1\}$ uniformly at random. It computes the re-encryption key $RK_{i\to *}$ as below:
  - Pick $\bar{s}_1, \bar{s}_2, \bar{x}, \beta \leftarrow \mathbb{Z}_q^*$. Define $s_1 = c\beta$ and $x_{i*} = b^2$.
  - Compute $RK^1_{i\to *} = g^{\bar{x}}$.
  - Compute $RK^2_{i\to *} = (g^{\frac{a}{b}\nu\bar{s}_2}) = h_1^{s_2}$.
  - Compute $RK^3_{i\to *} = (g^c)^{\beta} = g^{s_1}$.

  The challenger $\mathcal{C}$ simulates the second-level challenge ciphertext for m_ψ as shown stepwise:
  1. Pick $\sigma \leftarrow \mathbb{G}_T$ and implicitly defines $r = c$, where c is not known to the challenger $\mathcal{C}$.
  2. $\mathcal{C}$ picks $p \in \{0, 1\}^n$, and store tuple $\langle \sigma, p\rangle$ in list L_{H_3}.
  3. Compute $\mathbb{D}_4^* = m_\psi \oplus p$.
  4. Set $\mathbb{D}_1^* = g^c = g^r$.
  5. Compute $\mathbb{D}_2^* = g^{c\cdot\beta} = g^{s_1}$.

6. Pick $z^* \in \mathbb{Z}_q^*$, define $H_1(id^*)$ as $(g^b)^{z^*}$ and update list L_{H_1}.
7. Compute $\mathbb{D}_3^* = \sigma \cdot T^{\nu \bar{s}_2}$.
8. Set $\mathbb{D}_5^* = (g^{\frac{a}{b}})^{k_i} \cdot g^{\frac{1}{b}\nu \bar{s}_2} = RK_{i \to *}^2$.
9. Return the challenge ciphertext $\mathbb{D}^* = (\mathbb{D}_1^*, \mathbb{D}_2^*, \mathbb{D}_3^*, \mathbb{D}_4^*, \mathbb{D}_5^*)$.

Note that the challenge ciphertext $\mathbb{D}^* = (\mathbb{D}_1^*, \mathbb{D}_2^*, \mathbb{D}_3^*, \mathbb{D}_4^*, \mathbb{D}_5^*)$ is identically distributed to the real original ciphertext generated by the encryption algorithm. Hence, the challenge ciphertext is a valid ciphertext.

– **Phase-2:** The adversary $\mathcal{A}$ continues to query the oracles maintained by $\mathcal{C}$ with the restrictions stated in the security model.

– **Guess:** The adversary $\mathcal{A}$ eventually produces its guess $\psi' \in \{0, 1\}$. If $\psi' = \psi$, $\mathcal{C}$ decides $e(g, g)^{abc} = T$, else T is random.

– **Probability Analysis:** We calculate the probability that $\mathcal{C}$ aborts during the simulation. Let *Abort* denote the event that $\mathcal{C}$ aborts during the game and q_{KE} denote the number of queries made to the key extraction oracle. We note that $\mathcal{C}$ does not abort in the following events:
  – E_1: $\alpha^* = 0$ in the $Key - Extraction$ phase.
  – E_1: $\alpha^* = 1$ in the $Challenge$ phase.

  We have $Pr[\neg Abort] \geq \gamma^{q_{KE}}(1-\gamma)$, which has a maximum value at $\gamma_{OPT} = \frac{q_{KE}}{1+q_{KE}}$. Using γ_{OPT}, we obtain:

$$Pr[\neg Abort] \geq \frac{1}{e(1+q_{KE})}.$$

  We next analyze the simulation of the random oracles. Note that the simulation is perfect unless the following event takes place:
  – E_{H_5}: $\mathcal{A}$ queries H_5 oracle with $(T^{\beta z_i}, id_i, id_j)$.

  We note that $Pr[E_{H_5}] = \frac{q_{H_5}}{2^{2n}}$. Therefore, the advantage of $\mathcal{C}$ in solving the m-DBDH problem is:

$$\epsilon' \geq \frac{\epsilon}{e(1+q_{KE})} - \frac{q_{H_5}}{2^{2n}},$$

  where, e is the base of the natural logarithm. We also bound the running time of $\mathcal{C}$ to solve m-DBDH instance by:

$$\begin{aligned} t' \leq\ & t + (q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{KE} + q_{RK} + q_{DEC} + q_{REDEC})O(1) \\ & + (q_{H_1} + q_{H_4} + 5q_{RK} + 6q_{DEC} + 3q_{REDEC})t_{et} \\ & + (q_{RK} + q_{DEC} + 2q_{REDEC})t_{bp}. \end{aligned}$$

  This completes the proof of the theorem. □

5 Discussion on the collusion-resistant IB-PRE scheme in [18]

In the scheme due to Wang *et al.* [18], the first-level ciphertext C_{ID} consists of the following components: $(C_1, C_2, C_3, C_4, C_5, C_6, C_7)$=$(g^r, (g_2 g_3)^r, (g_1^{ID} h)^r$, SE.Enc

$(H_2(e(g_1, g_2))^r), M), H_1(svk)^r, svk, \sigma)$. Note that the hash function H_1 is defined as $H_1 : \mathbb{S} \rightarrow \mathbb{G}$, where $\mathbb{S}$ is the public key space of the one time signature scheme used in the construction. In order to simulate the ciphertext component $C_5 = H_1(svk)^r$ in the Challenge phase, the Challenger must know the value r to form a valid ciphertext component C_5, and when we know such an r, that would solve the discrete log problem with respect to C_1. Also, wellformedness of the ciphertext can be verified using the following check: $e(C_1, H_1(C_6)) \stackrel{?}{=} e(g, C_5)$. Therefore, the challenge ciphertext cannot be simulated without the knowledge of the exponent r, which would lead to solving the discrete log problem.

Another big omission noticed in [18] is that, the private key generation uses a signature scheme whose security is not proven or known. It is quite unlikely that a standard model proof is possible for this signature scheme since the key components d_1 and d_1' are not used as exponents of the generator but used directly as elements of $\mathbb{Z}_p^*$. In fact, there are no known signature schemes in the standard model with the keys used directly.

In light of the above observations, it is impossible to prove the security of the scheme due to Wang *et al.* [18].

6 Conclusion

Though there are several proxy re-encryption (PRE) schemes in the literature constructed in the identity based setting, only one IB-PRE scheme due to Wang *et al.*[18] which is CPA-secure for the first-level ciphertext and CCA-secure for the second-level ciphertext, has reported the collusion resistance property in the standard model and adheres to the standard definition of PRE. However, the scheme is not provably secure, as discussed in our work. Our collusion resistant IB-PRE scheme adheres to the standard definition of PRE, and is shown to be adaptively CCA secure in the random oracle model for both the first-level and second level ciphertexts. Also, the definition of collusion resistance is met wherein the colluders (proxy and the delegatee) cannot obtain the private key components of the delegator. Thus, this paper proposes the ***first*** provably secure collusion resistant IB-PRE scheme based on the Decisional Bilinear Diffie Hellman (DBDH) assumption and its variants in the random oracle model.

References

1. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *IN NDSS*, 2005.
2. Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan 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. Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 127–144. Springer, 1998.

4. Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.
6. Sherman S. M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng. Efficient unidirectional proxy re-encryption. *IACR Cryptology ePrint Archive*, 2009:189, 2009.
7. Sherman S. M. Chow, Jian Weng, Yanjiang Yang, and Robert H. Deng. Efficient unidirectional proxy re-encryption. In *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, pages 316–332, 2010.
8. Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security, 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007, Proceedings*, pages 288–306, 2007.
9. Jinguang Han, Willy Susilo, and Yi Mu. Identity-based data storage in cloud computing. *Future Generation Comp. Syst.*, 29(3):673–681, 2013.
10. Woo Kwon Koo, Jung Yeon Hwang, and Dong Hoon Lee. Security vulnerability in a non-interactive id-based proxy re-encryption scheme. *Information Processing Letters*, 109(23):1260 – 1262, 2009.
11. Benoît Libert and Damien Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Pairing-Based Cryptography - Pairing 2008, Second International Conference, Egham, UK, September 1-3, 2008. Proceedings*, pages 332–353, 2008.
12. J Qiu, J Jo, and H Lee. Collusion-resistant identity-based proxy re-encryption without random oracles. *International Journal of Security and Its Applications*, 9(9):337–344, 2015.
13. Jun Shao, Zhenfu Cao, and Peng Liu. SCCR: a generic approach to simultaneously achieve CCA security and collusion-resistance in proxy re-encryption. *Security and Communication Networks*, 4(2):122–135, 2011.
14. Smith. Tony. Dvd jon: buy drm-less tracks from apple itunes. http://www.theregister.co.uk/2005/03/18/itunes pymusique, 2005.
15. S Sree Vivek, S Sharmila Deva Selvi, V Radhakishan, and C Pandu Rangan. Efficient conditional proxy re-encryption with chosen ciphertext security. *International Journal of Network Security & Its Applications*, 4(2):179, 2012.
16. Hongbing Wang, Peng Zeng, and Kim-Kwang Raymond Choo. MDMR-IBE: efficient multiple domain multi-receiver identity-based encryption. *Security and Communication Networks*, 7(11):1641–1651, 2014.
17. Lihua Wang, Licheng Wang, Masahiro Mambo, and Eiji Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In *Pairing-Based Cryptography - Pairing 2010 - 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, pages 327–346, 2010.
18. X. A. Wang and W. Zhong. A new identity based proxy re-encryption scheme. In *2010 International Conference on Biomedical Engineering and Computer Science*, pages 1–4, April 2010.
19. Linchao Zhang, Hua Ma, Zhenhua Liu, and Enting Dong. Security analysis and improvement of A collusion-resistant identity-based proxy re-encryption scheme. In *Advances on Broad-Band Wireless Computing, Communication and Applications, Proceedings of the 11th International Conference On Broad-Band Wireless Computing, Communication and Applications, BWCCA 2016, Soonchunhyang University, Asan, Korea, November 5-7, 2016.*, pages 839–846, 2016.