Non-transferability in Proxy Re-Encryption Revisited

Arinjita Paul[1]*, Lihua Wang[2], S. Sharmila Deva Selvi[1], and C. Pandu Rangan[1]
[1]Department of Computer Science and Engineering, IIT Madras, Chennai, India
{arinjita,sharmila,prangan}@cse.iitm.ac.in
[2]National Institute of Information and Communications Technology (NICT), Tokyo, Japan
lh-wang@nict.go.jp

Abstract

Proxy re-encryption (PRE) is a cryptographic primitive envisioned by Blaze, Bleumer and Strauss to realise delegation of decryption rights from a delegator to a delegatee via a semi-trusted proxy. The widely accepted model for PRE security prevents the proxy, which is equipped with transformation power, to learn anything about the underlying plaintext. However, such a security notion is not sufficient in practical scenarios wherein the proxy could be corrupted. In this work, we study an attractive property of PRE, namely non-transferability that prevents the colluding proxy and the delegatee from re-delegating decryption rights to a malicious user. In Pairing 2010, a CPA secure non-transferable identity-based PRE scheme was presented in the random oracle model. In this work, we show that the scheme does not realize non-transferability. Also, we formalize the notion of a non-transferable PRE and introduce a security definition for the same. We then present the first provably secure construction of a non-transferable PRE scheme in the PKI setting based on bilinear maps. Our scheme meets chosen ciphertext security and non-transferability in the random oracle model assuming a variant of the decisional bilinear Diffie-Hellman problem.

Keywords: Proxy Re-Encryption, Non-transferability, Unidirectional, CCA-secure, Random Oracle, Bilinear Pairing

1 Introduction

As a solution towards achieving delegation of decryption rights, Blaze et al. [4] in Eurocrypt 1998 first proposed the concept of proxy re-encryption (PRE). PRE allows a proxy with specific information (re-encryption key) to translate ciphertexts originally intended for Alice (delegator) into ciphertexts (of the same message) for Bob (delegatee), without learning anything about the underlying plaintext or private keys of either parties. Proxy re-encryption has recently drawn huge interests befitting diverse applications, such as enabling secure data sharing in the cloud [19, 16]. A user could store shared data in the cloud encrypted under his public key, which can be transformed by a proxy-server (cloud server) into a ciphertext for an authorized user. This offloads the expensive task of secure data sharing to the semi-trusted resource-abundant proxy server. PRE additionally has extensive applications to distributed storage in blockchain-based systems, encrypted email forwarding, digital rights management, outsourced filtering of encrypted spam, revocation systems, network security among others [9, 20, 11, 27, 5, 1, 2, 28].

PRE schemes are classified into bidirectional and unidirectional schemes based on delegation direction. In unidirectional schemes, given a re-encryption key $RK_{\text{Alice}\rightarrow\text{Bob}}$ from a delegator Alice to delegatee Bob, a proxy can re-encrypt ciphertexts only from Alice to Bob and not vice-versa. In bidirectional schemes [26], the proxy is allowed re-encryption in both directions using the re-encryption key $RK_{\text{Alice}\leftrightarrow\text{Bob}}$. PRE schemes can also also classified into single-hop and multi-hop schemes. In a single

*Corresponding author: Department of Computer Science and Engineering, IIT Madras, Chennai-600036, India, Tel: +91-44-2257-5387 hop setting, a proxy cannot re-encrypt ciphertexts that have been already re-encrypted once. In a multi-hop scenario [23], the proxy can further transform a re-encrypted ciphertext multiple times. A good analysis of PRE schemes is given in [21]. This work is particularly focused on unidirectional single-hop PRE schemes.

The conventional security requirement in unidirectional PRE schemes prevents the proxy from learning any information about the underlying plaintext or the private keys of the delegator and the delegatee from the re-encryption key. However, such a security notion is impractical in scenarios where the proxy cannot be assumed trusted, for example in open cloud systems [8, 15, 25]. A corrupted proxy can collude with malicious delegatees to transfer decryption rights to illegitimate third parties. Illegal delegation of decryption power can cause damage to the delegator in every possible manner such as unauthorised sharing of sensitive data and financial loss. This marks achieving *non-transferability* as a crucial property in PRE.

In [2], Ateniese et al. introduced the notion of non-transferability as "*The malicious proxy and a set of colluding delegatees cannot re-delegate decryption rights*". A PRE scheme is non-transferable when the colluding proxies and the delegatees cannot delegate decryption rights to malicious users without compromising privacy of the colluding delegatees. Note that the proxy and the delegatee can always decrypt ciphertexts of the delegator and forward the messages to unauthorised parties. But this would require the delegatee to remain an active online participant. Again, the delegatee can always send his private key to the malicious parties, which however is a potential threat to his security. The security goal here is to prevent the colluders to provide a secret value to the third parties that can be used to decrypt the delegator's ciphertexts, given the delegatee is *offline*. Hence, the only way to transfer decryption rights is to reveal the delegatee's private key. In this work, we study PRE in the light of non-transferability. We formalize the security notion of non-transferable PRE and propose an efficient PRE construction fitting the model under appropriate complexity assumptions.

1.1 Related Work

While numerous protocols achieving proxy re-encryption in varied models and assumptions have been proposed, only a few results attempted to provide non-transferable property as well. The security objective of non-transferability, defined in [1] by Ateniese et al. in 2005, is to prevent corrupt proxies colluding with malicious delegatees from transferring decryption rights of the delegator to illicit third parties. This problem was first addressed in [17] by Libert et al. in 2008, which also stated the difficulty in curbing such collusion practices. They further coined a new notion called *traceable proxy re-encryption (TPRE)* as a potential strategy towards partly achieving non-transferability. Misbehaving proxies can be traced with their technique by the delegator after a collusion takes place. They also proposed a CPA secure TPRE scheme and provided its security analysis in the standard model (without the random-oracle heuristic [3]), proven CPA-secure under Augment Decisional Bilinear Diffie-Hellman assumption (ADBDH). Note that, it is more desirable to prevent collusion rather than discouraging it by imposing such penalties on the colluders after an unauthorised transfer. In 2010, Wang et al. [25] proposed a non-transferable PRE scheme in the identity based setting (ID-PRE), where a centralised authority termed PKG (Private Key Generator) is entrusted with the generation of the re-encryption keys. Such a technique requires the PKG to remain an active online participant during the process of re-encryption key generation. This introduces the *key-despotism problem* since the *PKG* can itself generate re-encryption keys and delegate decryption rights to arbitrary parties. Likewise, it also adds the *key-escrow problem* since the *PKG* can decrypt both original and re-encrypted ciphertexts of all users within the system. Their scheme is proven secure in the random oracle model, based on the Bilinear Decisional Diffie Hellman assumption (BDDH) and the Bilinear Computational Diffie Hellman assumption (BCDH). Further on, a non-transferable certificateless PRE was proposed by He et al. in [13] that addresses the former problems associated with the involvement of PKG. However, their design involves multiple rounds of interactions among all the system entities (PKG, delegator, delegatee and proxy) during the re-encryption key generation process, for partial-key generations and key-validations which makes their scheme less practical. The security of their construction is in the random oracle model, based on the Truncated Decision Augmented Bilinear Diffie-Hellman Exponent Assumption (Truncated q-ABDHE). Hayashi et al. [12] introduced the idea of unforgeability of re-encryption keys against collusion attack (UFReKey-CA) as a partial solution to the non-transferability problem. Their proposed PRE scheme achieves UFReKey-CA where the colluders cannot generate re-encryption keys for malicious third parties, based on the hardness of variants of the Diffie-Hellman inversion problem in the standard model. Later Isshiki et al. [14] demonstrated a forgeability attack on the re-encryption keys generated by the scheme due to Hayashi et al [12]. Guo et al. [10] resorts to indistinguishability obfuscation ($i\mathcal{O}$), which is an extremely complex primitive, to achieve non-transferability in proxy re-encryption.

1.2 Contribution

Our contribution in this work is threefold. In 2005, Ateniese et al. [1] states *"achieving a proxy scheme that is non-transferable, in the sense that the only way for Bob to transfer offline decryption capabilities to Carol is to expose his own secret key, seems to be the main open problem left for proxy re-encryption"*. In IWSEC 2011, Hayashi et al. [12] proposed a formal definition of non-transferability in proxy re-encryption. In this work, we formalise the notion of non-transferability and introduce a security model to realize the equivalent, which extends the work in [12].

In 2015, Guo et al. [10] resolve the problem of non-transferability using indistinguishability obfuscation ($i\mathcal{O}$), which is a complex primitive and highly impractical. Our second major contribution lies in providing a non-transferable unidirectional single-hop PRE scheme based on bilinear maps and group operations. This makes our construction much more practical, and to the best of our knowledge, there are no known PRE schemes satisfying non-transferability in the Public Key Infrastructure (PKI) setting based on group theoretic operations. Our scheme makes the knowledge of secret key (KOSK) assumption as in [5], such that the adversary obtains the honest public keys and corrupted public-private key pairs from the challenger and cannot determine which parties are to be corrupted adaptively. Our construction meets chosen ciphertext security and non-transferability in the random oracle model, assuming the modified Decisional Bilinear Diffie-Hellman (m-DBDH) problem and the 1-weak Decisional Bilinear Diffie-Hellman Inversion (1-wDBDHI) problem.

In Pairing 2010, Wang et al.[25] proposed an identity-based uni-directional non-transferable PRE scheme in the random oracle model based on the hardness of the Bilinear Decisional Diffie-Hellman (BDDH) Problem, wherein a fully trusted PKG generates the re-encryption keys. In this work, we present an attack on the non-transferability of their scheme. We demonstrate that the colluding delegatees and proxy can successfully construct an illegal decryption function, which enables malicious third parties to decrypt ciphertexts of the delegator, without compromising the private keys of the delegatees.

2 Preliminaries

2.1 Bilinear Pairings

Let $\mathbb{G}_1$ be a cyclic additive group generated by P whose order is a prime q. Let $\mathbb{G}_2$ be a cyclic multiplicative group with the same order q. A bilinear map is an efficient mapping $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\rightarrow\mathbb{G}_2$ which is both: *(bilinear)* for all $P,Q\in\mathbb{G}_1$ and $a,b\in\mathbb{Z}_q^*$, $\hat{e}(aP,bQ)=\hat{e}(P,Q)^{ab}$; and *(non-degenerate)* if $\mathbb{G}_1=\langle P\rangle$, then $\hat{e}(P,P)\neq 1$.

2.2 Hardness Assumptions

In this section, we state the computational hardness assumptions related to bilinear pairings which we use to prove the security of our scheme. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be an admissible bilinear map, where $\mathbb{G}_1$, $\mathbb{G}_2$ has a prime order q.

Modified Decisional Bilinear Diffie-Hellman (m-DBDH) assumption [24] : The modified decisional bilinear diffie-hellman (m-DBDH) assumption is, given elements $\{P, aP, bP, cP, a^{-1}P, a^{-2}P\} \in \mathbb{G}_1$ and $T \in \mathbb{G}_2$, there exists no probabilistic polynomial time algorithm which can determine whether $T = \hat{e}(P,P)^{abc}$ or a random element from $\mathbb{G}_2$ with a non-negligible advantage, where P is a generator of $\mathbb{G}_1$ and $a, b, c \in_R \mathbb{Z}_q^*$. A probabilistic algorithm $\mathscr{A}$ has an advantage ε in solving the m-DBDH assumption if:

$$\Big| \big[\Pr(P, aP, bP, cP, a^{-1}P, a^{-2}P, e(P,P)^{abc})\big] - \big[\Pr(P, aP, bP, cP, a^{-1}P, a^{-2}P, T)\big] \Big| \leq \varepsilon$$

where the probability is over the random choices of $a, b, c \in \mathbb{Z}_q^*$, the random choice of $T \in \mathbb{G}_2$, the random choice of $P \in \mathbb{G}_1$ and the random bits of $\mathscr{A}$.

1-weak Decisional Bilinear Diffie-Hellman Inversion (1-wDBDHI) assumption [18] : The 1-weak decisional bilinear diffie-hellman inversion (1-wDBDHI) assumption is, given elements $\{P, \frac{1}{a}P, bP\} \in \mathbb{G}_1$ and $T \in \mathbb{G}_2$, there exists no probabilistic polynomial time algorithm which can determine whether $T = \hat{e}(P,P)^{ab}$ or a random element from $\mathbb{G}_2$ with a non-negligible advantage, where P is a generator of $\mathbb{G}_1$ and $a, b \in_R \mathbb{Z}_q^*$. A probabilistic algorithm $\mathscr{A}$ has an advantage ε in solving the 1-wDBDHI assumption if:

$$\Big| \big[\Pr(P, \frac{1}{a}P, bP, e(P,P)^{ab})\big] - \big[\Pr(P, \frac{1}{a}P, bP, T)\big] \Big| \leq \varepsilon$$

where the probability is over the random choices of $a, b \in \mathbb{Z}_q^*$, the random choice of $T \in \mathbb{G}_2$, the random choice of $P \in \mathbb{G}_1$ and the random bits of $\mathscr{A}$.

3 Definition and Security Model

In this section, we first review the syntactical definition and CCA security definition for PRE, followed by a new security definition to capture the notion of non-transferability in PRE.

3.1 Definition

A unidirectional single-hop PRE scheme [25] consists of the following algorithms:

- ***Setup***(κ): A probabilistic algorithm that takes security parameter κ as input and returns a set of public parameters *params* shared by all users of the system.

- ***KeyGen***$(params)$: A probabilistic algorithm run by user i, which takes as input *params*. It returns a public key and private key pair (pk_i, sk_i) of the user i.

- ***ReKeyGen***$(sk_i, pk_i, pk_j, params)$: A probabilistic algorithm run by delegator i, which takes as input the private key sk_i and public key pk_i of the delegator i, public key pk_j of a delegatee j and public parameter *params*. It returns a re-encryption key $RK_{i \rightarrow j}$.

- ***Encrypt***$(m, pk_i, params)$: A probabilistic algorithm run by the sender, which takes as input a plaintext $m \in \mathscr{M}$, a public key pk_i and $params$. It returns a ciphertext $\mathbb{C}$ corresponding to m which is allowed to be re-encrypted for another user. The ciphertext $\mathbb{C}$ generated is termed *second level ciphertext*.
- ***Re-Encrypt***$(\mathbb{C}, RK_{i\to j}, params)$: A probabilistic algorithm run by the proxy, which takes as input a second level ciphertext $\mathbb{C}$ encrypted under pk_i, a re-encryption key $RK_{i\to j}$ and $params$. It returns a ciphertext $\mathbb{D}$ encrypted under public key pk_j. The re-encrypted ciphertext $\mathbb{D}$ is termed *first level ciphertext*.
- ***Decrypt***$(\mathbb{C}, sk_i, params)$: A deterministic algorithm run by user i, which takes as input a second level ciphertext $\mathbb{C}$ encrypted under public key pk_i, its private key sk_i and $params$. It returns a plaintext m or the error symbol $\perp$ if the ciphertext is invalid.
- ***Re-Decrypt***$(\mathbb{D}, sk_j,$params): The re-decryption algorithm is a deterministic algorithm run by user j, which takes as input a first level ciphertext $\mathbb{D}$ re-encrypted under pk_j, the private key sk_j of user j and public parameters $params$. It returns a plaintext m or the error symbol $\perp$ if the ciphertext is invalid.

3.2 Correctness

For all public parameters $params \leftarrow Setup(\kappa)$, any public-private key pairs $(pk_i, sk_i) \leftarrow KeyGen(params)$, $(pk_j, sk_j) \leftarrow KeyGen(params)$ and all messages $m \in \mathscr{M}$:

$$\textbf{\textit{Decrypt}}(\textbf{\textit{Encrypt}}(m, pk_i, params), sk_i, params) = m,$$

$$\textbf{\textit{Re-Decrypt}}(\textbf{\textit{Re-Encrypt}}(\mathbb{C}, RK_{i\to j}, params), sk_j, params) = m,$$

where $\mathbb{C} \leftarrow$ ***Encrypt***$(m, pk_i, params)$ is a second level ciphertext.

3.3 Game Based Definition of Security

In this subsection, we first restate the standard CCA security notions for proxy re-encryption schemes. Since there exists two levels of ciphertexts namely the first and the second levels in a PRE scheme, it is essential to prove security of both levels as defined in [17]. As in [6], our static corruption model makes the knowledge of secret key *(KOSK)* assumption, where the adversary $\mathscr{A}$ cannot determine which parties will be compromised adaptively. It receives a set of honest public keys $pk_{i:i\in HU}$ and corrupted public-private key pairs $\{pk_i, sk_i\}_{i:i\in CU}$ from the challenger $\mathscr{C}$. The adversary $\mathscr{A}$ can adaptively query a list of oracles, to which the challenger $\mathscr{C}$ responds as defined below to simulate an environment running PRE for the adversary $\mathscr{A}$.

- **Re-encryption key generation oracle** $\mathscr{O}_{RK}(pk_i, pk_j)$: The challenger $\mathscr{C}$ runs $RK_{i\to j} \leftarrow$ **ReKeyGen** $(sk_i, pk_i, pk_j, params)$ and returns the key $RK_{i\to j}$ from user i to j to $\mathscr{A}$.
- **Re-encryption oracle** $\mathscr{O}_{RE}(\mathbb{C}, pk_i, pk_j)$: To re-encrypt a second-level ciphertext $\mathbb{C}$ towards user j, the challenger $\mathscr{C}$ runs $\mathbb{D} \leftarrow$ ***Re-Encrypt***$(\mathbb{C}, RK_{i\to j}, params)$, where $RK_{i\to j} \leftarrow$ ***ReKeyGen***$(sk_i, pk_i, pk_j, params)$ and returns the first level ciphertext $\mathbb{D}$.
- **Second level decryption oracle** $\mathscr{O}_{Dec}(\mathbb{C}, pk_i)$: To decrypt a second level ciphertext $\mathbb{C}$ encrypted under pk_i, the challenger $\mathscr{C}$ runs ***Decrypt***$(\mathbb{C}, sk_i, params)$ and returns the plaintext m or the error symbol $\perp$ if the ciphertext is invalid.

- **First level decryption oracle** $\mathcal{O}_{RD}(\mathbb{D}, pk_j)$: To decrypt a first level ciphertext $\mathbb{D}$ encrypted under sk_j, the challenger $\mathcal{C}$ runs ***Re-Decrypt***$(\mathbb{D}, sk_j, params)$ and returns the plaintext m or the error symbol $\perp$ if the ciphertext is invalid.

We next define two levels of semantic security under chosen ciphertext attack (CCA) for single-hop unidirectional PRE.

3.3.1 Second level ciphertext security.

Second level ciphertext security models the scenario where the adversary $\mathcal{A}$ is challenged with a second level ciphertext $\mathbb{C}^*$ encrypted under the targeted public key pk_{i^*}. Following is the description of the game template for Chosen Ciphertext Security for second level ciphertexts which we denote as $Exp_{A,2}^{IND-PRE-CCA}(\kappa)$:

1. **Setup:** The challenger $\mathcal{C}$ takes as input a security parameter κ and executes the ***Setup*** algorithm to generate public parameters *params*. It executes ***KeyGen*** algorithm to generate uncorrupted public-private key pairs $\{pk_i, sk_i\}_{i:i \in HU}$ and corrupted public-private key pairs $\{pk_i, sk_i\}_{i:i \in CU}$. It returns *params*, the uncorrupted public keys $\{pk_i\}_{i:i \in HU}$ and the corrupted public-private key pairs $\{pk_i, sk_i\}_{i:i \in CU}$ to the adversary $\mathcal{A}$.

2. **Phase 1:** $\mathcal{A}$ adaptively queries the re-encryption key generation, re-encryption, decryption and re-decryption oracles and $\mathcal{C}$ responds to the queries accordingly.

3. **Challenge:** $\mathcal{A}$ outputs two equal-length messages m_0,$m_1 \in \mathcal{M}$ and target public key pk_i^*, subject to below constraints to prevent $\mathcal{A}$ from decrypting challenge ciphertext trivially:

   – The target public key pk_i^* must be uncorrupted.
   – $\mathcal{A}$ must not have queried the re-encryption key generation oracle $\mathcal{O}_{RK}(pk_i^*, pk_j)$ beforehand, where key pk_j is corrupted. In such a case, $\mathcal{A}$ can trivially decrypt the challenge ciphertext by first re-encrypting it into a first level ciphertext under pk_j and further decrypting it with sk_j.

   On receiving $\{m_0, m_1\}$, $\mathcal{C}$ flips a coin $\psi \in \{0,1\}$, computes and returns the challenge ciphertext $\mathbb{C}^*$ as the second level encryption of m_ψ under pk_i^* to $\mathcal{A}$.

4. **Phase 2:** $\mathcal{A}$ issues queries as in Phase 1 obeying the following constraints.

   - $\mathcal{A}$ can query rekey generation oracle $\mathcal{O}_{RK}(pk_i^*, pk_j)$ only if pk_j is uncorrupted.
   - If $\mathcal{A}$ issues a re-encryption query $\mathcal{O}_{RE}(\mathbb{C}, pk_i, pk_j)$ such that the pk_j is corrupted, (pk_i,$\mathbb{C}$) should not be a challenge derivative (defined next) of (pk_i^*,$\mathbb{C}^*$).
   - $\mathcal{A}$ cannot issue a decryption query $\mathcal{O}_{Dec}$($\mathbb{C}$,pk_i) or a re-decryption query $\mathcal{O}_{RD}$($\mathbb{C}$,pk_i) if (pk_i,$\mathbb{C}$) is a *challenge derivative* (explained next) of (pk_i^*,$\mathbb{C}^*$).

   Definition 1. (Challenge Derivative for CCA Security). *The challenge derivative of* $(pk_i, \mathbb{C})$ *in the CCA setting is defined in [6] as shown below:*

   - *Reflexivity:* $(pk_i, \mathbb{C})$ *is a challenge derivative of its own.*
   - *Derivative by re-encryption key: if* $\mathbb{D} \leftarrow$ ***Re-Encrypt***$(\mathbb{C}, RK_{i \to j}, params)$, *where the re-encryption key* $RK_{i \to j} \leftarrow \mathcal{O}_{RK}(pk_i, pk_j)$, *then* $(pk_j, \mathbb{D})$ *is a challenge derivative of* $(pk_i, \mathbb{C})$.

- *Derivative by re-encryption: if* $\mathbb{D} \leftarrow \mathscr{O}_{RE}(\mathbb{C}, pk_i, pk_j)$*, then* $(pk_j, \mathbb{D})$ *is a challenge derivative of* $(pk_i, \mathbb{C})$.

5. **Guess:** Finally, $\mathscr{A}$ outputs a guess $\psi' \in \{0,1\}$.

CCA-Security Game Template $Exp_{A,2}^{IND-PRE-CCA}(\kappa)$

params $\leftarrow$ Setup(κ), $\{(pk_i, sk_i) \leftarrow$ KeyGen(params)$\}_{i \in CU}$;
$\{(pk_j, sk_j) \leftarrow$ KeyGen(params)$\}_{j \in HU}$, $(pk_i^*, sk_i^*) \leftarrow$ KeyGen(params);
$(m_0, m_1, \text{St}) \leftarrow \mathscr{A}^{\mathscr{O}_{RK}, \mathscr{O}_{RE}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}}(\{pk_i, sk_i\}_{i \in CU}, \{pk_j\}_{j \in HU})$;
$\psi \; \varepsilon_R \; \{0,1\}$, $\mathbb{C}^* \leftarrow$Encrypt(m_ψ, pk_i^*,params);
$\psi' \leftarrow \mathscr{A}^{\mathscr{O}_{RK}, \mathscr{O}_{RE}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}}(\mathbb{C}^*, St)$; //query constraints are described in Phase 2
if $\psi' = \psi$ return 1, else return 0

Game Template 1: The IND-PRE-CCA security game for second level ciphertexts of PRE schemes. Note that *St* is the state information maintained by $\mathscr{A}$.

Definition 2. Given a single-hop unidirectional PRE scheme, the advantage of any probabilistic polynomial time adversary $\mathscr{A}$ denoted by $Adv_{\mathscr{A}}$ in winning the IND-PRE-CCA game $Exp_{A,2}^{IND-PRE-CCA}(\kappa)$ given in Game Template 1 for second level ciphertext is shown as

$$Adv_{\mathscr{A},2}^{IND-PRE-CCA} := \left| Pr\left[Exp_{A,2}^{IND-PRE-CCA}(\kappa) \right] - \frac{1}{2} \right|$$

where the probability is taken over the coin tosses of the challenger $\mathscr{C}$ *and the adversary* $\mathscr{A}$.

The scheme is *IND-PRE-CCA* secure for the second level ciphertext against any t-time adversary $\mathscr{A}$ that makes atmost q_{RK} queries to the re-encryption key generation oracle, q_{RE} queries to the re-encryption oracle, q_{Dec} queries to the decryption oracle and q_{RD} queries to the re-decryption oracle, if the advantage of $\mathscr{A}$ is negligibly small:

$$Adv_{\mathscr{A},2}^{IND-PRE-CCA} \leq \varepsilon.$$

3.3.2 First level ciphertext security.

First level ciphertext security models the scenario where the adversary $\mathscr{A}$ is challenged with a first level ciphertext $\mathbb{D}^*$ re-encrypted from a delegator pk_i' to a the targeted delegatee pk_{i^*}. This security game allows the adversary $\mathscr{A}$ to obtain all possible re-encryption keys, since *single hop* PRE schemes prevents further re-encryption of first level ciphertexts. Consequently, no second-level decryption or re-encryption oracle is required by the adversary $\mathscr{A}$. Following is the description of the game template for Chosen Ciphertext Security for first level ciphertexts which we denote as $Exp_{A,1}^{IND-PRE-CCA}(\kappa)$:

1. **Setup:** Same as second level ciphertext security game.

2. **Phase 1:** $\mathscr{A}$ adaptively queries the re-encryption key generation, decryption and re-decryption oracles and challenger $\mathscr{C}$ responds to the queries accordingly.

3. **Challenge:** $\mathscr{A}$ outputs two equal-length messages m_0,$m_1 \in \mathscr{M}$, delegator's public key pk_i' and a target public key of delegatee pk_i^* such that pk_i^* must be uncorrupted. $\mathscr{C}$ flips a coin $\psi \in \{0,1\}$, computes and returns the challenge ciphertext $\mathbb{D}^*$ as the first level encryption of m_ψ under pk_i^* to the adversary $\mathscr{A}$.

4. **Phase 2:** $\mathscr{A}$ issues queries as in Phase 1 subject to the constraint that it cannot issue a decryption query $\mathscr{O}_{Dec}(\mathbb{D},pk_i)$ on the issued challenge ciphertext $\mathbb{D}^*$.

5. **Guess:** Finally, $\mathscr{A}$ outputs a guess $\psi' \in \{0,1\}$.

```
**CCA-Security Game Template** $Exp^{IND-PRE-CCA}_{A,1}(\kappa)$
params ← Setup($\kappa$), $\{(pk_i, sk_i) \leftarrow$ KeyGen(params)$\}_{i \in CU}$;
$\{(pk_j, sk_j) \leftarrow$ KeyGen(params)$\}_{j \in HU}$, $(pk_i^*, sk_i^*) \leftarrow$ KeyGen(params);
$(m_0, m_1, \text{St}) \leftarrow \mathscr{A}^{\mathscr{O}_{RK},\, \mathscr{O}_{RE}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}}(\{pk_i, sk_i\}_{i \in CU}, \{pk_j\}_{j \in HU})$;
$\psi \; \varepsilon_R \; \{0,1\}$, $\mathbb{D}^* \leftarrow$ReEncrypt($m_\psi, RK_{pk_{i'} \to pk_i^*}$,params);
$\psi' \leftarrow \mathscr{A}^{\mathscr{O}_{RK}, \mathscr{O}_{RE}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}}(D^*, St)$; //query constraints are described in Phase 2
if $\psi' = \psi$ return 1, else return 0
```

Game Template 2: The IND-PRE-CCA security game for first level ciphertexts of PRE schemes. Note that *St* is the state information maintained by $\mathscr{A}$.

Definition 3. Given a single-hop unidirectional PRE scheme, the advantage of any probabilistic polynomial-time adversary $\mathscr{A}$ denoted by $Adv_{\mathscr{A}}$ in winning the IND-PRE-CCA game $Exp^{IND-PRE-CCA}_{A,1}(\kappa)$ given in Game Template 2 for first level ciphertext is shown as

$$Adv^{IND-PRE-CCA}_{\mathscr{A},1} := \left| Pr\left[Exp^{IND-PRE-CCA}_{A,1}(\kappa)\right] - \frac{1}{2} \right|$$

where the probability is taken over the coin tosses of the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$.

The scheme is *IND-PRE-CCA* secure for the first level ciphertext against any t-time adversary $\mathscr{A}$ that makes atmost q_{RK} queries to the re-encryption key generation oracle to the q_{RD} queries to re-decryption oracle, if the advantage of $\mathscr{A}$ is negligibly small:

$$Adv^{IND-PRE-CCA}_{\mathscr{A},1} \leq \varepsilon.$$

3.4 Non-transferability

In [1], Ateniese et al. describes the notion of non-transferability as "*A proxy scheme is non-transferable when the only way for Bob (corrupted delegatee) to transfer offline decryption capabilities to Carol (malicious user) is to expose his own secret key*". Non-transferability is attained when ciphertexts of the delegator Alice can be decrypted by the malicious user Carol only with access to Bob's private keys and re-encryption key $RK_{\text{Alice}\to\text{Bob}}$, otherwise it obtains nothing useful. Our security definition of non-transferability follows from the definition of the same proposed by Hayashi et al. in [12]. We use the following subscripts: A to denote a target honest delegator, H to denote an honest user, B to denote a corrupted delegatee and J to denote a malicious user respectively. Following is the description of the game template for non-transferability which we denote as $Exp^{NT-PRE}_{\mathscr{P}}(\kappa)$:

1. **Setup:** Challenger $\mathscr{C}$ executes ***Setup*** algorithm to generate the public parameters *params*. It then executes the key generation algorithm to generate the set of public-private key pairs $\{pk_A, sk_A\}$, $\{pk_H, sk_H\}$, $\{pk_B, sk_B\}$ and $\{pk_J, sk_J\}$ corresponding to the public-private key pairs of the honest delegator pk_A, an honest user pk_H, a dishonest delegatee pk_B and a malicious third party pk_J. It returns *params*, public keys pk_A, pk_H, pk_B and public-private key pair $\{pk_J, sk_J\}$ to adversary $\mathscr{P}$.

2. **Phase 1:** $\mathscr{P}$ adaptively queries the re-encryption key generation, re-encryption, decryption and re-decryption oracles as described above in the CCA security game template in Section 3.3 and the challenger $\mathscr{C}$ responds to the queries accordingly. Additionally, the challenger also provides $\mathscr{P}$ with an oracle access to an illegal re-encrypion box $\mathscr{X}$ that transforms second-level ciphertexts of the honest delegator pk_A to the malicious party pk_J. The adversary $\mathscr{P}$ also supplies the description of arbitrary polynomial time computable functions f_i. The challenger simulates a function oracle $\mathscr{O}_{pf}(f_i)$ which computes $f_i(sk_B, RK_{A\to B})$, where sk_B is the private key corresponding to the corrupt delegatee pk_B, and returns the output to the adversary $\mathscr{P}$.

3. **Challenge:** Once $\mathscr{P}$ decides that Phase 1 is over, it outputs two equal-length messages m_0,$m_1 \in \mathscr{M}$. On receiving $\{m_0, m_1\}$, the challenger $\mathscr{C}$ flips a coin $\psi \in \{0,1\}$, computes and returns the challenge ciphertext $\mathbb{C}^*$ as the second level encryption of m_ψ under the public key pk_A of the delegator to the adversary $\mathscr{P}$.

4. **Phase 2:** $\mathscr{P}$ issues queries as in Phase 1 with the following constraints:
   - $\mathscr{P}$ cannot query the re-encryption key generation oracle $\mathscr{O}_{RK}(pk_A, pk_J)$.
   - If $\mathscr{P}$ cannot issue a re-encryption query to the illegal re-encryption box $\mathscr{O}_{\mathscr{X}}(\mathbb{C}^*)$.
   - $\mathscr{A}$ cannot issue a decryption query to $\mathscr{O}_{Dec}(\mathbb{C}^*,pk_A)$ or a re-decryption query to $\mathscr{O}_{RD}(\mathbb{D},pk_i)$ if $(pk_i,\mathbb{D})$ is a *challenge derivative* of $(pk_A,\mathbb{C}^*)$.

5. **Guess:** Finally, $\mathscr{P}$ outputs a guess $\psi' \in \{0,1\}$.

Definition 4. ***Non-transferability***. *Given a single-hop unidirectional PRE scheme, the advantage of any probabilistic polynomial-time adversary $\mathscr{P}$ denoted by $Adv_{\mathscr{P}}^{NT-PRE}$ in winning the NT-PRE game $Exp_A^{NT-PRE}(\kappa)$ given in Game Template 3 is shown as*

$$Adv_{\mathscr{P}}^{NT-PRE} := \left| Pr\left[Exp_P^{NT-PRE}(\kappa)\right] - \frac{1}{2} \right|$$

where the probability is taken over the coin tosses of the challenger $\mathscr{C}$ and the adversary $\mathscr{P}$.

The scheme is said to meet non-transferability against any t-time adversary $\mathscr{P}$ that makes atmost q_{RK} queries to the re-encryption key generation oracle, q_{pf} queries to the function oracle, $q_{\mathscr{X}}$ to the illegal re-encryption box, q_{RE} queries to the re-encryption oracle, q_{Dec} queries to the decryption oracle and q_{RD} queries to the re-decryption oracle, if the advantage of $\mathscr{P}$ in winning the NT-PRE game is negligible:

$$Adv_{\mathscr{P}}^{NT-PRE} \leq \varepsilon.$$

The definition in Game Template 3 states that, if the delegatee pk_B tries to construct an illegal re-encryption box $\mathscr{X}$ to re-delegate decryption rights of the target honest user pk_A towards a malicious user pk_J, then user pk_J can exploit X to compromise the decryption capabilities of the delegatee pk_B. Therefore, the colluders should compromise on the private keys of the delegatee in an attempt to generate a decryption-box to decrypt the delegator's ciphertext.

```
Our Non-transferability Game Template $Exp_A^{NT-PRE}(\kappa)$
params←Setup($\kappa$),$(pk_A, sk_B)$ ←Keygen(params),$(pk_B, sk_B)$ ←Keygen(params);
$(pk_H, sk_H)$ ←Keygen(params), $(pk_J, sk_J)$ ← Keygen(params);
$\{RK_{A\to B} \leftarrow$ ReKeyGen$(sk_A, pk_B)\}; \{RK_{H\to B} \leftarrow$ReKeyGen$(sk_H, pk_B)\}$;
$(m_0, m_1$,St$) \leftarrow \mathscr{P}^{\mathscr{O}_{RK}, \mathscr{O}_{RE}, \mathscr{O}_{pf}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}, \mathscr{O}_{\mathscr{X}}}(\{pk_A\}, \{pk_H\}, \{pk_B\}, \{pk_J, sk_J\})$;
$\psi\ \varepsilon_R\ \{0,1\}$, $\mathbb{C}^* \leftarrow$Encrypt$(m_\psi, pk_A$,params);
$\psi' \leftarrow \mathscr{P}^{\mathscr{O}_{RK}, \mathscr{O}_{RE}, \mathscr{O}_{Dec}, \mathscr{O}_{RD}, \mathscr{O}_{\mathscr{X}}}(\mathbb{C}^*, St)$; //query constraints are described in Phase 2
if $\psi' = \psi$ return 1, else return 0
```

Game Template 3: Our security game for non-transferability in PRE schemes. *St* is the state information maintained by the adversary $\mathscr{P}$.

4 Analysis of a CPA-Secure Non-Transferable Identity-based PRE Scheme by Wang et al. [25]

4.1 Review of the scheme

- **Setup(κ):** The trusted authority PKG runs the BDH parameter generator algorithm with input κ to generate two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order p and bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let g is a generator of $\mathbb{G}_1$ and $\alpha \in_R \mathbb{Z}_p^*$. The PKG computes $g_1 = g^\alpha \in \mathbb{G}_1$. It then picks $g_2, \eta \in_R \mathbb{G}_1$ where η is termed the re-encryption parameter. If chooses a hash function $H : \{0,1\}^l \to \mathbb{G}_1$, where l is a fixed positive parameter. The public parameters returned by the PKG are $params = \{\mathbb{G}_1, \mathbb{G}_2, p, \hat{e}, g, g_1, g_2, \eta, H\}$. The master secret key $msk = g_2^\alpha$ is kept with the PKG.

- **KeyGen($id, params$):** The PKG computes the private key for a user id_i as follows:

  - Pick $u \in \mathbb{Z}_p^*$. Compute the private key $sk_{id} = (d_0, d_1) = (g_2^\alpha H(id)^u, g^u)$.
  - The PKG can derive value u by computing $u = h_{msk}(id)$, used later during re-encryption key generation. Note that $h_{msk}()$ is a keyed hash function.
  - Upon receiving the private key sk_{id}, the user id validates the key using the following check:

$$\hat{e}(d_0, g) \stackrel{?}{=} \hat{e}(g_1, g_2) \cdot \hat{e}(H(id), d_1)$$

- **ReKeyGen($id, id', params$):** The re-encryption key is generated by the PKG and a user id as below:

  - The PKG returns a seed of the re-encryption key to delegator id:

$$\tilde{rk}_{id\to id'} = \left(\frac{H(id)}{H(id')}\right)^{u'}$$

    Here, u' is chosen by PKG during private key generation of user id'.
  - Delegator id selects $\delta \in \mathbb{Z}_p^*$ at random and computes the re-encryption key as:

$$rk_{id\to id'} = (rk_1, rk_2) = \left(\eta^\delta \left(\frac{H(id)}{H(id')}\right)^{u'}, g^\delta\right)$$

  - User id sends re-encryption key $rk_{id\to id'}$ to the proxy via a secure channel.

- **Encrypt**($m, id, params$)**:** To encrypt a message $m \in \mathbb{G}_2$, a user chooses $r \in_R \mathbb{Z}_p^*$ and computes the second level ciphertext $\mathbb{C}$ as below:

$$\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4) = (m.\hat{e}(g_1, g_2)^r, g^r, H(id)^r, \eta^r)$$

- **Re-Encrypt**($\mathbb{C}, RK_{id \to id'}, params$)**:** To re-encrypt a second-level ciphertext, the proxy does the following:
  - Check the well-formedness of the ciphertext using:
  $$\hat{e}(\mathbb{C}_2, \eta) \stackrel{?}{=} \hat{e}(\mathbb{C}_4, g)$$
  - If it holds, compute:
  $$\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3) = \left(\mathbb{C}_1 . \tfrac{\hat{e}(\mathbb{C}_4, rk_2)}{\hat{e}(\mathbb{C}_2, rk_1)}, \mathbb{C}_2, \mathbb{C}_3\right)$$
  - Return the first level ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3)$ to user id'.

- **Decrypt**($\mathbb{C}, sk_{id}, params$)**:** User id decrypts the second level ciphertext by computing:

$$m = \mathbb{C}_1 . \tfrac{\hat{e}(\mathbb{C}_3, d_1)}{\hat{e}(\mathbb{C}_2, d_0)}$$

- **Re-Decrypt**($\mathbb{D}, sk_{id'}$)**:** The delegatee id' decrypts the first level ciphertext by computing:

$$m = \mathbb{D}_1 . \tfrac{\hat{e}(\mathbb{D}_3, d'_1)}{\hat{e}(\mathbb{D}_2, d'_0)}$$

4.2 Attack on the Scheme

We revise the attack shown in [22] on the non-transferable property of the identity based PRE scheme due to Wang et al [25]. In the attack presented in [22], the inputs to the illegal decryption box leaks a re-encryption key component rk_2. We modify the attack such that none of the inputs to the illegal decryption box reveal the re-encryption key or private key of the delegatees, as per the definition of non-transferability. From definition Section 3.4, the adversary is allowed to obtain a pair of re-encryption key and private key of a corrupt delegatee id_j. Hence, the adversary queries for a re-encryption key $(rk_{id_i \to id_j}) = (rk_1, rk_2)$ and a private key of delegatee id_j $sk_{id_j} = (d_0, d_1) = (g_2^\alpha H(id_j)^{u_j}, g^{u_j})$. Next, on input of a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4)$, the adversary computes the decryption box as below:

1. Choose $\beta, \gamma \in_R \mathbb{Z}_p^*$.
2. Define $d' \stackrel{\Delta}{=} d_1 \cdot g^\beta = g^{u_j + \beta}$.
3. Compute a partial decryption key psk_{id_i}:
$psk_{id_i} = (rk_1 \cdot d_0 \cdot \eta^\gamma \cdot H(id_i)^\beta)$
$= \eta^{\delta+\gamma} \left(\frac{H(id_i)}{H(id_j)}\right)^{u_j} \cdot g_2^\alpha H(id_j)^{u_j} \cdot H(id_i)^\beta$
$= \eta^{\delta+\gamma} \cdot H(id_i)^{u_j+\beta} \cdot g_2^\alpha$.
(Computing a partial decryption key gives the adversary a function of the private key of the delegator id_i. This is used by the adversary to compute a decryption box that can be used to decrypt all ciphertexts encrypted towards the delegator id_i.)

4. Randomise the second re-encryption key component as below:

$$rk_2' = rk_2 \cdot g^{\gamma} = g^{\delta+\gamma}.$$

5. Construct decryption box for second level ciphertexts of delegator id_i as below:

$$m = \frac{\mathbb{C}_1}{\hat{e}(\mathbb{C}_2, psk_{id_i}) \cdot \hat{e}(\mathbb{C}_3, d')^{-1} \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1}}$$

Given a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4)$ of the delegator id_i, a malicious third party can extract the plaintext m as follows:

$$\begin{aligned}
&\frac{\mathbb{C}_1}{\hat{e}(\mathbb{C}_2, psk_{id_i}) \cdot \hat{e}(\mathbb{C}_3, d')^{-1} \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1}} \\
&= \frac{\mathbb{C}_1}{\hat{e}(\mathbb{C}_2, \eta^{\delta+\gamma} \cdot H(id_i)^{u_j+\beta} \cdot g_2^{\alpha}) \cdot \hat{e}(\mathbb{C}_3, d')^{-1} \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1}} \\
&= \frac{\mathbb{C}_1}{\hat{e}(g^r, g_2^{\alpha}) \cdot \hat{e}(g^r, \eta^{\delta+\gamma}) \cdot \hat{e}(g^r, H(id)^{u_j+\beta}) \cdot \hat{e}(\mathbb{C}_3, d')^{-1} \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1}} \\
&= \frac{\mathbb{C}_1}{\hat{e}(g^{\alpha}, g_2^{r}) \cdot \hat{e}(\eta^r, g^{\delta+\gamma}) \cdot \hat{e}(g^{u_j+\beta}, H(id_i)^r) \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1} \cdot \hat{e}(d', \mathbb{C}_3)^{-1}} \\
&= \frac{m \cdot \hat{e}(g_1, g_2)^r}{\hat{e}(g_1, g_2)^r \cdot \hat{e}(\mathbb{C}_4, rk_2') \cdot \hat{e}(d', \mathbb{C}_3) \cdot \hat{e}(\mathbb{C}_4, rk_2')^{-1} \cdot \hat{e}(d', \mathbb{C}_3)^{-1}} \\
&= m.
\end{aligned}$$

Note that neither the private key sk_{id_j} of the corrupt delegatee nor the re-encryption keys $RK_{id_i \to id_j}$ are compromised, while revealing the plaintext m encrypted as the second level ciphertext for delegator id_i to malicious users. Thus, this violates non-transferability of the PRE scheme.

5 A CCA-secure Non-transferable Scheme

5.1 Our Scheme

This section presents our non-transferable PRE construction [22], which consists of the following algorithms:

- ***Setup***(κ): $\mathbb{G}_1, \mathbb{G}_2$ are two groups of prime order q and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is an admissible bilinear map. Let P be a generator of the group $\mathbb{G}_1$. Pick $Q \in_R \mathbb{G}_1$. Compute $\alpha = \hat{e}(P, P)$. Choose five cryptographic hash functions: $\tilde{H} : \mathbb{G}_1 \to \mathbb{Z}_q^*, H_1 : \mathbb{G}_1^4 \to \mathbb{Z}_q^*, H_2 : \mathbb{G}_2 \to \{0,1\}^{l_m + l_\omega}, H_3 : \{0,1\}^{l_m + l_\omega} \to \mathbb{Z}_q^*, H_4 : \mathbb{G}_1^2 \times \{0,1\}^{l_m + l_\omega} \times \mathbb{G}_1 \to \mathbb{G}_1$, which are modelled as random oracles in the proof of security. Here, l_m and l_ω are security parameters determined by κ. The message space $\mathscr{M}$ is defined over $\{0,1\}^{l_m}$. The public parameters are params $:= \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P, Q, \tilde{H}, H_1, H_2, H_3, H_4, \alpha\}$.

- ***KeyGen***$(params)$: The public key and its corresponding private key of user i is generated as below:
 - Choose $x_i, y_i, z_i \leftarrow \mathbb{Z}_q^*$.
 - Set the private key $sk_i = (x_i, y_i, z_i)$.
 - Compute the public key $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_i P, y_i P, z_i P, y_i Q)$. Compute $h_i = H_1(pk_i)$.
 - Return the public and private key pair (pk_i, sk_i).

- ***ReKeyGen***$(sk_i, pk_i, pk_j, params)$: A re-encryption key from a user with public key $pk_i = (X_i, Y_i, Z_i, Q_i)$ and private key $sk_i = (x_i, y_i, z_i)$ towards a user with public key $pk_j = (X_j, Y_j, Z_j, Q_j)$is generated as below:
  - Choose $s, \delta, \beta \leftarrow \mathbb{Z}_q$. Compute $T = \frac{z_i+h_i}{\delta+\beta} \in \mathbb{Z}_q^*$.
  - Compute two re-encryption key components R and S:

$$\begin{aligned} R &= {x_i}^{-1}(\delta Y_j + sP) + {x_i}^{-1}\tilde{H}(X_j)Q \\ &= {x_i}^{-1}(\delta y_j + s)P + {x_i}^{-1}\tilde{H}(X_j)Q \in \mathbb{G}_1. \\ S &= {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j \\ &= {y_i}^{-1}(\beta y_j - s)P + {y_i}^{-1}Q_j \in \mathbb{G}_1. \end{aligned}$$

  - Return the re-encryption key $RK_{i\to j} = (R, S, T)$.

- ***Encrypt***$(m, pk_i, params)$: The second level ciphertext for a message $m \in \mathscr{M}$ under a public key $pk_i = (X_i, Y_i, Z_i, Q_i)$ is generated as below:
  - Pick $\omega \in_R \{0,1\}^{l_\omega}$.
  - Set $r = H_3(m, \omega) \in Z_q^*$.
  - Compute $\mathbb{C}_1 = rX_i \in \mathbb{G}_1$, $\mathbb{C}_2 = rY_i \in \mathbb{G}_1$.
  - Compute $\mathbb{C}_3 = (m||\omega) \oplus H_2(\hat{e}(Z_i + h_iP, P)^r) = (m||\omega) \oplus H_2(\hat{e}(P,P)^{(z_i+h_i)r})$.
  - Compute $\mathbb{C}_4 = r \cdot H_4(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5) \in \mathbb{G}_1$.
  - Compute $\mathbb{C}_5 = r \cdot Q \in \mathbb{G}_1$.
  - Return second level ciphertext $\mathbb{C}$= $(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$.

- ***Re-Encrypt***$(\mathbb{C}, RK_{i\to j}, params)$: A second level ciphertext $\mathbb{C}$= $(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ is re-encrypted using a re-encryption key $RK_{i\to j} = (R, S, T)$ as below:
  - Check well-formedness of ciphertext $\mathbb{C}$ by testing if:

$$\hat{e}(\mathbb{C}_4, X_i) \overset{?}{=} \hat{e}(H_4(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5), \mathbb{C}_1) \tag{1}$$

$$\hat{e}(X_i + Y_i, \mathbb{C}_5) \overset{?}{=} \hat{e}(\mathbb{C}_1 + \mathbb{C}_2, Q) \tag{2}$$

  If either of the above check fails, return $\perp$. Else compute

$$\mathbb{D}_1 = \left[\frac{\hat{e}(\mathbb{C}_1, R) \cdot \hat{e}(\mathbb{C}_2, S)}{\hat{e}(\tilde{H}(X_j) \cdot P, \mathbb{C}_5) \cdot \hat{e}(Y_j, \mathbb{C}_5)}\right]^T$$

$$= \hat{e}(P,P)^{(z_i+h_i)ry_j} \in \mathbb{G}_2. \tag{3}$$

  - Set $\mathbb{D}_2 = \mathbb{C}_3$ and $\mathbb{D}_3 = \mathbb{C}_5$.
  - Return $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3)$ as the first level ciphertext.

- ***Decrypt***$(\mathbb{C}, sk_i, params)$: To decrypt a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ using private key sk_i, first check if conditions (1) and (2) hold. If they fail, return $\perp$, otherwise compute

$$(m||\omega) = H_2(\hat{e}((\mathbb{C}_1 + \mathbb{C}_2, \frac{1}{(x_i + y_i)}P)^{(z_i+h_i)}) \oplus \mathbb{C}_3 \tag{4}$$

Remark 1. *Conditions* (1) *and* (2) *allow for the public verifiability of ciphertext* $\mathbb{C}$*, following which the plaintext* $(m||\omega)$ *is recovered. As a next step, it is sufficient to verify any one condition from* (6) *to* (9) *in* ***Verify*** *algorithm.*

Remark 2. *To avoid checking conditions* (1) *and* (2) *which incur heavy computation cost from pairing operation as indicated in Table 2, on extracting* $(m||\omega)$*, well-formedness of* $\mathbb{C}$ *can be checked using* ***Verify***$(pk_i,(m||\omega),\mathbb{C},params) = valid$*. If check fails, return* $\perp$*, else return* $(m||\omega)$*.*

- ***Re-Decrypt***$(\mathbb{D},sk_j,params)$: To decrypt first level ciphertext $\mathbb{D} =(\mathbb{D}_1,\mathbb{D}_2,\mathbb{D}_3)$ using private key sk_j, compute:

$$(m||\omega) = H_2(\mathbb{D}_1^{y_j^{-1}}) \oplus \mathbb{D}_2. \tag{5}$$

- ***Verify***$(pk_i,m||\omega,\mathbb{C},params)$: To check the well-formedness of a second level ciphertext $\mathbb{C} = (\mathbb{C}_1,\mathbb{C}_2,\mathbb{C}_3,\mathbb{C}_4,\mathbb{C}_5)$, which is an encryption of message $(m||\omega)$ towards a public key pk_i, compute $r = H_3(m||\omega)$ and check if all the following conditions $(6)-(9)$ hold:

$$\mathbb{C}_1 \overset{?}{=} r\cdot X_i \tag{6}$$

$$\mathbb{C}_2 \overset{?}{=} r\cdot Y_i \tag{7}$$

$$\mathbb{C}_4 \overset{?}{=} r\cdot H_4(\mathbb{C}_1,\mathbb{C}_2,\mathbb{C}_3,\mathbb{C}_5) \tag{8}$$

$$\mathbb{C}_5 \overset{?}{=} r\cdot Q \tag{9}$$

If all conditions satisfy, return $valid$ else return $invalid$.

Remark 3. *In order to re-delegate decryption rights to an illegal user in our scheme, the colluding delegatee with the proxy must construct the decryption box* $(\mathbb{D}_1' \oplus \mathbb{C}_3)$ *by defining* $\mathbb{D}_1' \overset{\Delta}{=} \mathbb{D}_1^{y_j^{-1}} = e(P,P)^{(z_i+h_i)r\cdot y_j\cdot y_j^{-1}} = e(P,P)^{(z_i+h_i)r}$*. Now any malicious third party can decrypt the second level ciphertext* $\mathbb{C}$ *of the delegator by computing* $\mathbb{D}_1' \oplus \mathbb{C}_3$*. However, the malicious party can succeed in decryption only when the delegatee shares with the malicious user his private key component* y_j *explicitly as* y_j^{-1} *must be used to exponentiate* $\mathbb{D}_1$ *to compute* $\mathbb{D}_1'$ *and further extract* $(m||\omega)$*. Note that the value of* $\mathbb{D}_1$ *changes for every delegation due to a fresh random element* $\omega \in \mathbb{Z}_q^*$ *being used for every encryption. The colluders cannot compute* $\mathbb{D}^{y_j^{-1}}$ *offline. Hence, the delegatee must expose his private key for illegal decryption transference to a third party. The formal proof of non-transferability is given in Section 5.3 of the paper.*

5.2 Correctness

- Correctness of equation (1):

$$\begin{aligned} RHS &= \hat{e}(H_4(C_1,C_2,C_3,C_5),C_1) \\ &= \hat{e}(H_4(C_1,C_2,C_3,C_5),rX_i) \\ &= \hat{e}(r\cdot H_4(C_1,C_2,C_3,C_5),X_i) \\ &= LHS. \end{aligned}$$

- Correctness of equation (2):

$$\begin{aligned} LHS &= \hat{e}(X_i + Y_i, C_5) \\ &= \hat{e}(X_i + Y_i, r \cdot Q) \\ &= \hat{e}(r(X_i + Y_i), Q) \\ &= RHS. \end{aligned}$$

- Correctness of equation (3):

$$\begin{aligned} \mathbb{D}_1 &= \left[\frac{\hat{e}(\mathbb{C}_1, R) \cdot \hat{e}(\mathbb{C}_2, S)}{\hat{e}(\tilde{H}(X_j) \cdot P, \mathbb{C}_5) \cdot \hat{e}(Y_j, \mathbb{C}_5)}\right]^T \\ &= \left[\frac{\hat{e}\big(rX_i, {x_i}^{-1}(\delta Y_j + sP) + {x_i}^{-1}\tilde{H}(X_j) \cdot Q\big) \cdot \hat{e}\big(rY_i, {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j\big)}{\hat{e}(\tilde{H}(X_j) \cdot P, \mathbb{C}_5) \cdot \hat{e}(Y_j, \mathbb{C}_5)}\right]^T \\ &= \left[\frac{\hat{e}(rP, y_j(\delta + \beta)P) \cdot \hat{e}(rP, \tilde{H}(X_j) \cdot Q) \cdot \hat{e}(rP, y_j Q)}{\hat{e}(\tilde{H}(X_j) \cdot P, \mathbb{C}_5) \cdot \hat{e}(Y_j, \mathbb{C}_5)}\right]^T \\ &= \left[\hat{e}(P,P)^{ry_j(\delta+\beta)}\right]^{\frac{z_i + h_i}{\delta + \beta}} \\ &= \hat{e}(P,P)^{ry_j(z_i + h_i)} \end{aligned}$$

- Consistency between *Encrypt* and *Decrypt*:
  From equation (4),

$$\begin{aligned} RHS &= H_2\Big(\hat{e}\big(\mathbb{C}_1 + \mathbb{C}_2, \frac{1}{(x_i + y_i)} P\big)^{(z_i + h_i)}\Big) \oplus \mathbb{C}_3 \\ &= H_2\big(\hat{e}(P,P)^{(z_i + h_i)r}\big) \oplus (m||\omega) \oplus H_2\big(\hat{e}(P,P)^{(z_i + h_i)r}\big) \\ &= (m||\omega) \\ &= LHS. \end{aligned}$$

- Consistency between *Re-Encrypt* and *Re-Decrypt*:
  From equations (3) and (5),

$$\begin{aligned} RHS &= H_2\big(\mathbb{D}_1^{y_j^{-1}}\big) \oplus \mathbb{D}_2 \\ &= H_2\big(\hat{e}(P,P)^{(z_i + h_i) r y_j y_j^{-1}}\big) \oplus (m||\omega) \oplus H_2(\hat{e}(Z_i + h_i P, P))^r) \\ &= (m||\omega) \\ &= LHS. \end{aligned}$$

5.3 Security Proof

Proof of Non-transferability:

Theorem 1. *Our unidirectional single-hop proxy re-encryption scheme meets non-transferability under the variant of the m-DBDH assumption. If there exists an adversary $\mathscr{P}$ that has an advantage ε in breaking the non-transferability of the proposed scheme in time t, then we can construct a challenger $\mathscr{C}$ who breaks the hard problem instance with an advantage ε' within time t' where:*

$$\varepsilon' \geq \varepsilon,$$

$$t^* \leq t + (q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{pf} + q_{RE} + q_{Dec} \\ + q_{RD})O(1) + (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} + 2q_{\mathscr{X}} \\ + 2q_{Dec} + q_{ReDec})t_{et} + (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp},$$

where t_e, t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation.

Proof. If an adversary $\mathscr{P}$ with given access to the random oracles $\tilde{H}, H_1, H_2, H_3, H_4$ breaks the non-transferability of our scheme, we can construct an algorithm $\mathscr{C}$ who, given $(P, aP, a^{-1}P, a^{-2}P, bP, cP)$ breaks the variant of the m-DBDH assumption in $\mathbb{G}_1$. We use the following subscripts A, H, B, J to denote a target honest delegator, an honest user, a corrupted delegatee and a malicious user respectively. In our security game, the environment for $\mathscr{P}$ is simulated as follows. Note that $\mathscr{C}$ sets $Q = \frac{1}{a}P$.

- **Key Generation:** $\mathscr{C}$ generates the keys as follows:
  -*Corrupted Key Generation:* Pick $x_i, y_i, z_i \in \mathbb{Z}_q^*$, and set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding $sk_i = (x_i, y_i, z_i)$. Store the tuple $\langle pk_i, sk_i, c_i = 0\rangle$ to the list L_{key}. Corrupted delegatee keys pk_B and malicious user keys pk_J are generated in this manner.
  -*Uncorrupted Key Generation:*
  ○ For the honest delegator, pick $\bar{x_A}, \bar{y_A}, \bar{z_A} \in \mathbb{Z}_q^*$, set $pk_A = (X_A, Y_A, Z_A, Q_A) = (x_AP, y_AP, z_AP, y_AQ)$ and the corresponding private key $sk_A = (x_A, y_A, z_A) = \left(\frac{\bar{x_A}}{a}, \frac{\bar{y_A}}{a}, c\bar{z_A}\right)$. Store the tuple $\langle pk_A, sk_A, c_A = 1\rangle$ to the list L_{key}.
  ○ For all other honest users, pick $x_i, \bar{y_i}, z_i \in \mathbb{Z}_q^*$, set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding private key $sk_i = (x_i, y_i, z_i) = \left(x_i, \frac{\bar{y_i}}{a}, z_i\right)$. Store the tuple $\langle pk_i, sk_i, c_i = 1\rangle$ to the list L_{key}.

- **Re-encryption Key Generation:** Inorder to generate the re-encryption keys $RK_{i\to j}$ from user i to user j, $\mathscr{C}$ constructs the different components R, S, T of $RK_{i\to j}$ according to the different cases :

  ○ If $(c_i = 1 \wedge pk_i = pk_A)$ and $(c_j = 0 \wedge pk_j = pk_J)$: output "failure" and abort the simulation.

  ○ If $(c_i = 1 \wedge pk_i = pk_A)$ and $(c_j = 1 \vee pk_j = pk_B)$:

  1. Pick $T, \delta, \bar{s} \in_R \mathbb{Z}_q^*$.
  2. From definition, $\beta = \frac{z_A + h_A}{T} - \delta$.
  3. Set $s = \frac{\bar{s}}{a}$ and compute

$$R = \bar{x_A}^{-1}(\bar{y_j}\delta + \bar{s})P + \bar{x_A}^{-1}\tilde{H}(X_j)P$$
$$S = \frac{\bar{y_j}\bar{z_A}}{\bar{y_A}T} \cdot cP + \frac{\bar{y_j}}{\bar{y_A}T}(h_A - \delta T)P - \bar{y_A}^{-1}\bar{s}P + \frac{\bar{y_j}}{\bar{y_A}}Q$$

  Observe that the re-encryption key components R and S computed is identically distributed as the components generated by the ***ReKeyGen*** algorithm given in the construction. Infact, we have:

$$\begin{aligned}
R &= \bar{x_A}^{-1}(\bar{y_j}\delta+\bar{s})P+\bar{x_A}^{-1}\tilde{H}(X_j)P \\
&= \Big(\frac{\bar{x_A}}{a}\Big)^{-1}\Big(\frac{\bar{y_j}}{a}\delta+\frac{\bar{s}}{a}\Big)P+\Big(\frac{\bar{x_A}}{a}\Big)^{-1}\tilde{H}(X_j)\cdot a^{-1}P \\
&= x_A{}^{-1}(\delta Y_j+sP)+x_A{}^{-1}\tilde{H}(X_j)Q \\
S &= \frac{\bar{y_j}\bar{z_A}}{\bar{y_A}T}\cdot cP+\frac{\bar{y_j}}{\bar{y_A}T}(h_A-\delta T)P-\bar{y_A}^{-1}\bar{s}P+\frac{\bar{y_j}}{\bar{y_A}}Q \\
&= \bar{y_A}^{-1}\bar{y_j}\Big(\frac{c\bar{z_A}+h_A}{T}-\delta\Big)P-\bar{y_A}^{-1}\bar{s}P+\bar{y_A}^{-1}\bar{y_j}Q \\
&= \Big(\frac{\bar{y_A}}{a}\Big)^{-1}\Big(\frac{\bar{y_j}}{a}\beta-\frac{\bar{s}}{a}\Big)P+\Big(\frac{\bar{y_A}}{a}\Big)^{-1}\frac{\bar{y_j}}{a}Q \\
&= y_A{}^{-1}(\beta Y_j-sP)+y_A{}^{-1}Q_j
\end{aligned}$$

4. Return $RK_{i\to j}=(R,S,T)$. Note that the computed value of the re-encryption key matches with the original distribution of computed re-encrypted key given in the scheme.

○ Else, choose $\delta,\beta,\bar{s}\in_R \mathbb{Z}_q^*$, set $s=\frac{\bar{s}}{a}$ and compute $T=\frac{z_i+h_i}{\delta+\beta}$ for each user i. Compute the other re-encryption key components for the following cases:

– If $(c_i=0)$ and $(c_j=1)$, compute:

$$\begin{aligned}
R &= x_i{}^{-1}(\bar{y_j}\delta+\bar{s})a^{-1}P+x_i{}^{-1}\tilde{H}(X_j)Q \\
S &= y_i{}^{-1}(\bar{y_j}\beta-\bar{s})a^{-1}P+y_i{}^{-1}\bar{y_j}a^{-2}P
\end{aligned}$$

Observe that the re-encryption key components R and S computed is identically distributed as the components generated by the ***ReKeyGen*** algorithm given in the construction. Infact, we have:

$$\begin{aligned}
R &= x_i{}^{-1}(\bar{y_j}\delta+\bar{s})a^{-1}P+x_i{}^{-1}\tilde{H}(X_j)Q \\
&= x_i{}^{-1}\big(\frac{\bar{y_j}}{a}\delta+\frac{\bar{s}}{a}\big)P+x_i{}^{-1}\tilde{H}(X_j)Q \\
&= x_i{}^{-1}(\delta Y_j+sP)+x_i{}^{-1}\tilde{H}(X_j)Q \\
S &= y_i{}^{-1}(\bar{y_j}\beta-\bar{s})a^{-1}P+y_i{}^{-1}\bar{y_j}a^{-2}P \\
&= y_i{}^{-1}\big(\frac{\bar{y_j}}{a}\beta-\frac{\bar{s}}{a}\big)P+y_i{}^{-1}\frac{\bar{y_j}}{a}a^{-1}P \\
&= y_i{}^{-1}(\beta Y_j-sP)+y_i{}^{-1}Q_j
\end{aligned}$$

– If $(c_i=1\wedge pk_i\neq pk_A)$ and $(c_j=0)$, compute:

$$\begin{aligned}
R &= x_i{}^{-1}(\delta y_j+\bar{s})P+x_i{}^{-1}\tilde{H}(X_j)Q \\
&= x_i{}^{-1}(\delta Y_j+\bar{s}P)+x_i{}^{-1}\tilde{H}(X_j)Q \\
S &= \bar{y_i}^{-1}(y_j\beta-\bar{s})aP+\bar{y_i}^{-1}y_jP
\end{aligned}$$

Observe that the re-encryption key component S computed is identically distributed as the component generated by the ***ReKeyGen*** algorithm given in the construction. Infact, we have:

$$\begin{aligned}
S &= \bar{y_i}^{-1}(y_j\beta-\bar{s})aP+\bar{y_i}^{-1}y_jP \\
&= \Big(\frac{\bar{y_i}}{a}\Big)^{-1}(y_j\beta-\bar{s})P+\Big(\frac{\bar{y_i}}{a}\Big)^{-1}y_ja^{-1}P \\
&= y_i{}^{-1}(\beta Y_j-\bar{s}P)+y_i{}^{-1}Q_j
\end{aligned}$$

– If $(c_i = 1 \wedge pk_i \neq pk_A)$ and $(c_j = 1)$, compute:

$$R = {x_i}^{-1}(\bar{y_j}\delta + \bar{s})a^{-1}P + {x_i}^{-1}\tilde{H}(X_j)Q$$
$$S = \bar{y_i}^{-1}(\bar{y_j}\beta - \bar{s})P + \bar{y_i}^{-1}\bar{y_j}Q$$

Observe that the re-encryption key components R and S computed is identically distributed as the component generated by the ***ReKeyGen*** algorithm given in the construction. Infact, we have:

$$\begin{aligned} R &= {x_i}^{-1}(\bar{y_j}\delta + \bar{s})a^{-1}P + {x_i}^{-1}\tilde{H}(X_j)Q \\ &= {x_i}^{-1}(\frac{\bar{y_j}}{a}\delta + \frac{\bar{s}}{a})P + {x_i}^{-1}\tilde{H}(X_j)Q \\ &= {x_i}^{-1}(\delta Y_j + sP) + {x_i}^{-1}\tilde{H}(X_j)Q \\ S &= \bar{y_i}^{-1}(\bar{y_j}\beta - \bar{s})P + \bar{y_i}^{-1}\bar{y_j}Q \\ &= \left(\frac{\bar{y_i}}{a}\right)^{-1}(\frac{\bar{y_j}}{a}\beta - \frac{\bar{s}}{a})P + \frac{\bar{y_j}/a}{\bar{y_i}/a}Q \\ &= {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j \end{aligned}$$

∘ Return $RK_{i \to j} = (R, S, T)$, and note that the re-encryption keys in all the cases are identically distributed as the real re-encryption keys from the construction.

- **Phase 1:** $\mathscr{C}$ interacts with $\mathscr{A}$ in the following ways:
  - *Oracle Queries*: $\mathscr{C}$ responds to the oracle queries as below:
    - *$\tilde{H}(X_i)$ Oracle*: $\mathscr{C}$ responds to the queries of $\mathscr{P}$ by maintaining a list $L_{\tilde{H}}$ with tuples of the form $\langle X_i \in \mathbb{G}_1, k \in \mathbb{Z}_q^* \rangle$. If a tuple of the form $\langle X_i, k \rangle$ already exists in $L_{\tilde{H}}$, retrieve and return the value k. Else, choose $k \leftarrow \mathbb{Z}_q^*$, set $\tilde{H}(X_i) = k$, store the tuple $\langle X_i, k \rangle$ in $L_{\tilde{H}}$ and return k.
    - *$H_1(pk_i)$ Oracle*: $\mathscr{C}$ maintains a list L_{H_1} with tuples of the form $\langle X_i \in \mathbb{G}_1, h_i \in \mathbb{Z}_q^* \rangle$. If the tuple $\langle X_i, h_i \rangle$ already exists in L_{H_1}, retrieve and return the value h_i. Else, choose $h_i \leftarrow \mathbb{Z}_q^*$, set $H_1(\mathrm{pk}_i) = h_i$, store the tuple $\langle X_i, h_i \rangle$ to L_{H_1} and return h_i.
    - *$H_2(R)$ Oracle*: To respond to the queries of $\mathscr{P}$, $\mathscr{C}$ maintains a list L_{H_2} with tuples of the form $\langle R \in \mathbb{G}_2, \rho \in \{0,1\}^{l_m + l_\omega} \rangle$. If the tuple $\langle R, \rho \rangle$ already exists in L_{H_2}, retrieve and return value ρ. Else, choose $\rho \leftarrow \mathbb{Z}_q^*$, set $H_2(R) = \rho$, store $\langle R, \rho \rangle$ to L_{H_2} and return ρ.
    - *$H_3(m, \omega)$ Oracle*: $\mathscr{C}$ maintains a list L_{H_3} with tuples of the form $\langle m, \omega, r \rangle$. If the tuple $\langle m, \omega, r \rangle$ already exists in L_{H_3}, retrieve and return the value r. Else, choose $r \leftarrow \mathbb{Z}_q^*$, set $H_3(m, \omega) = r$, store the tuple $\langle m, \omega, r \rangle$ to L_{H_3} and return r.
    - *$H_4(pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ Oracle*: To respond to the queries of $\mathscr{P}$, $\mathscr{C}$ maintains a list L_{H_4} with tuples of the form $\langle pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5, h'_i \rangle$. If the tuple $\langle pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5, h'_i \rangle$ already exists in L_{H_4}, retrieve and return the value h'_i. Else, choose $t \leftarrow \mathbb{Z}_q^*$, set $h'_i = t \cdot cP$, store the tuple $\langle pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5, h'_i \rangle$ to L_{H_4} and return h'_i.
  - *Function Queries $\mathscr{O}_{pf}(f_i)$:* The adversary $\mathscr{P}$ supplies the description of arbitrary polynomial time computable functions f_i to the challenger. The challenger computes $f_i(sk_B, RK_{A \to B})$, where sk_B is the private key corresponding to the corrupt delegatee pk_B, and returns the output to $\mathscr{P}$.
  - *Re-encryption Queries ($\mathscr{O}_{RE}(\mathbb{C}, pk_i, pk_j)$):* Given a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ and public keys pk_i and pk_j as inputs, first check if either of the conditions (1) or (2)

holds. If they do not hold, output *invalid*. Otherwise, retrieve the key tuples $\langle pk_i, x_i, y_i, z_i, c_i\rangle$, $\langle pk_j, x_j, y_j, z_j, c_j\rangle$ from the list L_{key} and consider the possibilities:

- If $(c_i = 1 \wedge pk_i = pk_A)$ and $j \in CU$, retrieve the value of $H_4(pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ to get h'_i, where $h'_i = t \cdot cP$ for some known $t \in_R \mathbb{Z}_q^*$, and compute:

$$\begin{aligned}\mathbb{D}_1 &= \hat{e}(\mathbb{C}_5, aP)^{y_j h_i} \cdot \hat{e}(t^{-1}\mathbb{C}_4, Y_j)^{\bar{z}_i}\\ &= \hat{e}(r\frac{1}{a}P, aP)^{y_j h_i} \cdot \hat{e}(t^{-1}rtcP, y_jP)^{\bar{z}_i}\\ &= \hat{e}(P,P)^{ry_j(z_i+h_i)}\end{aligned}$$

  Set $\mathbb{D}_2 = \mathbb{C}_3, \mathbb{D}_3 = \mathbb{C}_5$. Return the second level ciphertext $\mathbb{D}$.

- For all other cases, run the re-encryption key generation query $\mathscr{O}_{RK}(pk_i, pk_j)$ to obtain $RK_{i\to j}$. Run *Re-Encrypt*$(\mathbb{C}, RK_{i\to j})$ and return the second level ciphertext $\mathbb{D}$.

- *Illegal re-encryption box* $\mathscr{O}_{\mathscr{X}}(\mathbb{C})$: To obtain the re-encryption of a ciphertext $\mathbb{C}$ towards the malicious user pk_J, retrieve the value of $H_4(pk_A, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ to get h'_A, where $h'_A = t \cdot cP$ for some known $t \in_R \mathbb{Z}_q^*$, and compute:

$$\begin{aligned}\mathbb{D}_1 &= \hat{e}(\mathbb{C}_5, aP)^{y_J h_A} \cdot \hat{e}(t^{-1}\mathbb{C}_4, Y_J)^{\bar{z}_A}\\ &= \hat{e}(r\frac{1}{a}P, aP)^{y_J h_A} \cdot \hat{e}(t^{-1}rtcP, y_JP)^{\bar{z}_A}\\ &= \hat{e}(P,P)^{ry_J h_A} \cdot \hat{e}(P,P)^{y_J rc\bar{z}_A}\\ &= \hat{e}(P,P)^{ry_J(z_A+h_A)}\end{aligned}$$

  Set $\mathbb{D}_2 = \mathbb{C}_3, \mathbb{D}_3 = \mathbb{C}_5$. Return the second level ciphertext $\mathbb{D}$.

- *Second Level Decryption Queries* $(\mathscr{O}_{Dec}(\mathbb{C}, pk_i))$: Given a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ and a public key pk_i as input, first check if either of the conditions (1) or (2) holds. If it does not hold, output *invalid*. Otherwise, retrieve the value of $H_4(pk_{i^*}, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ to get h'_i, where $h'_i = t \cdot cP$ for some known $t \in_R \mathbb{Z}_q^*$. Define the value:

$$\begin{aligned}\tilde{\alpha} &= \hat{e}(t^{-1} \cdot \mathbb{C}_4, \bar{z}_iP) \cdot \hat{e}(\mathbb{C}_1, aP)^{\bar{x}_i^{-1} \cdot h_i}\\ &= \hat{e}(t^{-1}rtcP, \bar{z}_iP) \cdot \hat{e}(\frac{\bar{x}_i}{a}rP, aP)^{\bar{x}_i^{-1} \cdot h_i}\\ &= \hat{e}(P,P)^{rc\bar{z}_i} \cdot \hat{e}(P,P)^{rh_i}\\ &= \hat{e}(P,P)^{r(z_i+h_i)}\end{aligned}$$

  Note that $\tilde{\alpha} \oplus \mathbb{C}_3 = (m||\omega)$. Now verify the ciphertext components:

  - Compute $r = H_3(m, w)$.
  - Verify if $\mathbb{C}_1 \stackrel{?}{=} r \cdot X_i$, $\mathbb{C}_2 \stackrel{?}{=} r \cdot Y_i$, $\mathbb{C}_3 \stackrel{?}{=} (m||w) \oplus \tilde{\alpha}$, $\mathbb{C}_4 \stackrel{?}{=} r \cdot H_4(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ and $\mathbb{C}_5 \stackrel{?}{=} r \cdot Q$.

  If all the checks hold, return m to $\mathscr{A}$, else return $\perp$. For a decryption query with pk_i such that $c_i = 1$, re-encrypt $\mathbb{C}$ under the public key pk_j of a user $j \in CU$ and decrypt trivially.

- *Re-Decryption Queries* $(\mathscr{O}_{RD}(D, pk_i))$*:* Given a first ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3)$ and a public key pk_i as input, first check if either of the condition (1) or (2) holds, else output *invalid*.

Search list L_{H_3} and L_{H_2} to check whether there exists a tuple $\langle m, \omega, r\rangle \in L_{H_3}$ and $\langle R, \psi\rangle \in L_{H_2}$ respectively such that:

$$H_2(R) \oplus (m||\omega) = \mathbb{D}_2,$$
$$r = H_1(m||\omega),$$
$$\mathbb{D}_3 = r \cdot Q$$

If yes, return $(m||\omega)$ to $\mathscr{A}$, else return $\perp$.

- **Challenge:** $\mathscr{P}$ outputs two equal length messages $m_0, m_1 \in \{0,1\}^l$. $\mathscr{C}$ picks $\psi \leftarrow \{0,1\}$ and simulates the challenge ciphertext as given below:

  1. Pick ω^* at random, and implicitly define $H_3(m_b, \omega^*) = ab$.
  2. Compute $\mathbb{C}_1^* = \bar{x}_A(bP) = \frac{\bar{x}_A}{a} abP = r \cdot X_A$.
  3. Compute $\mathbb{C}_2^* = \bar{y}_A(bP) = \frac{\bar{y}_A}{a} abP = r \cdot Y_A$.
  4. Compute $\mathbb{C}_3^* = (m_\psi||\omega^*) \oplus H_2(T^{\bar{z}_A} \cdot \hat{e}(aP, bP)^{h_A})$.
  5. Compute $\mathbb{C}_5^* = t'bP = t'ab \cdot \frac{1}{a}P = rQ$.
  6. Check if the tuple $\langle pk_{i^*}, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*, h'_{i^*}\rangle$ already exists in H_4. If so, GOTO step 1 to recompute with fresh random values. Set $H_4(pk_A, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*) = t'\frac{1}{a}P$, where $t' \in \mathbb{Z}_q^*$ is a known quantity to $\mathscr{C}$. Compute $\mathbb{C}_4^* = t'bP = t'ab\frac{1}{a}P = r.H_4(pk_A, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*)$.
  7. Return the challenge $\mathbb{C}^*$= $(\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_4^*, \mathbb{C}_5^*)$.

- **Phase 2:** The adversary $\mathscr{P}$ issues queries as in Phase 1 with the restrictions described in our NT-PRE game, and the challenger $\mathscr{C}$ responds to the queries, as in Phase 1.

- **Guess:** $\mathscr{P}$ eventually produces his output bit $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathscr{C}$ decides that $T = \hat{e}(P,P)^{abc}$, else T is random.

- **Probability Analysis:** Note that, in this simulation of the first level ciphertext security, there are no aborting scenarios in the training phase and challenger phase. Hence $\mathscr{C}$ solves the 1-*wDBDHI* problem with the same advantage as $\mathscr{A}$. Therefore, the advantage of $\mathscr{C}$ in solving the 1-*wDBDHI* instance is:

$$\varepsilon' \geq \varepsilon,$$

The running time of $\mathscr{P}$ is:

$$\begin{aligned} t^* \leq t + (&q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{pf} + q_{RE} + q_{Dec} \\ &+ q_{RD})O(1) + (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} \\ &+ 2q_{\mathscr{X}} + 2q_{Dec} + q_{RD})t_{et} + (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp}, \end{aligned}$$

where t_e, t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation. This completes the proof of the theorem.

□

Second Level Ciphertext Security:

Theorem 2. *Our unidirectional single-hop proxy re-encryption scheme is CCA-secure for the second level ciphertext under the m-DBDH assumption. If there exists an adversary $\mathscr{A}$ that has an advantage ε in breaking the CCA-security of the proposed scheme in time t, then we can construct a challenger C who breaks the hard problem instance with an advantage ε' within time t' where:*

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

$$\begin{aligned} t^* \leq t &+ (q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{RE} + q_{Dec} + q_{RD})O(1) \\ &+ (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} + 2q_{\mathscr{X}} + 2q_{Dec} + q_{ReDec})t_{et} \\ &+ (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp}, \end{aligned}$$

where t_e,t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation.

Proof. If a second level ciphertext adversary $\mathscr{A}$ with given access to the random oracles $\tilde{H}, H_1, H_2, H_3, H_4$ breaks the IND-PRE-CCA security of our scheme, we can construct an algorithm $\mathscr{C}$ who, given $(P, aP, a^{-1}P, a^{-2}P, bP, cP)$ breaks the variant of the m-DBDH assumption in $\mathbb{G}_1$. In our security game, the environment for $\mathscr{A}$ is simulated as follows. Note that $\mathscr{C}$ sets $Q = \frac{1}{a}P$.

- **Key Generation:** $\mathscr{C}$ generates the uncorrupted keys using Coron's technique [7] by flipping a biased coin c_i that takes the value 1 with some probability $\mathscr{E}$, the analysis of which is shown in a later part of the paper. $\mathscr{C}$ maintains a list L_{key} to store the public key-private key pair list as a tuple $\langle pk_i, sk_i, c_i\rangle$. We denote the set of corrupted users as CU and the set of honest users as HU. The keys are generated as follows:
  - *Corrupted Key Generation:* Pick $x_i, y_i, z_i \in \mathbb{Z}_q^*$, and set the public key $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding private key $sk_i = (x_i, y_i, z_i)$. Store the tuple $\langle pk_i, sk_i, c_i = -\rangle$ to the list L_{key}.
  - *Uncorrupted Key Generation:* The uncorrupted keys are generated according to two cases as below:
  ∘ If $c_i = 0$, pick $\bar{x}_i, \bar{y}_i, \bar{z}_i \in \mathbb{Z}_q^*$, set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding private key $sk_i = (x_i, y_i, z_i) = \left(\frac{\bar{x}_i}{a}, \frac{\bar{y}_i}{a}, c\bar{z}_i\right)$. Store the tuple $\langle pk_i, sk_i, c_i = 0\rangle$ to the list L_{key}.
  ∘ If $c_i = 1$, pick $x_i, \bar{y}_i, z_i \in \mathbb{Z}_q^*$, set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding private key $sk_i = (x_i, y_i, z_i) = \left(x_i, \frac{\bar{y}_i}{a}, z_i\right)$. Store the tuple $\langle pk_i, sk_i, c_i = 1\rangle$ to the list L_{key}.

- **Re-encryption Key Generation:** In order to generate the re-encryption keys $RK_{i\to j}$ from user i to user j, $\mathscr{C}$ constructs the different components R, S, T of $RK_{i\to j}$ according to the different cases :
  - ∘ If $(i \in HU \wedge c_i = 0)$ and $(j \in CU)$: output "failure" and abort.
  - ∘ If $(i \in HU \wedge c_i = 0)$ and $(j \in HU \wedge c_j \in \{0,1\})$:
    1. Pick $T, \delta, \bar{s} \in_R \mathbb{Z}_q^*$.
    2. From definition, $\beta = \frac{z_i + h_i}{T} - \delta$.

3. Set $s = \frac{\bar{s}}{a}$ and compute:

$$\begin{aligned}
R &= \bar{x_i}^{-1}(\bar{y}_j\delta + \bar{s})P + \bar{x_i}^{-1}\tilde{H}(X_j)P \\
&= \Big(\frac{\bar{x_i}}{a}\Big)^{-1}\Big(\frac{\bar{y_j}}{a}\delta + \frac{\bar{s}}{a}\Big)P + \Big(\frac{\bar{x_i}}{a}\Big)^{-1}\tilde{H}(X_j)\cdot a^{-1}P \\
&= x_i^{-1}(\delta Y_j + sP) + x_i^{-1}\tilde{H}(X_j)Q \\
S &= \frac{\bar{y_j}\bar{z_i}}{\bar{y_i}T}\cdot cP + \frac{\bar{y_j}}{\bar{y_i}T}(h_i - \delta T)P - \bar{y_i}^{-1}\bar{s}P + \frac{\bar{y_j}}{\bar{y_i}}Q \\
&= \bar{y_i}^{-1}\bar{y_j}\Big(\frac{c\bar{z_i}+h_i}{T} - \delta\Big)P - \bar{y_i}^{-1}\bar{s}P + \bar{y_i}^{-1}\bar{y}_jQ \\
&= \Big(\frac{\bar{y_i}}{a}\Big)^{-1}\Big(\frac{\bar{y_j}}{a}\beta - \frac{\bar{s}}{a}\Big)P + \Big(\frac{\bar{y_i}}{a}\Big)^{-1}\frac{\bar{y_j}}{a}Q \\
&= y_i^{-1}(\beta Y_j - sP) + y_i^{-1}Q_j
\end{aligned}$$

4. Return $RK_{i\to j} = (R,S,T)$. Note that the computed value of the re-encryption key matches with the original distribution of computed re-encrypted key given in the scheme.

- Else, choose $\delta, \beta, \bar{s} \in_R \mathbb{Z}_q^*$, set $s = \frac{\bar{s}}{a}$. Compute $T = \frac{z_i+h_i}{\delta+\beta}$ for each i.
  - If $(i \in CU)$ and $(j \in HU \wedge c_j \in \{0,1\})$, compute:

$$\begin{aligned}
R &= x_i^{-1}(\bar{y}_j\delta + \bar{s})a^{-1}P + x_i^{-1}\tilde{H}(X_j)Q \\
&= x_i^{-1}\big(\frac{\bar{y_j}}{a}\delta + \frac{\bar{s}}{a}\big)P + x_i^{-1}\tilde{H}(X_j)Q \\
&= x_i^{-1}(\delta Y_j + sP) + x_i^{-1}\tilde{H}(X_j)Q \\
S &= y_i^{-1}(\bar{y}_j\beta - \bar{s})a^{-1}P + y_i^{-1}\bar{y}_ja^{-2}P \\
&= y_i^{-1}\big(\frac{\bar{y_j}}{a}\beta - \frac{\bar{s}}{a}\big)P + y_i^{-1}\frac{\bar{y_j}}{a}a^{-1}P \\
&= y_i^{-1}(\beta Y_j - sP) + y_i^{-1}Q_j
\end{aligned}$$

  - If $(i \in HU \wedge c_i = 1)$ and $(j \in HU \wedge c_j \in \{0,1\})$, compute:

$$\begin{aligned}
R &= x_i^{-1}(\bar{y}_j\delta + \bar{s})a^{-1}P + x_i^{-1}\tilde{H}(X_j)Q \\
&= x_i^{-1}\big(\frac{\bar{y_j}}{a}\delta + \frac{\bar{s}}{a}\big)P + x_i^{-1}\tilde{H}(X_j)Q \\
&= x_i^{-1}(\delta Y_j + sP) + x_i^{-1}\tilde{H}(X_j)Q \\
S &= \bar{y_i}^{-1}(\bar{y}_j\beta - \bar{s})P + \bar{y_i}^{-1}\bar{y}_jQ \\
&= \Big(\frac{\bar{y_i}}{a}\Big)^{-1}\big(\frac{\bar{y_j}}{a}\beta - \frac{\bar{s}}{a}\big)P + \frac{\bar{y_j}/a}{\bar{y_i}/a}Q \\
&= y_i^{-1}(\beta Y_j - sP) + y_i^{-1}Q_j
\end{aligned}$$

  - If $(i \in HU \wedge c_i = 1)$ and $(j \in CU)$, compute:

$$\begin{aligned}
R &= x_i^{-1}(\delta y_j + \bar{s})P + x_i^{-1}\tilde{H}(X_j)Q \\
&= x_i^{-1}(\delta Y_j + \bar{s}P) + x_i^{-1}\tilde{H}(X_j)Q \\
S &= \bar{y_i}^{-1}(y_j\beta - \bar{s})aP + \bar{y_i}^{-1}y_jP \\
&= \Big(\frac{\bar{y_i}}{a}\Big)^{-1}(y_j\beta - \bar{s})P + \Big(\frac{\bar{y_i}}{a}\Big)^{-1}y_ja^{-1}P \\
&= y_i^{-1}(\beta Y_j - \bar{s}P) + y_i^{-1}Q_j
\end{aligned}$$

- ∘ Return $RK_{i\to j} = (R,S,T)$, and note that the re-encryption keys in all the cases are identically distributed as the real re-encryption keys from the construction.

- **Phase 1:** $\mathscr{C}$ interacts with $\mathscr{A}$ in the following ways:
  - *Oracle Queries*: Same as in the security game for non-transferability.
  - *Re-encryption Queries* ($\mathscr{O}_{RE}(\mathbb{C}, pk_i, pk_j)$)*:* Given a second level ciphertext $\mathbb{C} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4, \mathbb{C}_5)$ and public keys pk_i and pk_j as input, first check if either of the conditions (1) or (2) holds. If they do not hold, output *invalid*. Otherwise, retrieve the key tuples $\langle pk_i, x_i, y_i, z_i, c_i\rangle$, $\langle pk_j, x_j, y_j, z_j, c_j\rangle$ from the list L_{key} and consider the possibilities:
    - If $(i \in HU \wedge c_i = 0)$ and $j \in CU$, retrieve the value of $H_4(pk_i, \mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_5)$ to get h'_i, where $h'_i = t \cdot cP$ for known $t \in_R \mathbb{Z}_q^*$. Compute:

$$\begin{aligned}\mathbb{D}_1 &= \hat{e}(\mathbb{C}_5, aP)^{y_j h_i} \cdot \hat{e}(t^{-1}\mathbb{C}_4, Y_j)^{\bar{z}_i}\\ &= \hat{e}(r\frac{1}{a}P, aP)^{y_j h_i} \cdot \hat{e}(t^{-1}rtcP, y_jP)^{\bar{z}_i}\\ &= \hat{e}(P,P)^{ry_jh_i} \cdot \hat{e}(P,P)^{y_jrc\bar{z}_i}\\ &= \hat{e}(P,P)^{ry_j(z_i+h_i)}\end{aligned}$$

      Set $\mathbb{D}_2 = \mathbb{C}_3, \mathbb{D}_3 = \mathbb{C}_5$. Return the second level ciphertext $\mathbb{D}$.
    - For all other cases, run the re-encryption key generation query $\mathscr{O}_{RK}(pk_i, pk_j)$ to obtain $RK_{i\to j}$. Run *Re-Encrypt*$(\mathbb{C}, RK_{i\to j})$ and return the second level ciphertext $\mathbb{D}$ to $\mathscr{A}$.
  - *Second Level Decryption Queries* ($\mathscr{O}_{Dec}(\mathbb{C}, pk_i)$) and *Re-Decryption Queries* ($\mathscr{O}_{RD}(\mathbb{D}, pk_i)$) are same as given in the security game for non-transferability.

- **Challenge:** When $\mathscr{A}$ decides that Phase 1 is over, it outputs a target public key pk_i^* and two equal length messages $m_0, m_1 \in \{0,1\}^l$. $\mathscr{C}$ picks $\psi \leftarrow \{0,1\}$ and simulates the challenge ciphertext as given below:
  1. If $c^* = 1$, output failure and abort.
  2. Else, pick ω^* at random, and implicitly define $H_3(m_b, \omega^*) = ab$.
  3. Compute $\mathbb{C}_1^* = \bar{x}_i^*(bP) = \frac{\bar{x}_i^*}{a}abP = r \cdot X_i^*$.
  4. Compute $\mathbb{C}_2^* = \bar{y}_i^*(bP) = \frac{\bar{y}_i^*}{a}abP = r \cdot Y_i^*$.
  5. Compute $\mathbb{C}_3^* = (m_\psi || \omega^*) \oplus H_2(T^{\bar{z}_i^*} \cdot \hat{e}(aP, bP)^{h_i})$.
  6. Compute $\mathbb{C}_5^* = t'bP = t'ab \cdot \frac{1}{a}P = rQ$.
  7. Check if the tuple $\langle pk_{i^*}, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*, h'_{i^*}\rangle$ already exists in list L_{H_4}. If so, GOTO step 1 to recompute with fresh random values. Set $H_4(pk_i^*, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*) = t'\frac{1}{a}P$, where $t' \in \mathbb{Z}_q^*$ is a known quantity to the challenger $\mathscr{C}$. Compute $\mathbb{C}_4^* = t'bP = t'ab\frac{1}{a}P = r.H_4(pk_i^*, \mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_5^*)$.
  8. Return the challenge $\mathbb{C}^*$= ($\mathbb{C}_1^*, \mathbb{C}_2^*$,$\mathbb{C}_3^*$,$\mathbb{C}_4^*, \mathbb{C}_5^*$).

- **Phase 2:** The adversary $\mathscr{A}$ issues queries as in Phase 1 with the restrictions described in our IND-PRE-CCA game, and the challenger $\mathscr{C}$ responds to the queries, as in Phase 1.

- **Guess:** $\mathscr{A}$ eventually produces his output bit $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathscr{C}$ decides that $T = \hat{e}(P,P)^{abc}$, else T is random.

- **Probability Analysis:** Inorder to denote the advantage of the challenger $\mathscr{C}$ in solving the m-DBDH instance $(P, aP, a^{-1}P, a^{-2}P, bP, cP)$, we calculate the probability that $\mathscr{C}$ aborts during the simulation. Let *Abort* denote the event that $\mathscr{C}$ aborts during the game. We note that $\mathscr{C}$ does not abort in the following events:

  – E_1: $\mathscr{C}$ does not abort in the re-encryption key generation query.

  – E_2: $\mathscr{C}$ does not abort in the challenge phase.

  Since the probability that the target user pk_i^* has $c_i^* = 0$ is exactly $(1-\mathscr{E})$, $Pr[E_2]$ is exactly $(1-\mathscr{E})$. Let q_{RK} denote the number of queries made by $\mathscr{A}$ to the re-encryption key generation oracle. For each such query, $\mathscr{C}$ does not abort with a probability of $\mathscr{E}$. Hence, probability that $\mathscr{C}$ does not abort $Pr[\neg Abort] \geq \mathscr{E}^{q_{RK}} \cdot (1-\mathscr{E})$, which has a maximum value at $\mathscr{E}_{OPT} = \frac{q_{RK}}{1+q_{RK}}$. Using $\mathscr{E}_{OPT}$, we obtain:

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

  where e is the base of the natural logarithm. Therefore, the advantage of $\mathscr{C}$ in solving the *m-DBDH* instance is:

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

  The running time of $\mathscr{C}$ is:

$$\begin{aligned}t^* \leq\ & t + (q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{RE} + q_{Dec} + q_{RD}) \\ & O(1) + (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} + 2q_{\mathscr{X}} + 2q_{Dec} + \\ & q_{ReDec})t_{et} + (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp},\end{aligned}$$

  where e is the base of natural logarithm, t_e, t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation. This completes the proof of the theorem.

□

First Level Ciphertext Security:

Theorem 3. *Our unidirectional single-hop proxy re-encryption scheme is CCA-secure for the second level ciphertext under the m-DBDH assumption. If there exists an adversary $\mathscr{A}$ that has an advantage ε in breaking the CCA-security of the proposed scheme in time t, then we can construct a challenger C who breaks the hard problem instance with an advantage ε' within time t' where:*

$$\varepsilon' \geq \varepsilon,$$

$$\begin{aligned}t^* \leq\ & t + (q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{RE} + q_{Dec} + q_{RD})O(1) \\ & + (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} + 2q_{\mathscr{X}} + 2q_{Dec} + q_{ReDec})t_{et} \\ & + (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp},\end{aligned}$$

where t_e, t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation.

Proof. Given a first level adversary $\mathscr{A}$ who, given access to random oracle queries to $\tilde{H}, H_1, H_2, H_3, H_4$, breaks the IND-PRE-CCA security of our scheme, we can construct an algorithm $\mathscr{C}$ who, given $(P, \frac{1}{a}P, bP)$ can decide if $T = \hat{e}(P,P)^{ab}$ and breaks the 1-wDBDHI assumption in $\mathbb{G}_1$. In our security game, the environment for $\mathscr{A}$ is simulated as follows. Note that $\mathscr{C}$ sets $Q = \frac{1}{a}P$.

- **Key Generation:** $\mathscr{C}$ generates the keys as follows:
  - *Corrupted Key Generation (CH):* Pick $x_i, y_i, z_i \in \mathbb{Z}_q^*$, and set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding $sk_i = (x_i, y_i, z_i)$.
  - *Uncorrupted Key Generation (HU):* Pick $x_i, \bar{y}_i, z_i \in \mathbb{Z}_q^*$, set $pk_i = (X_i, Y_i, Z_i, Q_i) = (x_iP, y_iP, z_iP, y_iQ)$ and the corresponding private key $sk_i = (x_i, y_i, z_i) = (x_i, \frac{\bar{y}_i}{a}, z_i)$.

- **Re-encryption Key Generation:** Inorder to generate the re-encryption keys $RK_{i\to j}$ from user i to user j, $\mathscr{C}$ constructs the different components R, S, T of $RK_{i\to j}$ according to the different cases:

  ○ Choose $\delta, \beta, \bar{s} \in_R \mathbb{Z}_q^*$, set $s = \frac{\bar{s}}{a}$. Compute $T = \frac{z_i + h_i}{\delta + \beta}$ for each i,

   – If $(i \in HU)$ and $(j \in CU)$, compute:

$$\begin{aligned}
R &= {x_i}^{-1}(\delta y_j + \bar{s})P + {x_i}^{-1}\tilde{H}(X_j)Q \\
&= {x_i}^{-1}(\delta Y_j + \bar{s}P) + {x_i}^{-1}\tilde{H}(X_j)Q \\
S &= {\bar{y}_i}^{-1}(y_j\beta - \bar{s})aP + {\bar{y}_i}^{-1}y_jP \\
&= \left(\frac{\bar{y}_i}{a}\right)^{-1}(y_j\beta - \bar{s})P + \left(\frac{\bar{y}_i}{a}\right)^{-1} y_j a^{-1}P \\
&= {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j
\end{aligned}$$

   – If $(i \in CU)$ and $(j \in HU)$, compute:

$$\begin{aligned}
R &= {x_i}^{-1}(\bar{y}_j\delta + \bar{s})a^{-1}P + {x_i}^{-1}\tilde{H}(X_j)Q \\
&= {x_i}^{-1}\left(\frac{\bar{y}_j}{a}\delta + \frac{\bar{s}}{a}\right)P + {x_i}^{-1}\tilde{H}(X_j)Q \\
&= {x_i}^{-1}(\delta Y_j + sP) + {x_i}^{-1}\tilde{H}(X_j)Q \\
S &= {y_i}^{-1}(\bar{y}_j\beta - \bar{s})a^{-1}P + {y_i}^{-1}\bar{y}_j a^{-2}P \\
&= {y_i}^{-1}\left(\frac{\bar{y}_j}{a}\beta - \frac{\bar{s}}{a}\right)P + {y_i}^{-1}\frac{\bar{y}_j}{a}\frac{1}{a}P \\
&= {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j
\end{aligned}$$

   – If $(i, j \in HU)$, compute:

$$\begin{aligned}
R &= {x_i}^{-1}(\bar{y}_j\delta + \bar{s})a^{-1}P + {x_i}^{-1}\tilde{H}(X_j)Q \\
&= {x_i}^{-1}\left(\frac{\bar{y}_j}{a}\delta + \frac{\bar{s}}{a}\right)P + {x_i}^{-1}\tilde{H}(X_j)Q \\
&= {x_i}^{-1}(\delta Y_j + sP) + {x_i}^{-1}\tilde{H}(X_j)Q \\
S &= {\bar{y}_i}^{-1}(\bar{y}_j\beta - \bar{s})P + {\bar{y}_i}^{-1}\bar{y}_jQ \\
&= \left(\frac{\bar{y}_i}{a}\right)^{-1}\left(\frac{\bar{y}_j}{a}\beta - \frac{\bar{s}}{a}\right)P + \left(\frac{\bar{y}_i}{a}\right)^{-1}\frac{\bar{y}_j}{a}Q \\
&= {y_i}^{-1}(\beta Y_j - sP) + {y_i}^{-1}Q_j
\end{aligned}$$

  ○ Return $RK_{i\to j} = (R, S, T)$, and note that the re-encryption keys in all the cases are identically distributed as the real re-encryption keys from the construction.

- **Phase 1:** $\mathscr{C}$ interacts with $\mathscr{A}$ in the following ways:
  - *Oracle Queries*: Same as in the security game for non-transferability.
  - *Re-Decryption Queries ($\mathscr{O}_{RD}(\mathbb{D}, pk_i)$):* Given a first ciphertext $\mathbb{D} = (\mathbb{D}_1, \mathbb{D}_2, \mathbb{D}_3)$ and a public key pk_i as input, first check if either of the condition (1) or (2) holds, else output *invalid*. Search list L_{H_3} and L_{H_2} to check whether there exists a tuple $\langle m, \omega, r\rangle \in L_{H_3}$ and $\langle R, \psi\rangle \in L_{H_2}$ respectively such that:

    $$H_2(R) \oplus (m||\omega) \overset{?}{=} \mathbb{D}_2,\ r \overset{?}{=} H_1(m||\omega) \text{ and } \mathbb{D}_3 \overset{?}{=} r \cdot Q$$

    If yes, return $(m||\omega)$ to $\mathscr{A}$, else return $\perp$.

- **Challenge:** When $\mathscr{A}$ decides that Phase 1 is over, it outputs a target public key pk_i^* and two equal length messages $m_0, m_1 \in \{0,1\}^l$. $\mathscr{C}$ picks $\psi \leftarrow \{0,1\}$ and simulates the challenge ciphertext as given below:
  1. Pick $\omega^* \in_R \{0,1\}^{\omega}$, implicitly define $H_3(m_b||\omega^*) = r = ab$.
  2. Compute $\mathbb{D}_1 = \hat{e}(P, bP)^{y_{\bar{i}}*(z_i^*+h_i^*)}$
     $= \hat{e}(P,P)^{ry_i^*(z_i^*+h_i^*)}$.
  3. Compute $\mathbb{D}_2 = (m_{\psi}||\omega) \oplus H_2(T^{z_i^*+h_i^*})$.
  4. Compute $\mathbb{D}_3 = bP = ab\frac{1}{a}P = r \cdot Q$.

- **Phase 2:** Adversary $\mathscr{A}$ issues queries as in Phase 1 with the restrictions described in our IND-PRE-CCA game, and the challenger $\mathscr{C}$ responds to the queries, as in Phase 1.

- **Guess:** $\mathscr{A}$ eventually produces his output bit $\psi' \in \{0,1\}$. If $\psi' = \psi$, $\mathscr{C}$ decides that $T = \hat{e}(P,P)^{ab}$, else T is random.

- **Probability Analysis:** Note that, in this simulation of the first level ciphertext security, there are no aborting scenarios in the training phase and challenger phase. Hence $\mathscr{C}$ solves the $1-wDBDHI$ problem with the same advantage as $\mathscr{A}$. Therefore, the advantage of $\mathscr{C}$ in solving the $1-wDBDHI$ instance is $\varepsilon' \geq \varepsilon$. The running time of $\mathscr{C}$ is:

  $$\begin{aligned} t^* \leq\ & t + (q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + q_{RE} + q_{Dec} + q_{RD}) \\ & O(1) + (q_{H_4} + 6q_{RK} + q_{\mathscr{X}} + 7q_{Dec} + q_{RD})t_e + (2q_{RE} + 2q_{\mathscr{X}} + 2q_{Dec} + \\ & q_{ReDec})t_{et} + (6q_{RE} + 2q_{\mathscr{X}} + 6q_{Dec} + 4q_{RD})t_{bp}, \end{aligned}$$

  where t_e, t_{et} and t_{bp} are time taken for one exponentiation in $\mathbb{G}_1$, exponentiation in $\mathbb{G}_2$ and a bilinear pairing operation. This completes the proof of the theorem.

□

6 Comparison

This section provides a comparison of our PRE scheme with the existing single-hop PRE schemes in the literature in the light of non-transferable property. We use notations: $t_e, t_{et}, t_{bp}, t_s, t_v, t_{me}$ to denote the time required for exponentiation in groups $\mathbb{G}_1$ and $\mathbb{G}_2$, bilinear pairing, signing algorithm, verification algorithm and multi-exponentiation in group $\mathbb{G}_1$ respectively. Table 1 reports various PRE properties satisfied by the existing schemes alongside our PRE construction, whereby all the required properties of an efficient unidirectional single-hop PRE scheme are realized by our scheme. Table 2 shows the computational efficiency of our scheme along with a few well-known PRE schemes. The computational complexity of our second level decryption algorithm $Decrypt$ is computed considering conditions (1) and (2) are used alongside any one of conditions (6) to (9) of the $Verify()$ algorithm for public verification of ciphertext. In [17], $O(n) = O(logN)$, where N is the maximum number of delegatees for every delegator. The comparisons indicate that our proposed design is the first scheme that achieves non-transferability using bilinear pairing incurring minimal efficiency loss.

Table 1: Comparative analysis of the properties of PRE schemes with our scheme

Property	[25]	[13]	[12]	[10]	Our Scheme
Model	Random Oracle	Random Oracle	Standard	Standard	Random Oracle
Security	CCA	CCA	RCCA	CPA	CCA
Uni-directional	✓	✓	✓	✓	✓
Non-interactive	×	×	✓	✓	✓
Proxy invisibility	✓	✓	✓	✓	✓
Single-hop	✓	✓	✓	✓	✓
Key Optimal	✓	✓	✓	✓	✓
Collusion-safe	✓	✓	✓	✓	✓
Non-transitive	✓	✓	✓	✓	✓
Non-transferable	×	✓	×	✓	✓
Non-key escrow	✓	×	✓	✓	✓

Table 2: Efficiency comparison of unidirectional PRE schemes with our scheme

Scheme	**Encrypt**	**Decrypt**	**Re-Encrypt**	**Re-Decrypt**
[10]	$5t_e + 5t_{et} + 8t_{bp}$	$t_e + 6t_{et} + 4t_{bp}$	$t_e + t_{et} + t_{bp}$	$2t_e + 2t_{et} + 3t_{bp}$
[17]	$((n+2)t_e + t_{et})^*$	$t_e + t_{bp}$	$2t_{bp}$	t_{et}
[12]	$t_s + 4t_e + t_{et} + t_{bp}$ $+t_{me}$	$t_e + t_{et} + 9t_{bp} + t_v$	$t_e + 8t_{bp} + t_v$	$t_e + 2t_{et} + 18t_{bp}$ $+t_v$
[25]	$3t_e + t_{et} + t_{bp}$	$2t_{bp}$	$4t_{bp}$	$2t_{bp}$
Our Scheme	$5t_e + t_{et} + t_{bp}$	$5t_e + t_{et} + t_{bp}$ or $(2t_e + t_{et} + 5t_{bp})^{**}$	$t_{et} + 8t_{bp}$	t_{et}

7 Conclusion

Although there are several schemes achieving PRE in the literature, solely two protocols, due to [13] (certificateless-based setting) and [10] have reported non-transferability. However, [10] resorts to indistinguishability obfuscation ($i\mathscr{O}$) to achieve non-transferability, that involves complex operations making their scheme highly impractical. [13] designs a certificateless PRE protocol that consists of multiple rounds of interaction for key validations and partial-key generations. This too incurs computational overhead as reported in our comparison. Our non-transferable PRE scheme is practical as it is based on direct manipulation in groups. Our scheme is shown to be CCA secure in the random oracle model for both levels of ciphertext. Also, our design meets the definition of non-transferability whereby the colluders cannot re-delegate the decryption rights of the delegator. In fact, the private key components of the colluding delegatee gets exposed by any illegal decryption box constructed by the colluders to decrypt the ciphertexts encrypted towards the delegator. Thus, our efficient non-transferable PRE scheme affirmatively resolves the problem of illegal transference of decryption rights in PRE.

Note that, achieving non-transferability in proxy re-cryptosystems has been considered as *sine qua non* for the real-world deployment of such public-key cryptosystems. Designing unidirectional non-transferable PRE schemes with CCA security in other public key settings such as identity-based or certificateless frameworks based on random oracle and standard models with novel re-encryption key constructs that support proxy with a separate private key, translation power and additional operations is an interesting research direction to work on, and we leave it as an open problem.

References

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *Proc. of the 2005 Network and Distributed System Security Symposium NDSS'05, San Diego, California, USA*. The Internet Society, February 2005.

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9(1):1–30, February 2006.

[3] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the first ACM Conference on Computer and Communications Security (CCS'93), Fairfax, Virginia, USA*, pages 62–73. ACM, November 1993.

[4] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Proc. of the 17th International Conference on the Theory and Application of Cryptographic Techniques (Eurocrypt'98), Espoo, Finland*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, May-June 1998.

[5] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proc. of the 23rd International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'04), Interlaken, Switzerland*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer, May 2004.

[6] S. S. Chow, J. Weng, Y. Yang, and R. H. Deng. Efficient unidirectional proxy re-encryption. In *Proc. of the Third International Conference on Cryptology in Africa (AFRICACRYPT'10), Stellenbosch, South Africa*, volume 6055 of *Lecture Notes in Computer Science*, pages 316–332. Springer, 2010.

[7] J.-S. Coron. On the exact security of full domain hash. In *Proc. of the 20th Annual International Cryptology Conference (Crypto'00), Santa Barbara, California, USA*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer, August 2000.

[8] R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In *Proc. of the 7th International Conference on Cryptology and Network Security (CANS'08), Hong-Kong, China*, volume 5339 of *Lecture Notes in Computer Science*, pages 1–17. Springer, December 2008.

[9] M. Egorov and M. Wilkison. Nucypher KMS: decentralized key management system. *CoRR*, abs/1707.06140, 2017.

[10] H. Guo, Z. Zhang, and J. Xu. Non-transferable proxy re-encryption. *IACR Cryptology ePrint Archive*, 2015:1216, 2015.

[11] Y. Han, X. Gui, X. Wu, and X. Yang. Proxy encryption based secure multicast in wireless mesh networks. *Journal of Network and Computer Applications*, 34(2):469–477, March 2011.

[12] R. Hayashi, T. Matsushita, T. Yoshida, Y. Fujii, and K. Okada. Unforgeability of re-encryption keys against collusion attack in proxy re-encryption. In *Proc. of the 6th International Workshop on Advances in Information and Computer Security (IWSEC'11), Tokyo, Japan*, volume 7038 of *Lecture Notes in Computer Science*, pages 210–229. Springer, November 2011.

[13] Y.-J. He, T. W. Chim, L. C. K. Hui, and S.-M. Yiu. Non-transferable proxy re-encryption scheme. In *Proc. of the 5th International Conference on New Technologies, Mobility and Security (NTMS'12), Istanbul, Turkey*, pages 1–4. IEEE, May 2012.

[14] T. Isshiki, M. H. Nguyen, and K. Tanaka. Attacks to the proxy re-encryption schemes from iwsec2011. In *Proc. of the 8th International Workshop on Security (IWSEC'13), Okinawa, Japan*, pages 290–302, Novem-

ber 2013.

[15] S. Kamara and K. E. Lauter. Cryptographic cloud storage. In *Proc. of the 14th Financial Cryptograpy and Data Security International Conference (FC'10), Tenerife, Canary Islands, Spain*, volume 6054 of *Lecture Notes in Computer Science*, pages 136–149. Springer, January 2010.

[16] K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, T. V. X. Phuong, and Q. Xie. A dfa-based functional proxy re-encryption scheme for secure public cloud data sharing. *IEEE Transactions on Information Forensics and Security*, 9(10):1667–1680, October 2014.

[17] B. Libert and D. Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Proc. of the Second International Conference on Pairing-Based Cryptography (Pairing'08), Egham, UK*, volume 5209 of *Lecture Notes in Computer Science*, pages 332–353. Springer, September 2008.

[18] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. *IEEE Transactions on Information Theory*, 57(3):1786–1802, February 2011.

[19] Q. Liu, G. Wang, and J. Wu. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Information Sciences*, 258:355–370, February 2014.

[20] S. Mahajan and A. Jindal. Security and privacy in vanet to reduce authentication overhead for rapid roaming networks. *International Journal of Computer Applications*, 1(20):21–25, February 2010.

[21] D. Nuñez, I. Agudo, and J. López. Proxy re-encryption: Analysis of constructions and its application to secure access delegation. *Journal of Network and Computer Applications*, 87:193–209, June 2017.

[22] S. S. D. Selvi, A. Paul, and C. P. Rangan. An efficient non-transferable proxy re-encryption scheme. In *Proc. of the 8th International Conference on Applications and Techniques in Information Security (ATIS'17), Auckland, New Zealand*, volume 719 of *Communications in Computer and Information Science*, pages 35–47. Springer, July 2017.

[23] H. Wang, Z. Cao, and L. Wang. Multi-use and unidirectional identity-based proxy re-encryption schemes. *Information Sciences*, 180(20):4042–4059, October 2010.

[24] H. Wang, P. Zeng, and K. R. Choo. MDMR-IBE: efficient multiple domain multi-receiver identity-based encryption. *Security and Communication Networks*, 7(11):1641–1651, August 2013.

[25] L. Wang, L. Wang, M. Mambo, and E. Okamoto. New identity-based proxy re-encryption schemes to prevent collusion attacks. In *Proc. of the 4th International Conference on Pairing-Based Cryptography (Pairing'10), Yamanaka Hot Spring, Japan*, volume 6487 of *Lecture Notes in Computer Science*, pages 327–346. Springer, December 2010.

[26] J. Weng, R. H. Deng, S. Liu, and K. Chen. Chosen-ciphertext secure bidirectional proxy re-encryption schemes without pairings. *Information Sciences*, 180(24):5077–5089, December 2010.

[27] Y. Yang and M. Ma. Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. *IEEE Transactions on Information Forensics and Security*, 11(4):746–759, April 2016.

[28] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *Proc. of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS'10), Beijing, China*, pages 261–270. ACM, April 2010.

Author Biography

Arinjita Paul is a PhD research scholar at the Department of Computer Science and Engineering of Indian Institute of Technology Madras, Chennai, India. Her research interests include design and cryptanalysis of Public Key Cryptosystems and Network Security. She is currently exploring public key primitives such as proxy re-encryption and exploring their applications to blockchain technology.

Lihua Wang received the B.S. degree in mathematics from Northeast Normal University, China, in 1988, the M.S. degree in mathematics from the Harbin Institute of Technology, China, in 1994, and the Ph.D. degree in engineering from the University of Tsukuba, Japan, in 2006. She is currently a Senior Researcher with the Cybersecurity Research Institute, National Institute of Information and Communications Technology, Japan. Her current research interests are applied cryptography and privacy- preserving data mining.

S. Sharmila Deva Selvi is a Post Doctoral Researcher at the Department of Computer Science and Engineering from the Indian Institute of Technology Madras, Chennai, India. She received her PhD at IIT Madras in 2012. Her areas of interest are Provable Security of Public Key Cryptosystem, Cryptanalysis of Identity Based and Certificateless Cryptosystem, and designing solutions for secure storage and computation in the Cloud.

C. Pandu Rangan is a Chair Professor in the department of computer science and engineering of Indian Institute of Technology Madras, Chennai, India. He obtained his PhD at the Indian Institute of Science (IISc), Bangalore in 1984. His areas of interest are in theoretical computer science mainly focusing on Cryptography, Algorithms and Data Structures, Game Theory, Graph Theory and Distributed Computing.