A Provably Secure Conditional Proxy Re-Encryption Scheme without Pairing

Arinjita Paul*, S. Sharmila Deva Selvi, and C. Pandu Rangan
Department of Computer Science and Engineering, IIT Madras, Chennai, India
{arinjita,sharmila,prangan}@cse.iitm.ac.in

Received: March 2, 2021; Accepted: May 15, 2021; Published: May 31, 2021

Abstract

Blaze, Bleumer and Strauss introduced the notion of proxy re-encryption (PRE), a cryptographic primitive that enables a semi-trusted proxy to transform ciphertexts under the public key of a user (delegator) into ciphertexts under the public key of another user (delegatee). The important property to note here is, the proxy should not learn anything about the plaintext encrypted. In 2009, Weng *et al*. introduced the concept of conditional proxy re-encryption (CPRE), which permits the proxy to re-encrypt only ciphertexts satisfying a condition specified by the delegator into a ciphertext for the delegatee. CPRE enables fine-grained delegation of decryption rights useful in many practical scenarios, such as blockchain-enabled distributed cloud storage and encrypted email forwarding. Several CPRE schemes exist in the literature based on costly bilinear pairing operation in the random oracle model. We propose the first construction of an efficient CPRE scheme without pairing, satisfying chosen ciphertext security under the computational Diffie Hellman (CDH) assumption and its variant in the random oracle model.

Keywords: Proxy Re-Encryption, Public Key, Conditional, Pairing-less, Unidirectional, Single hop, CCA-secure

1 Introduction

Proxy re-encryption (PRE) introduced by Blaze, Bleumer and Strauss [5] is a cryptographic primitive which has come into light in the last two decades. It enables re-encryption of ciphertexts under a user's public key, say Alice, into ciphertexts under another user's public key, say Bob, with the help of a semi-trusted third party termed proxy. Note that, this can be trivially achieved by Alice who can decrypt the ciphertext and encrypt it further using Bob's public key. However, this requires Alice to remain online which is not feasible. In PRE, the proxy is authorised to re-encrypt ciphertexts under the public key of delegator Alice using a special information (re-encryption key) that is shared by Alice, without learning any information about the underlying plaintext or recovering private keys of Alice. Proxy re-encryption schemes are categorised on various basis. Based on the direction of rights to delegate, PRE schemes are classified into unidirectional and bidirectional schemes. Based on the number of re-encryptions allowed, PRE schemes are classified into single-hop and multi-hop schemes. In this work, we focus on single-hop unidirectional PRE schemes. PRE is extensively used in encrypted email forwarding, distributed file systems, DRM of Apple's iTunes, privacy in public transportation and outsourced filtering of encrypted spam [2, 3, 6, 34, 26].

Journal of Internet Services and Information Security (JISIS), volume: 11, number: 2 (May 2021), pp. 1-21
DOI:*10.22667/JISIS.2021.05.31.001*

*Corresponding author: Department of Computer Science and Engineering, IIT Madras, Chennai-600036, India, Tel: +91-44-2257-5387

Owing to its transformational property, a major application of PRE is enabling secure file sharing in blockchain powered data storage [1]. Decentralised applications (DApps) often rely on third parties to provide services such as data storage. Therein, PRE facilitates data owners to securely share encrypted content with other users, while the third party storage provider has no knowledge of the uploaded content. In 2009, Weng *et al*. [32] introduced the notion of conditional proxy re-encryption (CPRE), a variant of PRE, enabling the delegator to implement fine-grained delegation of decryption rights. Let us consider the following scenario. A DApp user Alice wishes to share her encrypted data stored on the cloud. However, Alice chooses to share only her *media* files with Bob, whereas share her *account* files with Carol alone. Note that, the traditional PRE schemes fail to provide the desired fine-grained control of delegation. Conditional PRE efficiently addresses the problem by assigning the decryption capability to Bob and Carol based on preferred conditions *media* and *account* set by Alice respectively. In CPRE, the ciphertexts and re-encryption keys are generated only with respect to the condition and decryption is possible only if the associated condition is satisfied. In order to facilitate secure data sharing in the distributed cloud, it is essential to build efficient CPRE protocols. All the existing CPRE protocols are designed using expensive bilinear pairing operations, which makes it very difficult to be adopted in practice. We address this issue and propose an efficient pairing-free solution tailored to practical settings such as cloud storage.

1.1 Related Work and Contribution

Removing pairing operations from unidirectional PRE constructions is one of the major open problems stated by Canetti *et al*. [7]. As a solution to providing flexible delegation of decryption rights, Weng *et al*. [32] introduced the notion of conditional proxy re-encryption. Although several constructions achieving CPRE has been proposed in the literature, to the best of our knowledge, all existing schemes are based on the expensive bilinear pairing operation. In [32], the first construction of a CCA-secure CPRE scheme is proposed in the random oracle model, assuming the 3-QBDH (3-Quotient Bilinear Diffie Hellman) assumption. Their design involves computing two separate keys (re-encryption key and condition key), used by the proxy to achieve conditional re-encryption. Tang *et al*. [29] independently proposed the concept of conditional re-encryption labelled as *type-based PRE* enabling the re-encryption key to transform a subset of ciphertexts. Their scheme is CCA secure based on the CBDH (Computational Bilinear Diffie Hellman) and KE (Knowledge of Exponent) assumptions in the random oracle model. Their construction limits the delegators and delegatees to be in different systems, such that a user in a given system can only act as either (not both) a delegator or a delegatee. Note that the security notions in [32, 29] consider only second level ciphertext security and does not address security of first level ciphertexts. Also, the scheme due to Weng *et al*. [32] is shown to be vulnerable to CCA attack by Weng *et al*. [33]. In 2009, Weng *et al*. [33] re-formalised the security notions of CPRE, by proposing a rigorous security model addressing ciphertext security at both levels. They also proposed a new construction of CCA-secure CPRE in the random oracle model, by combining both keys (re-encryption key and condition key) into a single key, used for re-encryption. Their scheme is CCA secure based on the DBDH (Decisional Bilinear Diffie Hellman) assumption in the random oracle model. Later, Chu *et al*. [9] introduced a generalised version of CPRE, termed conditional proxy broadcast re-encryption (CBPRE), that conditionally re-encrypts ciphertexts for only a set of users at a time satisfying the specified condition, and proposed an RCCA-secure CBPRE scheme based on the n-BDHE (n-Bilinear Diffie Hellman Exponent) assumption in the standard model. RCCA security is a weaker variant of CCA security wherein a harmless mauling of the challenge ciphertext is tolerated. Fang *et al*. [13] formalized the notion of anonymous RCCA-secure CPRE such that the condition associated with the ciphertext remains anonymous to the proxy, and presented a concrete construction of anonymous CPRE scheme based on the 3-QDBDH and truncated q-ABDHE (q-Augmented Bilinear Diffie Hellman Exponent) assumption in the standard model.

Fang *et al*. [11] proposed the first Chosen-Ciphertext Secure anonymous conditional proxy re-encryption with the added functionality supporting keyword search (C-PRES), based on the DBDH and truncated q-ABDHE assumption in the random oracle model. Shao *et al*. [25] proposed an identity based conditional proxy re-encryption scheme secure against CCA attack, secure under the DBDH assumption in the random oracle model. Further, Liang *et al*. [18] proposed a CCA-secure identity based CPRE scheme based on the DBDH assumption in the standard model. However, He *et al*. [14] shows that the scheme in [18] is vulnerable to CCA attack. Vivek *et al*. [30, 31] proposed a CCA secure CPRE scheme secure under the mCBDH (modified Computational Bilinear Diffie-Hellman) assumption in the random oracle model. Further, Son *et al*. [27] proposed a CCA secure CPRE construction relying on the hardness of DBDH assumption in the random oracle model, where they partially outsource expensive operations such as re-encryption key generation and decryption to an outsourcing server. Subsequently, Qiu *et al*. [23] proposed an efficient CPRE scheme and prove its chosen-ciphertext security DBDH assumption in the random oracle model. In [19], Liang *et al*. provided a generic approach to convert Hierarchical Identity-Based Encryption (HIBE) schemes to CCA secure CPRE schemes, and also gave an instantiation of a CCA secure CPRE scheme in the standard model, based on Waters-HIBE scheme. Till date, the scheme due to Vivek *et al*. [30] is the most efficient CPRE protocol based on bilinear pairing. Certificate-based conditional PRE schemes have been proposed in [16, 17], whose security is reduced to the intractability of the DBDH problem in the random oracle model. Recently, Liu *et al*. [22] proposed the notion of multi-conditional proxy broadcast re-encryption(MC-PBRE), which allows re-encryption of ciphertexts to users satisfying any number of conditions from a set of predefined conditions. Their construction is CCA secure under the n-BDHE assumption in the standard model. As our focus is on CPRE schemes in the Public Key Infrastructure (PKI) setting, we omit the variants and extensions of CPRE, such as hierarchical conditional PRE [12], conditional broadcast PRE [9], outsourcing CPRE [27], type-based PRE [29], multi-conditional PRE [22], collusion-resistant PRE and attributed based PRE [20, 15] for fine grained delegation of decryption rights.

Note that all the above mentioned schemes rely on bilinear pairing operations, as stated. Pairing has been considered as one of the most expensive operations ever since its introduction in public key cryptographic primitives with respect to computational complexity and memory requirement. Pairing operations are costly when compared to modular arithmetic operations, and takes more than twice the time taken by modular exponentiation computation despite recent advances in implementation techniques [4]. To the best of our knowledge, no pairing-free CPRE scheme exists in the literature. In this work, we propose an efficient pairing-free unidirectional single-hop conditional proxy re-encryption scheme in the random oracle model, by extending the PKI based PRE construction in [8]. We also demonstrate the efficiency of our construction as compared to the most efficient pairing-based CPRE protocols, in Section 4. Our scheme satisfies CCA security in the random oracle model and is based on the Computational Diffie Hellman (*CDH*) assumption and its variant. A notable feature of our work is that our scheme provides modularity, in the essence that, the CPRE scheme can function as a traditional PRE scheme in the absence of any condition, with minor changes to the scheme.

2 Definition and Security Model

2.1 Definition

We describe the syntactical definition of unidirectional single-hop conditional proxy re-encryption and its security notion. Weng *et al*. [32] designed the first CPRE model that generates two different sets (re-encryption key and condition key) of keys pertaining to a condition. A more generalized approach proposed by Weng *et al*. [33] combines both keys into a single key used for re-encryption. We adopt the definition of unidirectional single-hop conditional proxy re-encryption from the work of Weng *et al*.

[33], as described next. In our framework, the re-encryption key is composed of two parts: an ephemeral *conditional* re-encryption key and a *permanent* re-encryption key. Between two users, the conditional re-encryption key changes with a change in the condition, whereas the permanent re-encryption key component remains unaltered throughout.

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

- **KeyGen($i, params$):** The key generation algorithm is a probabilistic algorithm which takes as input a user index i and public parameters *params*. It returns the public-private key pair (pk_i, sk_i) of user i.

- **ReKeyGen($sk_i, \omega, pk_i, pk_j,$ *params*):** The re-encryption key generation algorithm is a probabilistic algorithm which takes as input the private key sk_i, a condition ω, the public keys pk_i and pk_j of users i and j and public parameter *params*, and returns a re-encryption key $RK_{i \overset{\omega}{\rightarrow} j} = (RK^{\langle c \rangle}, RK^{\langle p \rangle})$ comprising of two components: $RK^{\langle c \rangle}$ the ephemeral conditional re-encryption key and $RK^{\langle p \rangle}$ the permanent re-encryption key.

- **Encrypt($pk_i, m, \omega, params$):** The encryption algorithm is a probabilistic algorithm which takes as input the public key pk_i of a user i, a plaintext $m \in \mathscr{M}$, a condition ω and public parameters *params* and returns a ciphertext C_i corresponding to m and condition ω which is allowed to be re-encrypted towards another user. The ciphertext C_i is termed as second-level ciphertext.

- **Re-Encrypt($RK_{i \overset{\omega}{\rightarrow} j}, C_i, params$):** The re-encryption algorithm is a probabilistic algorithm which takes as input a re-encryption key $RK_{i \overset{\omega}{\rightarrow} j}$, second level ciphertext C_i encrypted under pk_i associated with condition ω and public parameters *params*, and returns ciphertext D_j encrypted under the public key pk_j. The ciphertext D_j is termed as first level ciphertext.

- **Decrypt**$(sk_i, C_i, params)$**:** The decryption algorithm is a deterministic algorithm which takes as input the private key sk_i of a user i, a second level ciphertext C_i, and public parameters *params* and returns a plaintext m or the error symbol $\perp$ if the ciphertext is invalid.

- **Re-Decrypt($sk_j, D_j, params$):** The re-decryption algorithm is a deterministic algorithm which takes as input the private key sk_j of a user j, a first level ciphertext D_j, and public parameters *params* and returns a plaintext m or the error symbol $\perp$ if the ciphertext is invalid.

The consistency of a CPRE scheme for any given public parameters *params* and a public-private key pair $\{(pk_i, sk_i), (pk_j, sk_j)\}$ is defined as follows:

1. Consistency between encryption and decryption:

$$Decrypt(sk_i, (Encrypt(pk_i, m, \omega, params), params) = m, \ \forall m \in \mathscr{M}.$$

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

$$Re\text{-}Decrypt(sk_j, D_j, params) = m, \ \forall m \in \mathscr{M},$$

where $D_j \leftarrow Re\text{-}Encrypt(RK_{i \overset{\omega}{\rightarrow} j}, C_i, params)$ is a first level ciphertext and $C_i \leftarrow Encrypt(pk_i, m, \omega,$ *params*$)$ is a second level ciphertext.

2.2 Security Model

We define the notion of semantic security for our CPRE scheme under chosen ciphertext attack. Since there exists two levels of ciphertexts in a PRE scheme, namely first level and second level ciphertexts, it is essential to define two types of security notions corresponding to the two levels [21]. The semantic security against chosen ciphertext attack (CCA-security) is defined by the following game between an adversary $\mathscr{A}$ and challenger $\mathscr{C}$. The challenger C simulates an environment running PRE for the adversary A by providing $\mathscr{A}$ with access to the following oracles and answering the oracle queries issued by A as below:

- Uncorrupted key generation oracle $\mathscr{O}_u(i)$: $\mathscr{C}$ runs $KeyGen(i, params)$ algorithm to generate the public and private key pair (pk_i, sk_i) and returns pk_i.
- Corrupted key generation oracle $\mathscr{O}_c(i)$: $\mathscr{C}$ runs $KeyGen(i, params)$ to generate the public and private key pair (pk_i, sk_i) and returns (pk_i, sk_i).
- Re-key generation oracle $\mathscr{O}_{RK}(pk_i, \omega, pk_j)$: $\mathscr{C}$ obtains $RK_{i \overset{\omega}{\rightarrow} j} \leftarrow ReKeyGen(sk_i, \omega, pk_i, pk_j, params)$ and returns $RK_{i \overset{\omega}{\rightarrow} j}$ to $\mathscr{A}$.
- Re-encryption oracle $\mathscr{O}_{RE}(pk_i, pk_j, (C_i, \omega))$: C computes $D_j \leftarrow Re\text{-}Encrypt(RK_{i \overset{\omega}{\rightarrow} j}, C_i, params)$, where $RK_{i \overset{\omega}{\rightarrow} j} \leftarrow ReKeyGen(sk_i, \omega, pk_i, pk_j, params)$ and returns D_j to $\mathscr{A}$.
- Second level decryption oracle $\mathscr{O}_{Dec}(C_i, pk_i)$: C runs $Decrypt(sk_i, C_i, params)$ and returns the result to $\mathscr{A}$.
- First level decryption oracle $\mathscr{O}_{ReDec}(D_j, pk_j)$: C runs $Re\text{-}Decrypt(sk_j, D_j, params)$ and returns the result to $\mathscr{A}$.

The CCA(chosen ciphertext attack) security models for the two ciphertext levels of a single-hop unidirectional CPRE scheme are discussed next.

2.2.1 Second level ciphertext security:

Second level ciphertext security models the scenario where the adversary $\mathscr{A}$ is challenged with a ciphertext C_i^*, where C_i^* is the challenge ciphertext under the targeted public key pk_i* and a condition w^*. The CCA game is played in the following phases:

1. Setup: The challenger $\mathscr{C}$ runs the $Setup$ algorithm to generate the public parameters $params$ and returns $params$ to the adversary $\mathscr{A}$.
2. Phase 1: The adversary $\mathscr{A}$ adaptively issues queries to the above oracles simulated by the challenger $\mathscr{C}$, and $\mathscr{C}$ answers the queries.
3. Challenge: The adversary $\mathscr{A}$ outputs a target public key pk_i^*, a condition ω^* and two equal-length messages $m_0, m_1 \in \mathscr{M}$ with the following constraints such that $\mathscr{A}$ is unable to decrypt the challenge ciphertext trivially:
   (a) pk_i^* cannot be a corrupt user ($pk_i^* \notin CU$).
   (b) For a public key $pk_j \in CU$, $\mathscr{A}$ cannot issue query $O_{RK}(pk_i^*, w^*, pk_j)$.

   On receiving m_0, m_1, $\mathscr{C}$ selects $\delta \in \{0,1\}$ at random and generates challenge ciphertext $C_i^* \leftarrow Encrypt(pk_i^*, m_\delta, \omega^*, params)$. It returns C_i^* to $\mathscr{A}$.

4. Phase 2: $\mathscr{A}$ continues to issue queries to $\mathscr{C}$ as in Phase 1, with the below constraints such that $\mathscr{A}$ cannot decrypt challenge ciphertext trivially:
   (a) For a public key $pk_j \in CU$, $\mathscr{A}$ cannot issue query $O_{RE}(pk_i^*, pk_j, C^*, \omega^*)$.
   (b) For a public key pk_j and a first level ciphertext $D_j \leftarrow Re\text{-}Encrypt(RK_{i^* \xrightarrow{\omega} j}, C_i^*, params)$, $\mathscr{A}$ cannot issue query $O_{ReDec}(D_j, pk_j)$.
   (c) $\mathscr{A}$ cannot issue query $O_{Dec}(D_j^*, pk_i^*)$.
5. Guess: $\mathscr{A}$ outputs its guess $\delta' \in \{0,1\}$.

We define advantage of $\mathscr{A}$ in winning the game against second level ciphertext security as $Adv_{A,second}^{IND-CPRE-CCA} = |2Pr[\delta' = \delta] - 1|$, where the probability is over the random coin tosses performed by the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$. A single hop unidirectional CPRE scheme is said to be $(t,\varepsilon)IND-CPRE-CCA$ secure for second level ciphertexts if the advantage of any t-time adversary $\mathscr{A}$ that makes atmost q_u queries to the uncorrupted key-generation oracle $\mathscr{O}_u$, q_c queries to the corrupted key-generation oracle $\mathscr{O}_c$, q_{RK} queries to the re-encryption key-generation oracle $\mathscr{O}_{RK}$, q_{RE} queries to re-encryption oracle $\mathscr{O}_{RE}$, q_{Dec} queries to the decryption oracle $\mathscr{O}_{Dec}$ and q_{ReDec} queries to the re-decryption oracle $\mathscr{O}_{ReDec}$ is $Adv_{A,second}^{IND-CPRE-CCA} \leq \varepsilon$.

2.2.2 First level ciphertext security:

In the first level ciphertext security game, adversary $\mathscr{A}$ is challenged with a first-level ciphertext, without any knowledge of its corresponding second level ciphertext. The security game between adversary $\mathscr{A}$ and challenger $\mathscr{C}$ is demonstrated below in phases:

1. Setup: The challenger $\mathscr{C}$ runs the *Setup* algorithm to generate the public parameters *params* and returns *params* to the adversary $\mathscr{A}$.
2. Phase 1: The adversary $\mathscr{A}$ adaptively issues queries to the above oracles simulated by the challenger $\mathscr{C}$, and $\mathscr{C}$ responds to the queries.
3. Challenge: Adversary $\mathscr{A}$ outputs a delegator's public key pk_i', public key of delegatee (target user) pk_j^*, condition ω^* and two equal-length messages $m_0, m_1 \in \mathscr{M}$ with a constraint that pk_j^* cannot be a corrupt user ($pk_j^* \notin CU$). This prevents $\mathscr{A}$ to decrypt the challenge ciphertext trivially. On receiving messages m_0, m_1, challenger $\mathscr{C}$ selects $\delta \in \{0,1\}$ at random and generates a challenge first level ciphertext D_j^* and returns to $\mathscr{A}$.
4. Phase 2: $\mathscr{A}$ continues to issue queries to $\mathscr{C}$ as in Phase 1, with a constraint that for given challenge second level ciphertext, $\mathscr{A}$ cannot issue query $O_{ReDec}(D_j^*, pk_j^*)$. This prevents $\mathscr{A}$ to decrypt the challenge ciphertext D_j^* trivially.
5. Guess: $\mathscr{A}$ outputs its guess $\delta' \in \{0,1\}$.

We define advantage of $\mathscr{A}$ in winning the game against first level ciphertext security as $Adv_{A,first}^{IND-CPRE-CCA} = |2Pr[\delta' = \delta] - 1|$, where the probability is over the random coin tosses performed by $\mathscr{C}$ and $\mathscr{A}$. A single hop unidirectional CPRE scheme is said to be $(t,\varepsilon)IND-CPRE-CCA$ secure for first level ciphertexts if the advantage of any t-time adversary $\mathscr{A}$, that makes atmost q_u queries to oracle $\mathscr{O}_u$, q_c queries to oracle $\mathscr{O}_c$, q_{RK} queries to oracle $\mathscr{O}_{RK}$, q_{RE} queries to oracle $\mathscr{O}_{RE}$, q_{Dec} queries to oracle $\mathscr{O}_{Dec}$ and q_{ReDec} queries to oracle $\mathscr{O}_{ReDec}$ is $Adv_{A,first}^{IND-CPRE-CCA} \leq \varepsilon$.

Hardness Assumptions

Let $\mathbb{G}$ be a cyclic group with a prime order q.

Definition 1. ***Computational Diffie-Hellman (CDH) assumption*** *: The Computational Diffie-Hellman (CDH) assumption for group $\mathbb{G}$ says that, given the elements $\{g, g^a, g^b\} \in \mathbb{G}$, there exists no probabilistic polynomial-time algorithm which can compute $g^{ab} \in \mathbb{G}$ with a non-negligible advantage, where g is a generator of $\mathbb{G}$ and $a, b \in_R \mathbb{Z}_q^*$.*

Definition 2. ***Modified Computational Diffie-Hellman (M-CDH) assumption*** *[28] : The Modified Computational Diffie-Hellman (M-CDH) assumption for group $\mathbb{G}$ says that, given the elements $\{g, g^a, g^b\} \in \mathbb{G}$, there exists no probabilistic polynomial-time algorithm which can compute $g^{b/a} \in \mathbb{G}$ with a non-negligible advantage, where g is a generator of $\mathbb{G}$ and $a, b \in_R \mathbb{Z}_q^*$.*

3 Our Unidirectional CCA-secure CPRE Scheme

3.1 Our Scheme

- **Setup(λ):** Let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_q^*$ with order q, where q is a prime. Let g be a generator of the group $\mathbb{G}$. Choose the following cryptographic hash functions: $H_1 : \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_q^*, H_2 : \mathbb{Z}_q^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^*, H_3 : \mathbb{G} \to \{0,1\}^{\log 2q}, H_4 : \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to \mathbb{Z}_q^*, H_5 : \mathbb{G} \to \{0,1\}^{l_0+l_1}, H_6 : \mathbb{G}^4 \times \{0,1\}^{l_0+l_1} \to \mathbb{G}, H_7 : \mathbb{G}^2 \times \{0,1\}^{l_0+l_1} \to \mathbb{Z}_q^*$. The message space $\mathscr{M}$ is $\{0,1\}^{l_0}$ and l_1 is determined by security parameter λ. Return public parameters $params = (\mathbb{G}, q, g, H_1, H_2, H_3, H_4, H_5, H_6, H_7, l_0, l_1)$.
- **KeyGen($i, params$):** On input of user index i and security parameter $params$, compute the private key and its corresponding public key as below:
  - Pick $x_{i,1},\ x_{i,2} \in_R \mathbb{Z}_q^*$. Set private key $sk_i = (x_{i,1}, x_{i,2})$.
  - Compute public key $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$.
  - Return the public-private key pair (pk_i, sk_i).
- **ReKeyGen($sk_i, \omega, pk_i, pk_j,\ params$):** On input of delegator's private key sk_i, condition ω, delegator's public key pk_i and delegatee's public key pk_j, generate re-encryption key as below:
  - Pick $z_1, h \in_R \mathbb{Z}_q^*$. Compute $v = H_2(z_1, h)$.
  - Compute permanent re-encryption key component $RK^{\langle p \rangle} = \{V, W\}$, where $V = pk_{j,2}^v$, $W = H_3(g^v) \oplus (z_1 || h)$. Note that component $RK^{\langle p \rangle}$ remains fixed for a pair of delegator and delegatee (pk_i, pk_j), throughout all re-encryption key generation for multiple conditions.
  - Compute $z = \frac{1}{x_{i,1} H_1(pk_{i,2}, \omega) + x_{i,2}}$. Note that the value z changes with a change in condition ω for a pair of delegator and delegatee (pk_i, pk_j).
  - Compute z_2 such that $z_1 \cdot z_2 = z \mod q$. Set conditional re-key component $RK^{\langle c \rangle} = z_2$.
  - Return $RK_{i \xrightarrow{\omega} j} = (RK^{\langle c \rangle}, RK^{\langle p \rangle}) = (z_2, V, W)$.

  Remark 1. *When condition ω between users i and j changes, only conditional re-encryption key z_2 alone needs to be recomputed for a fresh value of z and sent to proxy to avoid re-computation. All other components (z_1, v, V, W) remains unaltered and can be computed once and stored locally.*

  Remark 2. *In the absence of condition ω, by changing the hash function description H_1 to $H_1 : \mathbb{G} \to \mathbb{Z}_q^*$, the CPRE scheme functions as a traditional PRE scheme, where z is computed as $z = \frac{1}{x_{i,1} H_1(pk_{i,2}) + x_{i,2}}$. All other computations remain unaffected.*

- **Encrypt**$(pk_i, m, \omega, params)$**:** To encrypt message m under public key pk_i based on condition ω:
  - Pick $u \in \mathbb{Z}_q^*$, $r' \in \{0,1\}^{l_1}$. Set $r = H_4(m, r')$.
  - Compute $D = (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^u$.
  - Compute $E = (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^r$.
  - Compute $F = H_5(g^r) \oplus (m||r')$.
  - Compute $\overline{D} = H_6(pk_i, D, E, F)^u$.
  - Compute $\overline{E} = H_6(pk_i, D, E, F)^r$.
  - Compute $s = u + r \cdot H_7(D, \overline{E}, F) \mod q$.
  - Return the second level ciphertext $C_i = (D, \overline{E}, F, s, \omega)$.

  Remark 3. *In the absence of condition ω, by changing H_1 hash function description to $H_1 : \mathbb{G} \to \mathbb{Z}_q^*$, the CPRE scheme functions as a traditional PRE scheme. Ciphertext components D, E are computed as $D = (pk_{i,1}^{H_1(pk_{i,2})} pk_{i,2})^u$, $E = (pk_{i,1}^{H_1(pk_{i,2})} pk_{i,2})^r$. All other computations remain unaffected.*

- **Re-Encrypt**$(RK_{i \xrightarrow{\omega} j}, C_i, \ params)$**:** To re-encrypt a second-level ciphertext C_i under the public key pk_j using the re-encryption key $RK_{i \xrightarrow{\omega} j}$, parse C_i as $(D, \overline{E}, F, s, \omega)$ and $RK_{i \xrightarrow{\omega} j}$ as (z_2, V, W). Perform the following computations:
  - Compute the terms E and $\overline{D}$ as follows:

$$\begin{aligned}
E &= \left(\left(pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2}\right)^s \cdot D^{-1}\right)^{H_7(D,\overline{E},F)^{-1}} \\
&= \left(\left(pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2}\right)^{u+r\cdot H_7(D,\overline{E},F)} \cdot \left(pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2}\right)^{-u}\right)^{H_7(D,\overline{E},F)^{-1}} \\
&= \left(pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2}\right)^r. \\
\overline{D} &= H_6(pk_i, D, E, F)^s \cdot (\overline{E}^{H_7(D,\overline{E},F)})^{-1} \\
&= H_6(pk_i, D, E, F)^{u+r\cdot H_7(D,\overline{E},F)} \cdot \left(H_6(pk_i, D, E, F)^{rH_7(D,\overline{E},F)}\right)^{-1} \\
&= H_6\left(pk_i, D, E, F\right)^u.
\end{aligned}$$

  - Check condition satisfiability of ω for re-encryption by verifying if:

$$\left(pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2}\right)^s \stackrel{?}{=} D \cdot E^{H_7(D,\overline{E},F)} \tag{1}$$

    If above check fails, ciphertext is ill-formed and condition ω does not pertain to other ciphertext components $(D, \overline{E}, F, s)$. Hence, return $\perp$.
  - Check if the ciphertext is well-formed by verifying the following equation:

$$H_6(pk_i, D, E, F)^s \stackrel{?}{=} \overline{D} \cdot \overline{E}^{H_7(D,\overline{E},F)} \tag{2}$$

    If it fails, ciphertext is ill-formed and cannot be transformed. Return $\perp$.
  - If both conditions hold, compute $E' = E^{z_2}$.
  - Return the first level ciphertext $D_j = (E', F, V, W)$.

- **Decrypt**$(sk_i, C_i, params)$**:** On input of a private key sk_i and a second level ciphertext C_i, parse C_i as $(E, \overline{E}, F, s, \omega)$, decrypt as below:
  - Check if the ciphertext is well-formed by computing E and $\overline{D}$ and verifying if equations (1) and (2) hold. If they do not hold, return $\perp$.
  - Else, compute:
$$(m||r') = H_5(E^{\frac{1}{x_{i,1}H_1(pk_{i,2},\omega)+x_{i,2}}}) \oplus F. \tag{3}$$
  - Return m if $E \stackrel{?}{=} (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{H_4(m,r')}$ holds.
- **Re-Decrypt**$(sk_j, D_j, params)$**:** On input of a private key sk_j and a first level ciphertext D_j, parse D_j as (E', F, V, W) and decrypt as below:
  - Compute $(z_1||h) = H_3(V^{\frac{1}{x_{j,2}}}) \oplus W$.
  - Check if $V \stackrel{?}{=} pk_{j,2}^{H_2(z_1,h)}$. If the condition does not hold, return $\perp$.
  - Otherwise, compute:
$$(m||r') = H_5(E'^{z_1}) \oplus F. \tag{4}$$
  - If $F \stackrel{?}{=} (m||r') \oplus H_5(g^{H_4(m||r')})$, return m.

3.2 Correctness

- Correctness of equation (1) for condition satisfiability:
$$\begin{aligned} RHS &= D \cdot E^{H_7(D,\overline{E},F)} \\ &= (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^u \cdot (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{rH_7(D,\overline{E},F)} \\ &= (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{u+rH_7(D,\overline{E},F)} \\ &= (pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^s \\ &= LHS. \end{aligned}$$
- Correctness of equation (2) for ciphertext verification:
$$\begin{aligned} RHS &= \overline{D} \cdot \overline{E}^{H_7(D,\overline{E},F)} \\ &= H_6(pk_i, D, E, F)^u \cdot H_6(pk_i, D, E, F)^{rH_7(D,\overline{E},F)} \\ &= H_6(pk_i, D, E, F)^{u+rH_7(D,\overline{E},F)} \\ &= H_6(pk_i, D, E, F)^s \\ &= LHS. \end{aligned}$$
- Correctness of equation (3) for decryption of second-level ciphertext:
$$\begin{aligned} RHS &= H_5(E^{\frac{1}{x_{i,1}H_1(pk_{i,2},\omega)+x_{i,2}}}) \oplus F \\ &= H_5((pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{\frac{r}{x_{i,1}H_1(pk_{i,2},\omega)+x_{i,2}}}) \oplus (m||r') \oplus H_5(g^r) \\ &= H_5(g^r) \oplus (m||r') \oplus H_5(g^r) \\ &= (m||r') \\ &= LHS. \end{aligned}$$

- Correctness of equation (4) for decryption of first-level ciphertext:

$$\begin{aligned} RHS &= H_5(E'^{z_1}) \oplus F \\ &= H_5((pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{rz_2})^{z_1} \oplus (m||r') \oplus H_5(g^r) \\ &= H_5((pk_{i,1}^{H_1(pk_{i,2},\omega)} pk_{i,2})^{\frac{r}{x_{i,1}H_1(pk_{i,2},\omega)+x_{i,2}}}) \oplus (m||r') \oplus H_5(g^r) \\ &= (m||r') \\ &= LHS. \end{aligned}$$

3.3 Security Proof

3.3.1 Second Level Ciphertext Security:

Theorem 1. *Our CPRE scheme is IND-CPRE-CCA secure for the second ciphertext in the random oracle model, assuming M-CDH assumption holds in group $\mathbb{G}$ and the Schnorr signature [24] is $EUF-CMA$ secure. If there exists a (t,ε)IND-CPRE-CCA $\mathscr{A}$ who breaks the IND-CPRE-CCA security of the given scheme with an advantage ε in time t, then there exists an algorithm $\mathscr{C}$ that can solve the M-CDH problem with advantage ε' within time t' where:*

$$\varepsilon' \geq \frac{1}{q_{H_5}}\left(\frac{\varepsilon}{e(q_{RK}+1)} - \frac{q_{H_4}}{2^{l_1}} - \frac{q_{H_6}+q_{H_7}}{2^{l_0+l_1}} - q_{Dec}\left(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\right)\right),$$
$$t' \leq t + (q_{H_6} + 2q_u + 2q_c + 2q_{RK} + 8q_{RE} + 6q_{Dec} + 4q_{ReDec})t_e,$$

where e is the base of natural logarithm, t_e is the time taken for exponentiation operation in group $\mathbb{G}$. The number of queries to any oracle is a polynomial in the security parameter.

Proof. If there exists a t-time adversary $\mathscr{A}$ who can break the (t,ε)*IND-CPRE-CCA* security of our scheme, we show how to construct a polynomial time algorithm $\mathscr{C}$ who can break the *M-CDH* assumption in $\mathbb{G}$ or the existential unforgeability against chosen message attack (EUF-CMA) of the Schnorr Signature with non-negligible advantage. $\mathscr{C}$ is given an instance of M-CDH challenge tuple (g, g^a, g^b), with $a, b \in_R \mathbb{Z}_q^*$ as input, whose goal is to compute $g^{b/a}$. $\mathscr{C}$ acts as a challenger and plays the IND-CPRE-CCA game with the $\mathscr{A}$ as described next. $\mathscr{C}$ maintains a list L_K to store information about all user keys consisting of tuples $\langle (pk_i), x_{i,1}, x_{i,2}, c_i \rangle$. $\mathscr{C}$ also maintains a list L_{RK} to store information about all the generated re-encryption keys consisting of tuples $\langle (pk_i), (pk_j), \omega, z_2, V, W, \tau \rangle$.

– **Phase 1 :** $\mathscr{C}$ answers the queries issues by $\mathscr{A}$ to the oracles as follows:

- Uncorrupt Key Generation Query $\mathscr{O}_u(i)$: Apply Coron's technique [10] to generate uncorrupt keys by flipping a coin that takes value $c_i \in \{0,1\}$, where $c_i = 1$ is obtained with probability ρ, which is to be determine later. Compute pk_i according to the below cases:
  - If $c_i = 1$, pick $x_{i,1}, x_{i,2} \in_R \mathbb{Z}_q^*$ and compute $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$. Update L_K with the tuple $\langle (pk_i), x_{i,1}, x_{i,2}, c_i = 1 \rangle$.
  - If $c_i = 0$, pick $x_{i,1}, x_{i,2} \in_R \mathbb{Z}_q^*$ and compute $pk_i = (pk_{i,1}, pk_{i,2}) = ((g^a)^{x_{i,1}}, (g^a)^{x_{i,2}})$. Update L_K with the tuple $\langle (pk_i), x_{i,1}, x_{i,2}, c_i = 0 \rangle$.

  It is straightforward to note that $\mathscr{C}$ performs atmost 2 exponentiation operations for every invocation to the uncorrupt key generation oracle.
- Corrupt Key Generation Query $\mathscr{O}_c(i)$: Pick $x_{i,1}, x_{i,2} \in_R \mathbb{Z}_q^*$ and compute the public key $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_{i,1}}, g^{x_{i,2}})$. Update L_K with the tuple $\langle (pk_i), x_{i,1}, x_{i,2}, c_i = - \rangle$.
  It is straightforward to note that $\mathscr{C}$ performs atmost 2 exponentiation operations for every invocation to the corrupt key generation oracle.

- Hash Queries :
-H_1 *query*: If the query $H_1(pk_{i,2},\omega)$ has been placed on list L_{H_1} in a tuple $\langle pk_{i,2}\in\mathbb{G},\omega\in\{0,1\}^*,\delta\in\mathbb{Z}_q^*\rangle$, return the predefined value $H_1(pk_{i,2},\omega)=\delta$. Else pick $\delta\in_R\mathbb{Z}_q^*$, update L_{H_1} with the tuple $\langle pk_{i,2},\omega,\delta\rangle$ and respond with $H_1(pk_{i,2},\omega)=\delta$.

-H_2 *query*: If the query $H_2(z_1,h)$ has been placed on list L_{H_2} in a tuple $\langle z_1\in\mathbb{Z}_q^*,h\in\mathbb{Z}_q^*,v\in\mathbb{Z}_q^*\rangle$, return the predefined value $H_2(z_1,h)=v$. Else pick $v\in_R\mathbb{Z}_q^*$, update L_{H_2} with the tuple $\langle z_1,h,v\rangle$ and respond with $H_2(z_1,h)=v$.

-H_3 *query*: If the query $H_3(R)$ has been placed on the list L_{H_3} in a tuple $\langle R\in\mathbb{G},\kappa\in\{0,1\}^{\log 2q}\rangle$, return the predefined value $H_3(R)=\kappa$. Else pick $\kappa\in_R\{0,1\}^{\log 2q}$, update list L_{H_3} with $\langle R,\kappa\rangle$ and return $H_3(R)=\kappa$.

-H_4 *query*: If the query $H_4(m,r')$ has been placed on the list L_{H_4} in a tuple $\langle m\in\{0,1\}^{l_0},r'\in\{0,1\}^{l_1},r\in\mathbb{Z}_q^*\rangle$, return the predefined value $H_4(m,r')=r$. Else pick $r\in_R\mathbb{Z}_q^*$, update list L_{H_4} with the tuple $\langle m,r',r\rangle$ and respond with $H_4(m,r')=r$.

-H_5 *query*: If the query $H_5(R')$ has been placed on the list L_{H_5} in a tuple $\langle R'\in\mathbb{G},\psi\in\{0,1\}^{l_0+l_1}\rangle$, return the predefined value $H_5(R')=\psi$. Else pick $\psi\in_R\{0,1\}^{l_0+l_1}$, update list L_{H_5} with the tuple $\langle R',\psi\rangle$ and respond with $H_5(R')=\psi$.

-H_6 *query*: If the query $H_6(pk_i,D,E,F)$ has been placed on the list L_{H_6} in a tuple $\langle pk_i\in\mathbb{G}^2,D\in\mathbb{G},E\in\mathbb{G},F\in\{0,1\}^{l_0+l_1},\alpha\in\mathbb{Z}_q^*,\beta\in\mathbb{G}\rangle$, return the predefined value $H_6(pk_i,D,E,F)=\beta$. Else pick $\alpha\in_R\mathbb{Z}_q^*$, compute $\beta=g^\alpha$, update list L_{H_6} with the tuple $\langle pk_i,D,E,F,\alpha,\beta\rangle$ and respond with $H_6(pk_i,D,E,F)=\beta$.

-H_7 *query*: If the query $H_7(D,\overline{E},F)$ has been placed on the list L_{H_7} in a tuple $\langle D\in\mathbb{G},\overline{E}\in\mathbb{G},F\in\{0,1\}^{l_0+l_1},\theta\in\mathbb{Z}_q^*\rangle$, return the predefined value $H_7(D,\overline{E},F)=\theta$. Else pick $\theta\in_R\mathbb{Z}_q^*$, update list L_{H_7} with the tuple $\langle D,\overline{E},F,\theta\rangle$ and respond with $H_7(D,\overline{E},F)=\theta$.

- Re-encryption key generation query $\mathcal{O}_{RK}(pk_i,\omega,pk_j)$: Search list L_{RK} for a tuple $\langle(pk_i),(pk_j),\omega,z_2,V,W,\tau\rangle$ to check the existence of re-encryption key from pk_i to pk_j. If present, return re-encryption key $RK_{i\overset{\omega}{\to}j}=(z_2,V,W)$. Otherwise, compute re-encryption keys according to the following cases:
  - If $c_i=0$ and $c_j=-$, output "failure" and abort.
  - If $c_i\in\{1,-\}$, generate the re-encryption key as per re-encryption key generation algorithm:
    - Compute $z=\frac{1}{x_{i,1}H_1(pk_{i,2},\omega)+x_{i,2}}$.
    - Pick $h,z_1\in_R\mathbb{Z}_q^*$. Compute conditional re-encryption key z_2 such that $z_1\cdot z_2=z \bmod q$.
    - Compute $v=H_2(z_1,h)$. Compute the permanent re-encryption key components $V=pk_{j,2}^v$, $W=H_3(g^v)\oplus(z_1||h)$.
    - Update list L_{RK} with the tuple $\langle(pk_i),(pk_j),\omega,z_2,V,W,\tau=-\rangle$.
  - If $c_i=0$ and $c_j\in\{0,1\}$, generate the re-encryption key as per re-encryption key generation algorithm:
    - Pick $z_1,\ z_2,h\in_R\mathbb{Z}_q^*$.
    - Compute $v=H_2(z_1,h)$, $V=pk_{j,2}^v$, $W=H_3(g^v)\oplus(z_1||h)$.

· Update list L_{RK} with the tuple $\langle (pk_i),(pk_j),\omega,z_2,V,W,\tau=0\rangle$.

Note that $\mathscr{C}$ performs atmost 2 exponentiation operations for every invocation to re-encryption key generation oracle.

- Re-Encryption query $\mathscr{O}_{RE}(pk_i,pk_j,C_i)$: Check for condition satisfiability and ciphertext consistency by computing E and $\overline{D}$ and verifying if equations (1) and (2) hold. If they fail to hold, return $\perp$. Else, compute re-encrypted ciphertext D_j as per the following cases:
  - If $(c_i=0 \ \wedge \ c_j=-)$, search in list L_{H_4} for a tuple $\langle m,r',r\rangle$ such that $(pk_{i,1}^{H_4(pk_{i,2},\omega)} \cdot pk_{i,2})^r \stackrel{?}{=} E$ holds. Check list L_{RK} for a tuple $\langle (pk_i),(pk_j),\omega,z_2,V,W,\tau=1\rangle$. If such a tuple does not exist, compute the re-encryption key as shown :
    · Pick $z_1,\gamma \in_R \mathbb{Z}_q^*$. Set the conditional re-key $z_2=(z_1)^{-1}\cdot\gamma$.
    · Pick $h \in_R \mathbb{Z}_q^*$, compute $v=H_2(z_1\cdot\gamma^{-1},h)$.
    · Compute $V=pk_{j,2}^v$, $W=H_3(g^v)\oplus(z_1\cdot\gamma^{-1}||h)$.
    · Update list L_{RK} with the tuple $\langle (pk_i),(pk_j),\omega,z_2,V,W,\tau=1\rangle$.

    Compute $E'=g^{rz_2}$ and return $D_j=(E',F,V,W)$.
  - For all other values of c_i and c_j, first check the list L_{RK} for the presence of a tuple $\langle pk_i,pk_j,\omega,r,V,W,\tau\rangle$. If such a tuple does not exist, generate a new re-encryption key by invoking the re-key generation oracle $\mathscr{O}_{RK}(pk_i,\omega,pk_j)$ and further calling the re-encryption algorithm $Re\text{-}Encrypt(RK_{i\stackrel{\omega}{\to}j},C_i,params)$ to obtain D_j. Return the re-encrypted ciphertext D_j to $\mathscr{A}$.

  It is straightforward to note that $\mathscr{C}$ performs atmost 8 exponentiation operations for every invocation to the re-encryption oracle.
- Second level decryption query $\mathscr{O}_{Dec}(C_i,pk_i)$: Compute E, $\overline{D}$ and verify if equations (1) and (2) hold to check well-formedness. If they fail, return $\perp$. Else, if $c_i=-$ or $c_i=1$, run $Decrypt(sk_i,C_i,params)$ to decrypt C_i and return m. Else, if $c_i=0$, decrypt as below:
  - Retrieve tuple $\langle pk_i,D,E,F,\alpha,\beta\rangle$ from L_{H_6}.
  - Extract $g^r=(\overline{E})^{\frac{1}{\alpha}}$. Obtain the plaintext $(m||r')=F\oplus H_5(g^r)$. Return m.

  It is straightforward to note that $\mathscr{C}$ performs atmost 6 exponentiation operations for every invocation to the second level decryption oracle.
- First level decryption query $\mathscr{O}_{ReDec}(D_j,pk_j)$: Check the re-key list L_{RK} for the existence of a tuple $\langle (pk_i),(pk_j),\omega,z_2,V,W,\tau=0\rangle$. Decrypt according to the following two scenarios:
  - If such a tuple exists, compute $E=(E')^{\frac{1}{z_2}}$. Search L_{H_4} and L_{H_5} for tuples $\langle m,r',r\rangle$ and $\langle R',\psi\rangle$ respectively that satisfies:

    $$R' \stackrel{?}{=} g^r, \left(pk_{i,1}^{H_1(pk_{i,2},\omega)}pk_{i,2}\right)^r \stackrel{?}{=} E, \psi\oplus(m||r') \stackrel{?}{=} F.$$

    If all conditions hold, return m, else return $\perp$.
  - If such a tuple does not exist, search lists L_{H_2}, L_{H_3}, L_{H_5} and L_{H_4} for tuples $\langle z_1,h,v\rangle$, $\langle R,\kappa\rangle$, $\langle R',\psi\rangle$ and $\langle m,r',r\rangle$ respectively such that the following conditions hold:

    $$R \stackrel{?}{=} g^v, pk_{i,2}^v \stackrel{?}{=} V, \kappa\oplus(z_1||h) \stackrel{?}{=} W,$$
    $$R' \stackrel{?}{=} g^r, \psi\oplus(m||r') \stackrel{?}{=} F.$$

  If all conditions hold, return m. Else, return $\perp$.

It is straightforward to note that $\mathscr{C}$ performs atmost 4 exponentiation operations for every invocation to the first level decryption oracle.

– **Challenge :** $\mathscr{A}$ outputs target public key pk_i^*, condition ω^* and two messages $m_0, m_1 \in_R \{0,1\}^{l_0}$. $\mathscr{C}$ recovers tuple $\langle pk_i^*, x_{i,1}^*, x_{i,2}^*, c_i^* \rangle$ from list L_K and responds as follows:

1. If $c_i^* = 1$, report failure and abort. Else, if $c_i^* = 0$, select $\delta \in_R \{0,1\}$.
2. Choose $r'^* \in_R \{0,1\}^{l_1}$ and implicitly define $H_4(m_\delta, r'^*) = b/a$.
3. Select $s^*, t^* \in_R \mathbb{Z}_q^*$ and define $u \triangleq s^* - \frac{b}{a}t^*$.
4. Compute $D^* = g^{a(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)s^*} \cdot g^{b(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)(-t^*)}$.
5. Compute $E^* = (g^b)^{x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*}$.
6. Check list L_{H_6} for the existence of a tuple $\langle pk_{i^*}, D^*, E^*, F^*, \alpha', \beta \rangle$ or $\langle pk_{i^*}, D^*, E^*, F^*, \alpha, \beta' \rangle$. If such a tuple exists, recompute from step (2) with fresh random values.
7. Set $H_6(pk_{i^*}, D^*, E^*, F^*) = \beta$ where $\beta = (g^a)^\alpha,\ \alpha \in_R \mathbb{Z}_q^*$. Update L_{H_6} with new values.
8. Compute $\overline{E}^* = (g^b)^\alpha$, where α is obtained from step (8).
9. Pick $F^* \in_R \{0,1\}^{l_0+l_1}$. Set $H_7(D^*, \overline{E}^*, F^*) = t^*$.
10. Implicitly define $H_5(g^{b/a}) = (m_\delta || r'^*) \oplus F^*$.
11. Return the challenge ciphertext $C_i^* = (D^*, \overline{E}^*, F^*, s^*, \omega^*)$.

Note that challenge ciphertext C_i^* is identically distributed as the original ciphertext generated from encryption algorithm, as shown below:

$$
\begin{aligned}
D^* &= g^{a(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)s^*} g^{b(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)(-t^*)} \\
&= g^{a(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)(s^* - \frac{b}{a}t^*)} \\
&= \left(pk_{i^*,1}^{H_1(pk_{i^*,2})} pk_{i^*,2}\right)^u. \\
E^* &= g^{b(x_{i,1}^* H_1(pk_{i,2}^*, \omega^*) + x_{i,2}^*)} \\
&= \left(g^{a x_{i,1}^* H_1(pk_{i,2}^*, \omega) + a x_{i,2}^*}\right)^{\frac{b}{a}} \\
&= \left(pk_{i^*,1}^{H_1(pk_{i^*,2}, \omega^*)} pk_{i^*,2}\right)^r. \\
\overline{E}^* &= (g^b)^\alpha = (g^{a \cdot \alpha})^{\frac{b}{a}} = H_6(pk_{i^*}, D^*, E^*, F^*)^r. \\
F^* &= H_5(g^{b/a}) \oplus (m_\delta || r') = H_5(g^r) \oplus (m_\delta || r'). \\
s^* &= u + \frac{b}{a} \cdot t^* = u + r^* \cdot H_7(E^*, \overline{E}^*, F^*).
\end{aligned}
$$

– **Phase 2 :** $\mathscr{A}$ continues to issue queries to $\mathscr{C}$ as in Phase 1, with the restrictions described in IND-CPRE-CCA game.

– **Guess :** Eventually, $\mathscr{A}$ outputs its guess $\delta' \in_R \{0,1\}$ to $\mathscr{C}$. The challenger $\mathscr{C}$ randomly picks a tuple $\langle R', \psi \rangle$ from L_{H_5} list and outputs R' as the solution to the M-CDH problem instance.

– **Probability Analysis:** Next, we proceed to the analysis of the simulation. The main idea of the proof is borrowed from [8]. First, we evaluate the simulation of the random oracles. The simulations of the oracles are perfect unless the following events take place:

  – E_{4^*}: (m_δ, r'^*) is queried to H_4.

- E_{5^*}: $(g^{b/a})$ is queried to H_5.
- E_{6^*}: $(pk_{i^*}, D^*, E^*, F^*)$ is queried to H_6 before Challenge phase.
- E_{7^*}: $(D^*, \overline{E}^*, F^*)$ is queried to H_7 before Challenge phase.

As $\mathscr{C}$ chooses $F^* \in_R \{0,1\}^{l_0+l_1}$, $Pr[E_6] \leq \frac{q_{H_6}}{2^{l_0+l_1}}$ and $Pr[E_7] \leq \frac{q_{H_7}}{2^{l_0+l_1}}$.
The simulation for corrupted key generation and uncorrupted key generation is perfect. We evaluate the simulation of the re-encryption and decryption oracles. We denote the event that $\mathscr{C}$ aborts the game during the conditional re-key generation or Challenge phase using *Abort*. Note that in both phases, C does not abort in case of the following events:

- E_1: $c_i = 1$ in a re-encryption key generation query.
- E_2: $c_i^* = 0$ in the Challenge phase.

Then we have $Pr[\neg Abort] \geq \rho^{q_{RK}}(1-\rho)$, which is maximised at $\rho_{OPT} = \frac{q_{RK}}{1+q_{RK}}$. Using ρ_{OPT}, we obtain probability $Pr[\neg Abort]$ is atleast $\frac{1}{e(1+q_{RK})}$, where e is the base of natural logarithm.
We analyse the simulation of the decryption oracle. The simulation of the decryption oracle is perfect unless the simulation errs by rejecting valid ciphertexts, i.e., a valid ciphertext C_i is produced without querying g^r to H_5 where $r = H_4(m, r')$. We use the notations for the following events: E_{valid} to denote that the ciphertext is a valid ciphertext, E_4 to denote that (m, r') is queried to H_4 and E_5 to denote that g^r is queried to H_5. We estimate $Pr[E_{valid}|\neg E_4 \vee \neg E_5] \leq Pr[E_{valid}|\neg E_4] + Pr[E_{valid}|\neg E_5]$:

$$\begin{aligned} Pr[E_{valid}|\neg E_4] &= Pr[E_{valid} \wedge E_5|\neg E_4] + Pr[E_{valid} \wedge \neg E_5|\neg E_4] \\ &\leq Pr[E_5|\neg E_4] + Pr[E_{valid}|\neg E_5 \wedge \neg E_4] \\ &\leq \frac{q_{H_5}}{2^{l_0+l_1}} + \frac{1}{q} \end{aligned}$$

Similarly, we obtain $Pr[E_{valid}|\neg E_5] \leq \frac{q_{H_4}}{2^{l_0+l_1}} + \frac{1}{q}$. Let E_{derr} denote the event that $E_{valid}|(\neg E_4 + \neg E_5)$ happens for the entire simulation. Since the adversary can pose atmost q_{Dec} decryption queries, we obtain $Pr[E_{derr}] \leq q_{Dec}\left(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\right)$.
We use E_{err} to represent the event $(E_{4^*} \vee E_{5^*} \vee E_{6^*} \vee E_{7^*} \vee E_{derr})|\neg Abort$. If E_{err} does not happen, the adversary $\mathscr{A}$ has no advantage greater than $\frac{1}{2}$ in guessing the bit δ owing to to the randomness in the output of H_5. The probability that the adversary correctly guesses δ is:

$$\begin{aligned} Pr[\delta' = \delta] &= Pr[\delta' = \delta|\neg E_{err}]Pr[\neg E_{err}] + Pr[\delta' = \delta|E_{err}]Pr[E_{err}] \\ &= \frac{1}{2}(1 + Pr[E_{err}]). \end{aligned}$$

And, $Pr[\delta' = \delta] \geq Pr[\delta' = \delta|\neg E_{err}]Pr[\neg E_{err}] \geq \frac{1}{2}(1 - Pr[E_{err}])$.
From the definition of the advantage of IND-CPRE-CCA adversary, we have:

$$\begin{aligned} \varepsilon &= |2Pr[\delta' = \delta] - 1| \\ &\leq Pr[E_{err}] \\ &= Pr[(E_{4^*} \vee E_{5^*} \vee E_{6^*} \vee E_{7^*} \vee E_{derr})|\neg Abort] \\ &\leq \frac{Pr[E_{4^*}] + Pr[E_{5^*}] + Pr[E_{6^*}] + Pr[E_{7^*}] + Pr[E_{derr}]}{Pr[\neg Abort].} \end{aligned}$$

We have $Pr[E_{4^*}] \leq \frac{q_{H_4}}{2^{l_1}}$ since the challenger $\mathscr{C}$ picks $r' \in_R \{0,1\}^{l_1}$, and we obtain the bound on $Pr[E_{5^*}]$:

$$\begin{aligned} Pr[E_{5^*}] &\geq Pr[\neg Abort] \cdot \varepsilon - Pr[E_{4^*}] - Pr[E_{6^*}] - Pr[E_{7^*}] - Pr[E_{derr}] \\ &\geq \frac{\varepsilon}{e(q_{RK}+1)} - \frac{q_{H_4}}{2^{l_1}} - \frac{q_{H_6}}{2^{l_0+l_1}} - \frac{q_{H_7}}{2^{l_0+l_1}} - q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big) \end{aligned}$$

If $\mathscr{A}$ breaks the IND-CPRE-CCA security of the scheme, then $\mathscr{C}$ can solve the $M-CDH$ instance with the advantage:

$$\varepsilon' \geq \frac{1}{q_{H_5}}\left(\frac{\varepsilon}{e(q_{RK}+1)} - \frac{q_{H_4}}{2^{l_1}} - \frac{q_{H_6}+q_{H_7}}{2^{l_0+l_1}} - q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big)\right)$$

The reduction involves asking q_{H_6}, q_u, q_c, q_{RK}, q_{RE}, q_{Dec} and q_{ReDec} queries to the H_6 hash function, uncorrupted key-generation, corrupted key-generation, re-encryption key generation, re-encryption, second level decryption and first level decryption oracles, and the number of exponentiations done under these queries are $1,2,2,2,8,6$ and 4 respectively. Hence, the total number of exponentiation operations done is $(q_{H_6}+2q_u+2q_c+2q_{RK}+8q_{RE}+6q_{Dec}+4q_{ReDec})t_e$ where t_e is the time taken for one exponentiation operation. We can bound the running time of $\mathscr{C}$ to solve the hard instance as below:

$$t' \leq t + (q_{H_6}+2q_u+2q_c+2q_{RK}+8q_{RE}+6q_{Dec}+4q_{ReDec})t_e.$$

This completes the proof of the theorem.

□

Theorem 2. *Our CPRE scheme is IND-CPRE-CCA secure for the first ciphertext in the random oracle model, assuming the CDH assumption holds in group $\mathbb{G}$ and the Schnorr signature [24] is $EUF-CMA$ secure. If there exists a (t,ε)IND-CPRE-CCA $\mathscr{A}$ who breaks the IND-CPRE-CCA security of the given scheme with an advantage ε in time t, then there exists an algorithm $\mathscr{C}$ that can solve the CDH problem with advantage ε' within time t' where:*

$$\begin{aligned} \varepsilon &\geq \frac{1}{q_{H_5}}\Big(\frac{2\varepsilon}{e(2+q_{RK})^2)} - \frac{q_{H_4}}{2^{l_1}} - \frac{q_{H_7}}{2^{l_0+l_1}} - q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big)\Big) \\ t' &\leq t + (q_{H_6}+2q_u+2q_c+2q_{RK}+8q_{RE}+6q_{Dec}+4q_{ReDec})t_e, \end{aligned}$$

where e is the base of natural logarithm, and t_e is the time taken for exponentiation operation in group $\mathbb{G}$.

Proof. If there exists a t-time adversary $\mathscr{A}$ who can break the (t,ε)*IND-CPRE-CCA* security of our scheme, we show how to construct a polynomial time algorithm $\mathscr{C}$ which can break the *CDH* assumption in $\mathbb{G}$ or existential unforgeability against chosen message attack (EUF-*CMA*) of Schnorr Signature with non-negligible advantage. $\mathscr{C}$ is given an instance of CDH challenge tuple (g, g^a, g^b), with $a,b \in_R \mathbb{Z}_q^*$ as input, whose goal is to compute g^{ab}. $\mathscr{C}$ acts as a challenger and plays IND-CPRE-CCA game with $\mathscr{A}$ as described next. $\mathscr{C}$ maintains lists L_K and L_{RK} to store information about all public keys and re-encryption keys as described in the second level security game.

– **Phase 1 :** $\mathscr{C}$ answers queries as follows:

- Uncorrupt Key Generation Query $\mathscr{O}_u(i)$: Uncorrupted user keys are generated using Coron's technique [10] by flipping a coin that takes value $c_i \in \{0,1\}$, where $c_i = 1$ is obtained with probability ρ, which is to be determine later, in the exact manner as shown in second level ciphertext security game.
- Corrupt Key Generation Query $\mathscr{O}_c(i)$: The corrupted user keys are generated as shown in the second level ciphertext security game.
- The challenger $\mathscr{C}$ responds to the oracle queries, re-encryption key generation, re-encryption, second level and first level decryption queries are simulated as shown for the second level ciphertext security game.

– **Challenge:** $\mathscr{A}$ outputs two messages $m_0, m_1 \in_R \{0,1\}^{l_0}$, delegator and delegatee's (target) public keys $pk_{i'}$, pk_i^* and condition ω. $\mathscr{C}$ recovers tuples $\langle pk_{i'}, x_{i',1}, x_{i',2}, c_i' \rangle$, $\langle pk_i^*, x_{i,1}^*, x_{i,2}^*, c_i^* \rangle$ from list L_K. If $c_i' \in \{1, -\}$ and $c_i^* = 0$, $\mathscr{C}$ picks $\delta \in_R \{0,1\}$ and computes challenge ciphertext σ_i^* as below:

1. By definition $z^* = z_1 \cdot z_2 = \frac{1}{a(x_{i',1} H_1(pk_{i',2}, \omega) + x_{i',2})}$, where a is not known to $\mathscr{C}$.
2. Pick $z_1 \in_R \mathbb{Z}_q^*$. From definition, $z_2 = \frac{z_1^{-1}}{a(x_{i',1} H_1(pk_{i',2}, \omega) + x_{i',2})}$.
3. Pick $r' \in \{0,1\}^{l_1}$. Define $H_4(m_\delta, r'^*) = ab$.
4. Compute $E'^* = g^{b\left(\frac{1}{x_{i',1} H_1(pk_{i',1}, \omega) + x_{i',2}}\right)}$.
5. Pick $F^* \in_R \{0,1\}^{l_0 + l_1}$. Implicitly define $F^* = H_5(g^{ab}) \oplus (m_\delta || r')$.
6. Pick $h \in_R \mathbb{Z}_q^*$. Compute $v = H_2(z_1, h)$, $V^* = g^v$, $W^* = H_3(g^v) \oplus (z_1 || h)$.
7. Output the challenge ciphertext $D_i^* = (E'^*, F^*, V^*, W^*)$ to $\mathscr{A}$.

Note that the challenge ciphertext C_i^* is identically distributed as the actual ciphertext generated from the re-encryption algorithm, as shown below:

$$\begin{aligned} E'^* &= g^{b\left(\frac{1}{x_{i',1} H_1(pk_{i',1}, \omega) + x_{i',2}}\right)} \\ &= g^{ab \cdot \left(\frac{1}{a(x_{i',1} H_1(pk_{i',1}, \omega) + x_{i',2})}\right)} \\ &= g^{rz_2}. \\ F^* &= H_5(g^{ab}) \oplus (m_\delta || r') \\ &= H_5(g^r) \oplus (m_\delta || r'). \end{aligned}$$

– **Phase 2 :** $\mathscr{A}$ continues to issue queries to $\mathscr{C}$ as in Phase 1, with the restrictions described in the IND-CPRE-CCA game, and $\mathscr{C}$ responds in the same manner as in Phase 1.

– **Guess :** Eventually, $\mathscr{A}$ outputs its guess $\delta' \in_R \{0,1\}$ to $\mathscr{C}$. The challenger $\mathscr{C}$ randomly picks a tuple $\langle R', \psi \rangle$ from L_{H_5} list and outputs R' as the solution to the CDH problem instance.

– **Probability Analysis:** We proceed to the analysis of the simulation. The simulation of random oracles are perfect unless the following events occur:

- E_{4^*}: (m_δ, r'^*) is queried to H_4.
- E_{5^*}: (g^{ab}) is queried to H_5.

Next, we evaluate the simulation of re-encryption and decryption oracles. We denote the probability that the challenger $\mathscr{C}$ aborts the game during re-encryption key generation or Challenge phase using *Abort*. Note that in both phases, C does not abort in case of the following events:

- E_1: $c_i = 1$ in re-encryption key generation query.
- E_2: $c_i^* = 0 \wedge c_i' \neq 0$ in the Challenge phase.

We get $Pr[\neg Abort] \geq \rho^{q_{RK}}(1-\rho)^2$, which is maximum at $\rho_{OPT} = \frac{q_{RK}}{2+q_{RK}}$. Using ρ_{OPT}, we obtain the probability $Pr[\neg Abort]$ is atleast $\frac{2}{e(2+q_{RK})^2}$.

The analysis of the simulation of the decryption oracle is the same as shown in second level ciphertext security game. Since the adversary can pose atmost q_{Dec} decryption queries, we obtain:

$$Pr[E_{derr}] \leq q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big).$$

We use E_{err} to represent the event $(E_{4^*} \vee E_{5^*} \vee E_{derr})|\neg Abort$.

Following a similar analysis shown for second level ciphertext security, we obtain the following bound on $Pr[E_{5^*}]$:

$$\begin{aligned} Pr[E_{5^*}] &\geq Pr[\neg Abort] \cdot \varepsilon - Pr[E_{4^*}] - Pr[E_{derr}] \\ &\geq \frac{2\varepsilon}{e(2+q_{RK})^2)} - \frac{q_{H_4}}{2^{l_1}} - \frac{q_{H_7}}{2^{l_0+l_1}} - q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big) \end{aligned}$$

If the event E_{5^*} takes place, then $\mathscr{C}$ can solve the *CDH* instance with the advantage:

$$\varepsilon' \geq \frac{1}{q_{H_5}} Pr[E_{5^*}] \geq \frac{1}{q_{H_5}}\left(\frac{2\varepsilon}{e(2+q_{RK})^2)} - \frac{q_{H_4}}{2^{l_1}} - q_{Dec}\Big(\frac{q_{H_4}+q_{H_5}}{2^{l_0+l_1}} + \frac{2}{q}\Big)\right)$$

Also, if the event E_{5^*} takes place, $\mathscr{C}$ solves the *CDH* hard instance with a running time:

$$t' \leq t + (q_{H_6} + 2q_u + 2q_c + 2q_{RK} + 8q_{RE} + 6q_{Dec} + 4q_{ReDec})t_e.$$

This completes the proof of the theorem.

□

4 Comparison

This section provides a comparison of our CPRE scheme with the well-known pairing based CPRE scheme in the literature. We use the following notations: t_e and t_{bp} to denote the time required for exponentiation and bilinear pairing respectively. Note that pairing is as one of the most expensive operations in terms of computational complexity and memory requirement. In fact, pairing operations take more than twice the time taken by modular exponentiations. Till date, the scheme due to Vivek *et al*. [30] is the most efficient CPRE scheme based on bilinear pairing. Table 1 shows the computational efficiency of our scheme along with the a few existing pairing based CPRE schemes. The comparisons indicate that our proposed design is much more efficient than the existing pairing based CPRE schemes and incurs minimal efficiency loss.

Scheme	KeyGen	ReKeyGen	Encrypt	Decrypt	Re-Encrypt	Re-Decrypt
[27]	t_e	$4t_e$	$3t_e + 2t_{bp}$	$3t_e + 2t_{bp}$	$t_e + 2t_{bp}$	$3t_e + 2t_{bp}$
[23]	t_e	$3t_e$	$4t_e + t_{bp}$	$2t_e + 3t_{bp}$	$t_e + 5t_{bp}$	$2t_e + t_{bp}$
[19]	$4t_e$	$9t_e + t_{bp}$	$5t_e + t_{bp}$	$2t_e + 6t_{bp}$	$6t_e + 7t_{bp}$	$5t_e + 12t_{bp}$
[30]	$2t_e$	$3t_e$	$7t_e + t_{bp}$	$6t_e + t_{bp}$	$3t_e + t_{bp}$	$4t_e$
Our Scheme	$2t_e$	$2t_e$	$6t_e$	$6t_e$	$5t_e$	$3t_e$

Table 1: Efficiency comparison of pairing-based unidirectional CPRE schemes with our scheme. The comparison shows that our scheme is more efficient than all the existing CPRE schemes.

5 Conclusion

In this paper, we have proposed an efficient unidirectional CPRE scheme without pairing, which is CCA-secure in the random oracle model. To the best of our knowledge, ours is the only scheme that does not make use of pairing, and is hence more practical in real-world scenarios. Our model is designed to benefit blockchain based distributed cloud-storage and tag-based email application. A notable feature unique to our scheme is modularity, that converts the CPRE scheme to a traditional PRE scheme by a minor modification to a hash function description as discussed. In the proposed scheme as well as in all CPRE schemes proposed in the literature, the sender (encryptor) should be aware of the condition. Typically, the conditions will be known only to the delegator and hence we need an encryption scheme that is oblivious to the conditions that are imposed by the delegator on the proxy. Thus, devising a proxy re-encryption scheme that is independent of the condition will be an interesting and useful direction, and we leave it as an open problem. Note that we do not address collusion-resistance in our work. Designing collusion-resistant pairing-free CPRE is yet another interesting open problem to investigate into.

References

[1] KMSchain - decentralized key management service for dapps. `https://kmschain.com/` [Online; accessed on May 20, 2021], 2018.

[2] 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.

[3] 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, 2006.

[4] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Proc. of the 22nd Annual International Cryptology Conference (CRYPTO'02), Santa Barbara, California, USA*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer-Verlag Berlin Heidelberg, August 2002.

[5] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In *Proc. of the 1998 International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'98), Espoo, Finland*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer-Verlag Berlin Heidelberg, May 1998.

[6] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *Proc. of the 2004 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, Berlin, Heidelberg, May 2004.

[7] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proc. of the 2007 ACM Conference on Computer and Communications Security (CCS'07), Alexandria, Virginia, USA*, pages 185–194. ACM, October 2007.

[8] S. S. Chow, J. Weng, Y. Yang, and R. H. Deng. Efficient unidirectional proxy re-encryption. In *Proc. of the 3rd International Conference on Cryptology in Africa (AFRICACRYPT'10), Stellenbosch, South Africa*, volume 6055 of *Lecture Notes in Computer Science*, pages 316–332. Springer, Berlin, Heidelberg, May 2010.

[9] C. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng. Conditional proxy broadcast re-encryption. In *Proc. of the 14th Australasian Conference on Information Security and Privacy (ACISP'09), Brisbane, Australia*, volume 5594 of *Lecture Notes in Computer Science*, pages 327–342. Springer, Berlin, Heidelberg, July 2009.

[10] 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, Berlin, Heidelberg, 2000.

[11] L. Fang, W. Susilo, C. Ge, and J. Wang. Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search. *Theoretical Computer Science*, 462:39–58, November 2012.

[12] L. Fang, W. Susilo, C. Ge, and J. Wang. Hierarchical conditional proxy re-encryption. *Computer Standards & Interfaces*, 34(4):380–389, June 2012.

[13] L. Fang, W. Susilo, and J. Wang. Anonymous conditional proxy re-encryption without random oracle. In *Proc. of the 3rd International Conference on Provable Security (ProvSec'09), Guangzhou, China*, volume 5848 of *Lecture Notes in Computer Science*, pages 47–60. Springer, Berlin, Heidelberg, November 2009.

[14] K. He, J. Weng, R. H. Deng, and J. K. Liu. On the security of two identity-based conditional proxy re-encryption schemes. *Theoretical Computer Science*, 652:18–27, November 2016.

[15] Q. Huang, Y. Yang, and J. Fu. PRECISE: identity-based private data sharing with conditional proxy re-encryption in online social networks. *Future Generation Computer Systems*, 86:1523–1533, September 2018.

[16] J. Li, X. Zhao, and Y. Zhang. Certificate-based conditional proxy re-encryption. In *Proc. of the 8th International Conference on Network and System Security (NSS'14), Xi'an, China*, volume 8792 of *Lecture Notes in Computer Science*, pages 299–310. Springer, Cham, October 2014.

[17] J. Li, X. Zhao, Y. Zhang, and W. Yao. Provably secure certificate-based conditional proxy re-encryption. *Journal of Information Science and Engineering*, 32(4):813–830, July 2016.

[18] K. Liang, Z. Liu, X. Tan, D. S. Wong, and C. Tang. A cca-secure identity-based conditional proxy re-encryption without random oracles. In *Proc. of the 15th International Conference on Information Security and Cryptology (ICISC'12), Seoul, Korea*, volume 7839 of *Lecture Notes in Computer Science*, pages 231–246, November 2012.

[19] K. Liang, W. Susilo, J. K. Liu, and D. S. Wong. Efficient and fully CCA secure conditional proxy re-encryption from hierarchical identity-based encryption. *The Computer Journal*, 58(10):2778–2792, October 2015.

[20] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute based proxy re-encryption with delegating capabilities. In *Proc. of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS'09), Sydney, Australia*, pages 276–286. ACM, March 2009.

[21] B. Libert and D. Vergnaud. Tracing malicious proxies in proxy re-encryption. In *Proc. of the 2nd International Conference on Pairing-Based Cryptography, Egham, UK*, volume 5209 of *Lecture Notes in Computer Science*, pages 332–353. Springer, Berlin, Heidelberg, September 2008.

[22] Y. Liu, Y. Ren, C. Ge, J. Xia, and Q. Wang. A cca-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system. *Journal of Information Security and Applications*, 47:125–131, August 2019.

[23] J. Qiu, G. Hwang, and H. Lee. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *Proc. of the 9th Asia Joint Conference on Information Security (AsiaJCIS'14), Wuhan, China*, pages 104–110. IEEE, September 2014.

[24] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[25] J. Shao, G. Wei, Y. Ling, and M. Xie. Identity-based conditional proxy re-encryption. In *Proc. of the 2011*

IEEE International Conference on Communications (ICC'11), Kyoto, Japan, pages 1–5. IEEE, June 2011.

[26] T. Smith. Dvd jon: buy drm-less tracks from apple itunes. `https://www.theregister.co.uk/2005/03/18/itunes_pymusique/` [Online; accessed on May 20, 2021], March 2005.

[27] J. Son, D. Kim, R. Hussain, and H. Oh. Conditional proxy re-encryption for secure big data group sharing in cloud environment. In *Proc. of the 2014 IEEE Conference on Computer Communications Workshops (INFOCOMW'14), Toronto, Ontario, Canada*, pages 541–546. IEEE, April 2014.

[28] A. Srinivasan and C. P. Rangan. Certificateless proxy re-encryption without pairing: Revisited. In *Proc. of the 3rd International Workshop on Security in Cloud Computing (SCC'15), Singapore*, pages 41–52. ACM, April 2015.

[29] Q. Tang. Type-based proxy re-encryption and its construction. In *Proc. of the 9th International Conference on Cryptology in India (INDOCRYPT'08), Kharagpur, India*, volume 5365 of *Lecture Notes in Computer Science*, pages 130–144. Springer, Berlin, Heidelberg, December 2008.

[30] S. S. Vivek, S. S. D. Selvi, V. Radhakishan, and C. P. Rangan. Efficient conditional proxy reencryption with chosen cipher text security. *International Journal of Network Security & Its Applications*, 4(2):179–199, March 2012.

[31] S. S. Vivek, S. Sharmila Deva Selvi, V. Radhakishan, and C. Pandu Rangan. Conditional proxy re-encryption - a more efficient construction. In *Proc. of the 4th International Conference on Network Security and Applications (CNSA'11), Chennai, India*, pages 502–512. Springer Berlin Heidelberg, July 2011.

[32] J. Weng, R. H. Deng, X. Ding, C. Chu, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney Australia, March 10-12, 2009*, pages 322–332, 2009.

[33] J. Weng, Y. Yang, Q. Tang, R. H. Deng, and F. Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *Proc. of the 12th International Conference on Information Security (ISC'09), Pisa, Italy*, volume 5735 of *Lecture Notes in Computer Science*, pages 151–166, September 2009.

[34] 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.

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.