# An Efficient Certificateless Proxy Re-Encryption Scheme Without Pairing

S. Sharmila Deva Selvi, Arinjita Paul(✉), and Chandrasekaran Pandu Rangan

Theoretical Computer Science Lab, Department of Computer Science and Engineering, Indian Institute of Technology Madras, Chennai, India
{sharmila,arinjita,prangan}@cse.iitm.ac.in

**Abstract.** Proxy re-encryption (PRE) is a cryptographic primitive introduced by Blaze, Bleumer and Strauss [4] to provide delegation of decryption rights. PRE allows re-encryption of a ciphertext intended for Alice (delegator) to a ciphertext for Bob (delegatee) via a semi-honest proxy, who should not learn anything about the underlying message. In 2003, Al-Riyami and Patterson introduced the notion of certificateless public key cryptography which offers the advantage of identity-based cryptography without suffering from key escrow problem. The existing certificateless PRE (CLPRE) schemes rely on costly bilinear pairing operations. In ACM ASIA-CCS SCC 2015, Srinivasan *et al.* proposed the first construction of a certificateless PRE scheme without resorting to pairing in the random oracle model. In this work, we demonstrate a flaw in the CCA-security proof of their scheme. Also, we present the first construction of a CLPRE scheme without pairing which meets CCA security under the computational Diffie-Hellman hardness assumption in the random oracle model.

**Keywords:** Proxy re-encryption · Pairing-less · Public key · Certificateless · Unidirectional

## 1 Introduction

Due to segregation of data ownership and storage, security remains as one of the major concerns in public cloud scenario. In order to protect the stored data from illegal access and usage, users encrypt their data with their public keys before storing it in the cloud. To enable sharing of stored data, a naive approach would be that a user Alice shares her secret key with a legitimate user Bob. However, this would compromise the privacy of Alice. As a solution towards providing delegation of decryption rights, Blaze *et al.* [4] in 1998 proposed the concept

S. Sharmila Deva Selvi—Postdoctoral researcher supported by Project No. CCE/CEP/22/VK&CP/CSE/14-15 on Information Security & Awareness (ISEA) Phase-II by Ministry of Electronics & Information Technology, Government of India.
A. Paul and C. Pandu Rangan—Work partially supported by Project No. CCE/CEP/22/VK&CP/CSE/14-15 on ISEA-Phase II.

of proxy re-encryption, which allows a proxy server with special information (re-encryption key) to translate a ciphertext for Alice into another ciphertext (with the same message) for Bob without learning any information about the underlying plaintext. Besides, this approach offloads the costly burden of secure data sharing from Alice to the resource-abundant proxy. As Alice delegates her decryption rights to Bob, Alice is termed as *delegator* and Bob as *delegatee*. Ever since, PRE has found a lot of applications such as encrypted email forwarding, distributed file systems, digital rights management (DRM) of Apple's iTunes, outsourced filtering of encrypted spam and content distribution [2,3].

Based on the direction of delegation, PRE schemes are classified into unidirectional and bidirectional schemes. In unidirectional schemes, a proxy can re-encrypt ciphertexts from Alice to Bob but not vice-versa, while in the bidirectional schemes, the proxy is allowed to re-encrypt ciphertexts in both directions. PRE schemes are also classified into single-hop and multihop schemes. In a single-hop scheme, a proxy cannot re-encrypt ciphertexts that have been re-encrypted once. In a multi-hop scheme, the proxy can further re-encrypt the re-encrypted ciphertexts. In this paper, we focus on single-hop unidirectional PRE schemes.

Several PRE constructions have been proposed in the literature, either in the Public Key Infrastructure (PKI) or identity based (IBE) setting. The schemes in the PKI setting entrusts a third party called the Certification Authority (CA) to assure the authenticity of a user's public key by digitally signing it and issuing Digital Certificates. However, the overhead involved in the revocation, storage and distribution of certificates has long been a concern, which makes public key cryptography inefficient. As a solution to the authenticity problem, Identity-based cryptography was introduced by Shamir in 1984 [9], which involves a trusted third party called the Private Key Generator (PKG) to generate secret keys of all users. Yet again, due to the unconditional trust placed on the PKG, identity based cryptography suffers from *key-escrow problem*. To avoid both certificate management problem in the PKI setting and key-escrow problem in the ID-based setting, certificateless cryptography was introduced in 2003 by Al-Riyami and Patterson [1]. Certificateless cryptography splits the task of key-generation of a user between a semi-trusted entity called Key Generation Center (KGC) and the user himself. This approach no longer relies on the use of certificates for key authenticity and hence does not suffer from certificate management problem. Also, the KGC does not have access to the secret keys of the users, which addresses the key-escrow problem inherent in IBE setting.

In this paper, we study proxy re-encryption in the light of certificateless public key cryptography. Consider the following scenario where Alice stores her encrypted data in the cloud, which provides services to billions of users. The number of cloud users being large, certificate management for public key authenticity is an overhead. This makes proxy re-encryption in PKI setting unfit for cloud services. On the other hand, a malicious PKG, entrusted with the power to generate user secret keys, can decrypt confidential data of the users due to which PRE in

IBE setting is highly impractical. Certificateless PRE affirmatively solves both the certificate management problem and key-escrow problem in the above scenario.

### 1.1 Related Work and Contribution

While several schemes achieving PRE have been proposed in the literature, a majority of these schemes are either in the PKI or IBE setting. In 2010, Sur *et al.* [11] introduced the notion of certificateless proxy re-encryption (CLPRE) and proposed a CCA secure CLPRE scheme in the random oracle model. However, in 2013, their scheme was shown to be vulnerable to chosen ciphertext attack by Zheng *et al.* [13]. In 2013, Guo *et al.* [6] proposed a CLPRE scheme in the random oracle model based on bilinear pairing which satisfies RCCA-security, a weaker notion of security. In 2014, Yang, Xu and Zhang [12] proposed a pairing-free CCA-secure CLPRE scheme in the random oracle model, which was shown to be vulnerable to chain collusion attack in [10]. In 2015, Srinivasan *et al.* [10] proposed the first CCA-secure unidirectional certificateless PRE scheme without pairing under the computational Diffie-Hellman assumption in the random oracle model. In this paper, we expose a critical weakness in the security proof of the scheme and provide a potential fix to make the scheme provably secure.

Another major contribution of this work is we propose an efficient pairing-free unidirectional single-hop certificateless proxy re-encryption scheme in the random oracle model. As stated, all the existing CLPRE schemes are vulnerable to attacks except for [6]. The CLPRE scheme due to Guo *et al.* [6] is based on bilinear pairing which is an expensive operation as compared to modular exponentiation operations in finite fields. Besides, their scheme [6] satisfies a weaker notion of security, namely RCCA-security and is based on *q-weak Decisional Bilinear Assumption*. Our scheme satisfies CCA security against both Type-I and Type-II adversaries and is based on a much standard assumption called the Computational Diffie Hellman ($CDH$) assumption.

## 2 Definition and Security Model

### 2.1 Definition

We describe the syntactical definition of unidirectional single-hop certificateless proxy re-encryption and its security notion adopted from [10]. A PRE scheme consists of the following algorithms:

- **Setup**$(1^\lambda)$: A PPT algorithm run by the Key Generation Center (KGC), which takes the unary encoding of the security parameter $\lambda$ as input and outputs the public parameters $params$ and master secret key $msk$.
- **PartialKeyExtract**$(msk, ID_i, params)$: A PPT algorithm run by KGC which takes the master secret key $msk$, user identity $ID_i$ and public parameters $params$ as input, and outputs the partial public key and partial secret key pair $(PPK_i, PSK_i)$.

- **UserKeyGen**$(ID_i, params)$: A PPT algorithm run by the user, which takes the identity $ID_i$ of the user and the public parameters *params* as input, and outputs the user generated secret key and public key pair $(USK_i, UPK_i)$.
- **SetPrivateKey**$(ID_i, PSK_i, USK_i, params)$: A PPT algorithm run by the user, which takes as input the identity $ID_i$ of the user, partial secret key $PSK_i$, user generated secret key $USK_i$ and public parameters *params*, and outputs the full secret key $SK_i$ of the user.
- **SetPublicKey**$(ID_i, PPK_i, PSK_i, UPK_i, USK_i, params)$: A PPT algorithm run by the user, which takes as input the the identity $ID_i$ of the user, partial public key $PPK_i$, partial secret key $PSK_i$, user generated public key $UPK_i$, user generated secret key $USK_i$ and public parameters *params*, and outputs the full public key $PK_i$ of the user.
- **Re-KeyGen**$(ID_i, ID_j, SK_i, PK_j, params)$: A PPT algorithm run by the user (delegator) with identity $ID_i$ which takes as input the identity $ID_i$ of the delegator, identity $ID_j$ of the delegatee, the full secret key $SK_i$ of $ID_i$, full public key $PK_j$ of $ID_j$ and public parameters *params*, and outputs a re-encryption key $RK_{i\to j}$ or an error symbol $\perp$.
- **Encrypt**$(ID_i, PK_i, m, params)$: A PPT algorithm run by the sender which takes as input identity $ID_i$ of receiver, full public key $PK_i$ of $ID_i$, message $m \in \mathcal{M}$ and the public parameters *params*, and outputs the ciphertext $C$ or an error symbol $\perp$. Note that $C$ is termed as the first level ciphertext.
- **Re-Encrypt**$(ID_i, ID_j, C, RK_{i\to j}, params)$: A PPT algorithm run by the proxy which takes the identities $ID_i, ID_j$, a first level ciphertext $C$ encrypted under identity $ID_i$, a re-encryption key $RK_{i\to j}$ and public parameters *params* as input, and outputs a ciphertext $D$ or an error symbol $\perp$. Note that $D$ is termed as the second-level ciphertext.
- **Decrypt**$(ID_i, SK_i, C, params)$: A deterministic algorithm run by the receiver (delegator) which takes the identity $ID_i$, secret key $SK_i$ of identity $ID_i$, first-level ciphertext $C$ and public parameters *params* as input, and outputs the message $m \in \mathcal{M}$ or an error symbol $\perp$.
- **Re-Decrypt** $(ID_j, SK_j, D, params)$: A deterministic algorithm run by the receiver (delegatee) which takes the identity $ID_j$, secret key $SK_j$ of identity $ID_j$, a second-level ciphertext $D$ and public parameters *params* as input, and outputs the message $m \in \mathcal{M}$ or an error symbol.

The consistency of a CLPRE scheme for any given public parameters *params* and full public-private key pairs $\{(PK_i, SK_i), (PK_j, SK_j)\}$ is defined as follows:

1. Consistency between encryption and decryption; i.e.,

$$\textbf{Decrypt}(ID_i, SK_i, C, params) = m,\ \forall m \in \mathcal{M},$$

   where $C = \textbf{Encrypt}(ID_i, PK_i, m, params)$.

2. Consistency between encryption, proxy re-encryption and decryption; i.e.,

$$\textbf{Re-Decrypt}(ID_j, SK_j, D, params) = m,\ \forall m \in \mathcal{M},$$

   where $D = \textbf{Re-Encrypt}(ID_i, ID_j, C, RK_{i\to j}, params)$ and $C = \textbf{Encrypt}$ $(ID_i, PK_i, m, params)$.

## 2.2 Security Model

Due to the existence of two types of ciphertexts in a PRE scheme namely *first level* and *second level* ciphertexts, it is essential to prove the security for both levels. Again, there exists two types of adversaries specific to CLPRE: *Type-I* adversary and *Type-II* adversary. The Type I adversary models an attacker who can replace the public keys of the users by fake keys of its choice because of the absence of authenticating information for public keys [1]. However, the security proof demonstrates that the adversary cannot learn anything useful from this attack as it cannot derive the partial keys and in turn the full private keys needed for decryption without the cooperation of the KGC (who possesses the master secret key). The Type-II adversary models the semi-trusted KGC, who possesses the master secret key and tries to break the security of the system by eavesdropping or making decryption queries. Note that, the KGC is restrained from replacing the public keys of the users.

The security of a CLPRE scheme is modelled in the form of a security game between the two entities: the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. $\mathcal{A}$ can adaptively query the oracles as listed below which $\mathcal{C}$ answers and simulates an environment running CLPRE for $\mathcal{A}$. $\mathcal{C}$ maintains a list $P_{current}$ of the public keys to keep a track of the replaced public keys. $P_{current}$ consists of tuples of the form $\langle ID_i, PK_i, \hat{PK_i} \rangle$, where $\hat{PK_i}$ denotes the current value of the public key. To begin with, $\hat{PK_i}$ is assigned the value of the initial public key $\hat{PK_i} = PK_i$. $\mathcal{A}$ can make queries to the following oracles which are answered by $\mathcal{C}$:

- Public Key Extract($\mathcal{O}_{pe}(ID_i)$): Given an $ID_i$ as input, compute the partial public key and secret key pair: $(PPK_i, PSK_i) = PartialKeyExtract(msk, ID_i, params)$, the user public key and secret key pair: $(USK_i, UPK_i) = UserKeyGen(ID_i, params)$, the full public key $PK_i = SetPublicKey(ID_i, PPK_i, PSK_i, UPK_i, USK_i, params)$. Return $PK_i$.
- Partial Key Extract($\mathcal{O}_{ppe}(ID_i)$): Given an $ID_i$ as input, compute $(PPK_i, PSK_i) = PartialKeyExtract(msk, ID_i, params)$ and return $(PPK_i, PSK_i)$.
- User Key Extract($\mathcal{O}_{ue}(ID_i)$): Given an $ID_i$ as input, compute $(UPK_i, USK_i) = UserKeyGen(ID_i, params)$ and return $(USK_i, UPK_i)$.
- Re-Key Generation($\mathcal{O}_{rk}(ID_i, ID_j)$): Compute $RK_{i\to j} = \textit{Re-KeyGen}(ID_i, ID_j, SK_i, PK_j, params)$ and return $RK_{i\to j}$.
- Re-Encryption($\mathcal{O}_{re}(ID_i, ID_j, C)$): Given a first-level ciphertext $C$ and two identities $ID_i, ID_j$ as inputs, compute $RK_{i\to j} = \textit{Re-KeyGen}(ID_i, ID_j, SK_i, PK_j, params)$ and compute the second level ciphertext as $D = \textit{Re-Encrypt}(ID_i, ID_j, C, RK_{i\to j}, params)$.
- Decryption($\mathcal{O}_{dec}(ID_i, C)$): Given a first level ciphertext $C$ encrypted under the public key of $ID_i$ as input, compute the decryption of the ciphertext to obtain $m \in \mathcal{M}$. Return $m$ or return $\perp$ if the ciphertext is invalid.
- Re-Decryption($\mathcal{O}_{redec}(ID_i, C)$): Given a second level ciphertext $D$ re-encrypted under the public key $ID_j$ as input, compute the decryption of the ciphertext to obtain $m \in \mathcal{M}$. Return $m$ or return $\perp$ if the ciphertext is invalid.

– Public Key Replacement($\mathcal{O}_{rep}(ID_i, PK_i)$): Replace the value of the third component $\hat{PK_i}$ in the $PK_{current}$ list with the new value $PK_i$, provided $PK_i$ is a valid public key.

## Security Against Type-I Adversary $\mathcal{A}_I$

The Type-I adversary models an outside attacker without access to the master secret key, trying to learn some information about the underlying plaintext, given the ciphertext. We consider separate security models for the first level and second level ciphertexts against $\mathcal{A}_I$.

**First Level Ciphertext Security:** We consider the following security game where $\mathcal{A}_I$ interacts with the challenger $\mathcal{C}$ in following stages.

- Initialization: $\mathcal{C}$ runs $Setup(\lambda)$ to generate the public parameters $params$ and master secret key $msk$. It sends $params$ to $\mathcal{A}_I$ while keeping $msk$ secret.
- Phase 1: The challenger $\mathcal{C}$ sets up the list of corrupt and honest users, initialises $\hat{PK_i}$ to $PK_i$ for all users in the public key list $P_{current}$. $\mathcal{A}_I$ issues several queries to the above oracles simulated by $\mathcal{C}$, with the restriction that $\mathcal{A}_I$ cannot make partial key extract queries ($\mathcal{O}_{ppe}$) or user key extract queries ($\mathcal{O}_{ue}$) of the users whose public keys have been replaced as it is unreasonable to expect $\mathcal{C}$ to respond to such queries for public keys replaced by $\mathcal{A}_I$ [1].
- Challenge: $\mathcal{A}$ outputs two equal length messages $m_0$ and $m_1$ in $\mathcal{M}$ and the target identity $ID_{ch}$, with the following adversarial constraints:
  – $ID_{ch}$ should not be a corrupt user.
  – $\mathcal{A}_I$ must not query the partial key extract oracle ($\mathcal{O}_{ppe}$) or user key extract oracle ($\mathcal{O}_{ue}$) of $ID_{ch}$ at any point in time.
  – $\mathcal{A}_I$ must not query $\mathcal{O}_{rk}(ID_{ch}, ID_i)$, where $ID_i$ is a corrupt user.
  – If $\mathcal{A}_I$ replaces the public key of $ID_{ch}$, it should not query the partial key extract oracle ($\mathcal{O}_{ppe}$) for $ID_{ch}$.

  On receiving $\{m_0, m_1\}$, $\mathcal{C}$ picks $\delta \in \{0, 1\}$ at random and generates a challenge ciphertext $C^* = Encrypt(ID_{ch}, \hat{PK}_{ch}, m_\delta, params)$ and gives to $\mathcal{A}_I$.
- Phase 2: $\mathcal{A}_I$ issues the queries to the oracles similar to Phase 1, with the same adversarial constraint as mentioned in Phase 1 and the added constraints on the target identity $ID_{ch}$ as mentioned in the Challenge phase. Additionally, there are other constraints as below:
  – $\mathcal{A}_I$ cannot query $\mathcal{O}_{dec}(ID_{ch}, C^*)$, for the same public key of $ID_{ch}$ that was used to initially encrypt $m_\delta$.
  – $\mathcal{A}_I$ cannot query the re-decryption oracle $\mathcal{O}_{redec}(ID_i, C)$ if $(ID_i, C)$ is a challenge derivative[1].

[1] The definition of challenge derivative $(ID_i, C)$ is adopted from [5] as stated below:
• Reflexitivity: $(ID_i, C)$ is a challenge derivative of itself.
• Derivative by re-encryption: $(ID_j, C')$ is a challenge derivative of $(ID_i, C)$ if $C' \leftarrow \mathcal{O}_{re}(ID_i, ID_j, C)$.
• Derivative by re-encryption key: $(ID_j, C')$ is a challenge derivative of $(ID_i, C)$ if $RK_{i\rightarrow j} \leftarrow \mathcal{O}_{rk}(ID_i, ID_j)$ and $C' = Re - Encrypt(ID_i, ID_j, C, RK_{i\rightarrow j}, params)$.
.

  - $\mathcal{A}_I$ cannot query $\mathcal{O}_{re}(ID_i, ID_j, C)$, if $(ID_i, C)$ is a challenge derivative and $ID_j$ is a corrupt user.
  - $\mathcal{A}_I$ cannot query $\mathcal{O}_{rk}(ID_{ch}, ID_j)$, if $ID_j$ is a corrupt user.
- Guess: $\mathcal{A}_I$ outputs its guess $\delta' \in \{0, 1\}$.

We define the advantage of $\mathcal{A}_I$ in winning the game as:

$$Adv_{A_I,first}^{IND-CLPRE-CCA} = 2|Pr[\delta' = \delta] - \frac{1}{2}|$$

where the probability is over the random coin tosses performed by $\mathcal{C}$ and $\mathcal{A}_I$. The scheme is said to be $(t, \epsilon)IND-CLPRE-CCA$ secure for the first level ciphertext against Type-I adversary $\mathcal{A}_I$ if for all $t$-time adversary $\mathcal{A}_I$ that makes $q_{pe}$ queries to $\mathcal{O}_{pe}$, $q_{ppe}$ queries to $\mathcal{O}_{ppe}$, $q_{ue}$ queries to $\mathcal{O}_{ue}$, $q_{re}$ queries to $\mathcal{O}_{re}$, $q_{rk}$ queries to $\mathcal{O}_{rk}$, $q_{dec}$ queries to $\mathcal{O}_{dec}$, $q_{redec}$ queries to $\mathcal{O}_{redec}$ and $q_{rep}$ queries to $\mathcal{O}_{rep}$, the advantage of $\mathcal{A}_I$ is $Adv_{A_I,first}^{IND-CLPRE-CCA} \leq \epsilon$.

**Second Level Ciphertext Security:** We consider the following security game for security of the second level ciphertext against Type-I adversary $\mathcal{A}_I$, where $\mathcal{A}_I$ interacts with the challenger $\mathcal{C}$ in following stages.

- Initialization: $\mathcal{C}$ runs $Setup(\lambda)$ to generate the public parameters $params$ and master secret key $msk$. It sends $params$ to $\mathcal{A}_I$ while keeping $msk$ secret.
- Phase 1: The challenger $\mathcal{C}$ sets up the list of corrupt and honest users, initialises $\hat{PK}_i$ to $PK_i$ for all users and updates the public key list $P_{current}$. $\mathcal{A}_I$ issues several queries to the above oracles simulated by $\mathcal{C}$ with the restriction that it cannot make partial key extract queries ($\mathcal{O}_{ppe}$) or user key extract queries ($\mathcal{O}_{ue}$) of the users whose public keys have already been replaced.
- Challenge: $\mathcal{A}_I$ outputs two messages $m_0$, $m_1$ in $\mathcal{M}$ where $|m_0| = |m_1|$, the target identity $ID_{ch}$, and the delegator's identity $ID_{del}$ with the adversarial constraints as follows:
  - $ID_{ch}$ should not be a corrupt user.
  - $\mathcal{A}_I$ must not query the partial key extract oracle ($\mathcal{O}_{ppe}$) or user key extract oracle ($\mathcal{O}_{ue}$) of $ID_{ch}$ at any point in time.
  - If $\mathcal{A}_I$ replaces the public key of $ID_{ch}$, it should not query the partial key extract oracle ($\mathcal{O}_{ppe}$) for $ID_{ch}$.
  - $\mathcal{A}_I$ must not query $\mathcal{O}_{rk}(ID_{del}, ID_{ch})$.
  - $\mathcal{A}_I$ must not query $\mathcal{O}_{rk}(ID_{ch}, ID_i)$, where $ID_i$ is a corrupt user.

  On receiving $\{m_0, m_1\}$, $\mathcal{C}$ picks $\delta \in \{0, 1\}$ at random and generates a challenge ciphertext $D^* = Re-Encrypt(ID_{del}, ID_{ch}, Encrypt(ID_{ch}, \hat{PK}_{ch}, m_\delta, params), RK_{ID_{del} \to ID_{ch}}, params)$ and gives to $\mathcal{A}_I$.
- Phase 2: $\mathcal{A}_I$ issues the queries to the oracles similar to Phase 1, with the same adversarial constraint as mentioned in Phase 1 and constraints on the target identity $ID_{ch}$ mentioned in the Challenge phase. Additionally, $\mathcal{A}_I$ cannot query $\mathcal{O}_{redec}(ID_{ch}, C^*)$, for the same public key of $ID_{ch}$ that was used to initially encrypt $m_\delta$.
- Guess: $\mathcal{A}_I$ outputs its guess $\delta' \in \{0, 1\}$.

We define the advantage of $\mathcal{A}_I$ in winning the game as:

$$Adv_{A_I,second}^{IND-CLPRE-CCA} = 2|Pr[\delta' = \delta] - \frac{1}{2}|$$

where the probability is over the random coin tosses performed by $\mathcal{C}$ and $\mathcal{A}_I$. The scheme is said to be $(t, \epsilon)IND-CLPRE-CCA$ secure for the second level ciphertext against Type-I adversary $\mathcal{A}_I$ if for all $t$-time adversary $\mathcal{A}_I$ that makes $q_{pe}$ queries to $\mathcal{O}_{pe}$, $q_{ppe}$ queries to $\mathcal{O}_{ppe}$, $q_{ue}$ queries to $\mathcal{O}_{ue}$, $q_{re}$ queries to $\mathcal{O}_{re}$, $q_{rk}$ queries to $\mathcal{O}_{rk}$, $q_{dec}$ queries to $\mathcal{O}_{dec}$, $q_{redec}$ queries to $\mathcal{O}_{redec}$ and $q_{rep}$ queries to $\mathcal{O}_{rep}$, the advantage of $\mathcal{A}_I$ is $Adv_{A_I,second}^{IND-CLPRE-CCA} \leq \epsilon$.

## Security Against Type-II Adversary $\mathcal{A}_{II}$

The Type-II adversary models an *honest-but-curious KGC* who has access to the master secret key $msk$, but is not allowed to replace the public keys of users. We consider separate security models for the first and second level ciphertexts.

**First Level Ciphertext Security:** We consider the following security game where $\mathcal{A}_{II}$ interacts with the challenger $\mathcal{C}$ as follows.

- Initialization: $\mathcal{C}$ runs $Setup(\lambda)$ to generate the public parameters $params$ and master secret key $msk$. It sends both $params$ and $msk$ to $\mathcal{A}_{II}$.
- Phase 1: The challenger $\mathcal{C}$ maintains the list of honest and corrupt users and initialises $\hat{PK}_i$ to $PK_i$ for all the users in the public key list $P_{current}$. $\mathcal{A}_{II}$ issues several queries to the above stated oracles simulated by $\mathcal{C}$ with the restriction that it cannot make partial key extract queries ($\mathcal{O}_{ppe}$) or user key extract queries ($\mathcal{O}_{ue}$) of the users whose public keys have been replaced.
- Challenge: $\mathcal{A}_{II}$ outputs two equal length messages $\{m_0, m_1\}$ in $\mathcal{M}$ and the target identity $ID_{ch}$, with the adversarial constraints as follows:
  - $ID_{ch}$ should not be a corrupt user.
  - $\mathcal{A}_{II}$ must not replace the public key of $ID_{ch}$.
  - $\mathcal{A}_{II}$ must have not queried $\mathcal{O}_{rk}(ID_{ch}, ID_i)$, where $ID_i$ is a corrupt user.

  On receiving $\{m_0, m_1\}$, $\mathcal{C}$ selects $\delta \in \{0, 1\}$ at random, generates a challenge ciphertext $C^* = Encrypt(ID_{ch}, \hat{PK}_{ch}, m_\delta, params)$ and gives $C^*$ to $\mathcal{A}_{II}$.
- Phase 2: $\mathcal{A}_{II}$ issues the queries to the oracles similar to Phase 1, with the same adversarial constraints as mentioned in Phase 1 and the constraints on the target identity $ID_{ch}$ as mentioned in the Challenge phase. Additionally, there are other constraints as below:
  - $\mathcal{A}_{II}$ cannot query $\mathcal{O}_{dec}(ID_{ch}, C^*)$, for the same public key of $ID_{ch}$ that was used to initially encrypt $m_\delta$.
  - $\mathcal{A}_{II}$ cannot query the re-decryption oracle $\mathcal{O}_{redec}(ID, C)$ if $(ID, C)$ is a challenge derivative.
  - $\mathcal{A}_{II}$ cannot query $\mathcal{O}_{re}(ID_i, ID_j, C)$, if $(ID_i, C)$ is a challenge derivative and $ID_j$ is a corrupt user.
  - $\mathcal{A}_{II}$ cannot query $\mathcal{O}_{rk}(ID_{ch}, ID_j)$, if $ID_j$ is a corrupt user.
- Guess: $\mathcal{A}_{II}$ outputs its guess $\delta' \in \{0, 1\}$.

We define the advantage of $\mathcal{A}_{II}$ in winning the game as:

$$Adv_{\mathcal{A}_{II},first}^{IND-CLPRE-CCA} = 2|Pr[\delta' = \delta] - \frac{1}{2}|$$

where the probability is over the random coin tosses performed by $\mathcal{C}$ and $\mathcal{A}_{II}$. The scheme is said to be $(t,\epsilon)IND-CLPRE-CCA$ secure for the first level ciphertext against Type-II adversary $\mathcal{A}_{II}$ if for all $t$-time adversary $\mathcal{A}_{II}$ that makes $q_{pe}$ queries to $\mathcal{O}_{pe}$, $q_{ppe}$ queries to $\mathcal{O}_{ppe}$, $q_{ue}$ queries to $\mathcal{O}_{ue}$, $q_{re}$ queries to $\mathcal{O}_{re}$, $q_{rk}$ queries to $\mathcal{O}_{rk}$, $q_{dec}$ queries to $\mathcal{O}_{dec}$, $q_{redec}$ queries to $\mathcal{O}_{redec}$ and $q_{rep}$ queries to $\mathcal{O}_{rep}$, the advantage of $\mathcal{A}_{II}$ is $Adv_{\mathcal{A}_{II},first}^{IND-CLPRE-CCA} \leq \epsilon$.

**Second Level Ciphertext Security:** We consider the following security game where $\mathcal{A}_{II}$ interacts with the challenger $\mathcal{C}$ in the following stages.

- Initialization: $\mathcal{C}$ runs $Setup(\lambda)$ to generate the public parameters $params$ and master secret key $msk$. It sends both $params$ and $msk$ to $\mathcal{A}_{II}$.
- Phase 1: The challenger $\mathcal{C}$ sets up the list of corrupt and honest users, initialises $\hat{PK}_i$ to $PK_i$ for all the users and updates the public key list $P_{current}$. $\mathcal{A}_{II}$ issues several queries to the above stated oracles simulated by $\mathcal{C}$ with the restriction that it cannot make partial key extract queries ($\mathcal{O}_{ppe}$) or user key extract queries ($\mathcal{O}_{ue}$) of the users whose public keys have been replaced. Also, $\mathcal{A}_{II}$ cannot place queries to $\mathcal{O}_{ppe}$ as it already has access to $msk$ and can generate the partial keys itself.
- Challenge: $\mathcal{A}_{II}$ outputs two messages $m_0$ and $m_1$ in $\mathcal{M}$ where $|m_0| = |m_1|$, the target identity $ID_{ch}$, and the delegator's identity $ID_{del}$ with the adversarial constraints as follows:
  – $ID_{ch}$ should not be a corrupt user.
  – $\mathcal{A}_{II}$ must not query the user key extract oracle ($\mathcal{O}_{ue}$) of $ID_{ch}$.
  – $\mathcal{A}_{II}$ must not replace the public key of $ID_{ch}$.
  – $\mathcal{A}_{II}$ must not query $\mathcal{O}_{rk}(ID_{del}, ID_{ch})$.
  – $\mathcal{A}_{II}$ must have not queried $\mathcal{O}_{rk}(ID_{ch}, ID_i)$, where $ID_i$ is a corrupt user.

  On receiving $\{m_0, m_1\}$, $\mathcal{C}$ picks $\delta \in \{0,1\}$ at random and generates a challenge ciphertext $D^* = Re-Encrypt(ID_{del}, ID_{ch}, Encrypt(ID_{ch}, \hat{PK}_{ch}, m_\delta, params), RK_{ID_{del}\to ID_{ch}}, params)$ and gives to $\mathcal{A}_{II}$.
- Phase 2: $\mathcal{A}_{II}$ issues the queries to the oracles similar to Phase 1, with the same adversarial constraint as mentioned in Phase 1 and the added constraint on the target identity $ID_{ch}$ as mentioned in the Challenge phase. Additionally, $\mathcal{A}_{II}$ cannot query $\mathcal{O}_{redec}(ID_{ch}, C^*)$, for the same public key of $ID_{ch}$ that was used to initially encrypt $m_\delta$.
- Guess: $\mathcal{A}_{II}$ outputs its guess $\delta' \in \{0,1\}$.

We define the advantage of $\mathcal{A}_{II}$ in winning the game as:

$$Adv_{\mathcal{A}_{II},second}^{IND-CLPRE-CCA} = 2|Pr[\delta' = \delta] - \frac{1}{2}|$$

where the probability is over the random coin tosses performed by $\mathcal{C}$ and $\mathcal{A}_{II}$. The scheme is said to be $(t,\epsilon)IND-CLPRE-CCA$ secure for the second level ciphertext against Type-II adversary $\mathcal{A}_{II}$ if for all $t$-time adversary $\mathcal{A}_{II}$ that makes $q_{pe}$ queries to $\mathcal{O}_{pe}$, $q_{ppe}$ queries to $\mathcal{O}_{ppe}$, $q_{ue}$ queries to $\mathcal{O}_{ue}$, $q_{re}$ queries to $\mathcal{O}_{re}$, $q_{rk}$ queries to $\mathcal{O}_{rk}$, $q_{dec}$ queries to $\mathcal{O}_{dec}$, $q_{redec}$ queries to $\mathcal{O}_{redec}$ and $q_{rep}$ queries to $\mathcal{O}_{rep}$, the advantage of $\mathcal{A}_{II}$ is $Adv_{\mathcal{A}_{II},second}^{IND-CLPRE-CCA} \leq \epsilon$.

**Hardness Assumption**

We state the computational hardness assumption we use to prove the security of our scheme. 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 in* $\mathbb{G}$ *is, given elements* $\{P, aP, bP\} \in \mathbb{G}$, *there exists no PPT adversary which can compute* $abP \in \mathbb{G}$ *with a non-negligible advantage, where* $P$ *is a generator of* $\mathbb{G}$ *and* $a, b \in_R \mathbb{Z}_q^*$.

## 3 Analysis of a Certificateless PRE Scheme by Srinivasan *et al.* [10]

### 3.1 Review of the Scheme

- **Setup**$(1^\lambda)$:
  - Choose two large primes $p$ and $q$ such that $q|p-1$ and the security parameter $\lambda$ defines the bit length of $q$. Let $\mathbb{G}$ be a subgroup of $\mathbb{Z}_p^*$ of order $q$ and $g$ is a generator of $\mathbb{G}$. Pick $x \in_R \mathbb{Z}_q^*$ and compute $y = g^x$.
  - Choose the following cryptographic hash functions:

$$\begin{aligned}
H &: \mathbb{G} \rightarrow \mathbb{Z}_q^*,\\
H_1 &: \{0,\ 1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*,\\
H_2 &: \{0,\ 1\}^* \times \mathbb{G}^3 \rightarrow \mathbb{Z}_q^*,\\
H_3 &: \mathbb{G} \rightarrow \{0,\ 1\}^{l_0+l_1},\\
H_4 &: \{0,\ 1\}^{l_0} \times \{0,\ 1\}^{l_1} \rightarrow \mathbb{Z}_q^*,\\
H_5 &: \mathbb{G}^2 \times \{0,\ 1\}^{l_0+l_1} \rightarrow \mathbb{Z}_q^*,\\
H_6 &: \{0,\ 1\}^* \times \mathbb{G}^2 \rightarrow \mathbb{Z}_q^*
\end{aligned}$$

    Here $l_0 = \log q$ and $l_1$ is determined by the security parameter $\lambda$. The message space $\mathcal{M}$ is set to $\{0,\ 1\}^{l_0}$.
  - Return the public parameters $params = (p, q, \mathbb{G}, g, y, H, H_1, H_2, H_3, H_4, H_5, H_6)$. The master secret key is $msk = x$.
- **PartialKeyExtract**$(msk, ID_i, params)$:
  - Pick $s_1, s_2, s_3 \in_R \mathbb{Z}_q^*$ and compute $Q_1 = g^{s_1},\ Q_2 = g^{s_2},\ Q_3 = g^{s_3}$.
  - Compute $S_1 = s_1 + xH_1(ID_i, Q_1)$, $S_2 = s_2 + xH_1(ID_i, Q_2)$ and $S_3 = s_3 + xH_2(ID_i, Q_1, Q_2, Q_3)$.
  - Return the partial public key $PPK = (Q_1, Q_2, Q_3, S_3)$ and the partial secret key $PSK = (S_1, S_2)$.

- **UserKeyGen**$(ID_i, params)$:
  - Pick $z_1, z_2 \in_R \mathbb{Z}_q^*$ and compute $(g^{z_1}, g^{z_2})$.
  - Return $USK = (U_1, U_2) = (z_1, z_2)$ and $UPK = (P_1, P_2) = (g^{z_1}, g^{z_2})$.
- **SetPublicKey**$(ID_i, PPK, PSK, UPK, USK, params)$:
  - Pick $t_1, t_2 \in_R \mathbb{Z}_q^*$. Compute $T_1 = g^{t_1}$ and $T_2 = g^{t_2}$.
  - Compute $\mu_1 = t_1 + S_1 H_6(ID_i, P_1, T_1)$ and $\mu_2 = t_2 + S_2 H_6(ID_i, P_2, T_2)$.
  - Return the full public key $PK = (P_1, P_2, Q_1, Q_2, Q_3, S_3, T_1, T_2, \mu_1, \mu_2)$.
- **Public Verify**$(ID_i, PK, params)$:
  - Compute $R_1 = Q_1 \cdot y^{H_1(ID_i, Q_1)}$ and $R_2 = Q_2 \cdot y^{H_1(ID_i, Q_2)}$.
  - Check if $g^{\mu_1} \overset{?}{=} (T_1)(R_1)^{H_6(ID_i, P_1, T_1)}$, $g^{\mu_2} \overset{?}{=} (T_2)(R_2)^{H_6(ID_i, P_2, T_2)}$, $g^{S_3} \overset{?}{=} (Q_3)(y^{H_2(ID_i, Q_1, Q_2, Q_3)})$.
  - If all the above checks are satisfied, return *success*, else return $failure$.
- **SetPrivateKey**$(ID_i, PSK, USK, params)$:
  - Output the full secret key of the identity $ID_i$ as $SK = (U_1, U_2, S_1, S_2)$.
- **Re-KeyGen**$(ID_i, ID_j, SK_i, PK_j, params)$:
  - Check the validity of the public key of $ID_j$ by verifying if *Public* $Verify(ID_j, PK_j, params) = success$. If the check fails, return $\perp$.
  - Compute $R_{j,1} = Q_{j,1}(y^{H_1(ID_j, Q_{j,1})})$, $X_1 = P_{j,1}(R_{j,1}^{H(P_{j,1})})$.
  - Compute $X = P_{j,1}(P_{j,2})^{H(P_{j,1})}$ and $\alpha = H(X)$.
  - Select $h \in_R \{0,1\}^{l_0}$ and $\pi \in_R \{0,1\}^{l_1}$. Compute $v = H_4(h, \pi)$.
  - Compute $V = (X_1)^v$, $W = H_3(g^v) \oplus (h||\pi)$.
  - Compute $rk = \frac{h}{U_{i,1} + H(P_{i,1})U_{i,2} + \alpha(S_{i,1} + H(R_{i,1})S_{i,2})}$.
  - Output the re-encryption key $RK_{i \to j} = (rk, V, W)$.
- **Encrypt**$(ID_i, PK_i, m, params)$:
  - Check the validity of the public key $PK_i$ by verifying if *Public* $Verify(ID_i, PK_i, params) = success$. If the check fails, output $\perp$.
  - Compute $R_{i,1} = Q_{i,1}(y^{H_1(ID_i, Q_{i,1})})$, $R_{i,2} = Q_{i,2}(y^{H_1(ID_i, Q_{i,2})})$, $X = P_{i,1}(P_{i,2})^{H(P_{i,1})}$, $Y = R_{i,1}(R_{i,2})^{H(R_{i,1})}$, $\alpha = H(X)$ and set $Z = (X(Y)^\alpha)$.
  - Select $u \in_R \mathbb{Z}_q^*$ and $\omega \in_R \{0,1\}^{l_1}$. Compute $r = H_4(m, \omega)$.
  - Compute $D = (Z)^u$, $E = Z^r$, $F = H_3(g^r) \oplus (m||\omega)$, $s = u + rH_5(D, E, F)$.
  - Return the ciphertext $C = (D, E, F, s)$ as the first level ciphertext.
- **Re-Encrypt**$(ID_i, ID_j, C, RK_{i \to j}, params)$:
  - Check validity of the ciphertext by computing $Z$ as shown in $Encrypt(ID_i, PK_i, m, params)$ and performing the following checks.

    $$(Z)^s \overset{?}{=} D \cdot E^{H_5(D,E,F)} \tag{1}$$

    If the check fails, return $\perp$.
  - Else, compute $E' = R^{rk}$.
  - Output $D = (E', F, V, W)$ as the second level ciphertext.
- **Decrypt**$(ID_i, SK_i, C, params)$:
  - Obtain the public key $PK_i$ corresponding to $ID_i$. Check validity of the ciphertext by checking if Eq. 1 holds. If the check fails, output $\perp$.
  - Else, compute $R_{i,1} = Q_{i,1}(y^{H_1(ID_i, Q_{i,1})})$, $R_{i,2} = Q_{i,2}(y^{H_1(ID_i, Q_{i,2})})$, $X = P_{i,1}(P_{i,2})^{H(P_{i,1})}$, $Y = R_{i,1}(R_{i,2})^{H(R_{i,1})}$, $\alpha = H(X)$ and set $Z = (X(Y)^\alpha)$. Set $K = U_{i,1} + H(P_{i,1})U_{i,2} + \alpha(S_{i,1} + H(R_{i,1})S_{i,2})$.

  - Compute $(m||\omega) = F \oplus H_3(E^{\frac{1}{K}})$. Output $m$ if $E \stackrel{?}{=} (Z)^{H_4(m,\omega)}$ holds. Else, return $\perp$.
- **Re-Decrypt**$(ID_j, SK_j, D, params)$:
  - Compute $R_{j,1} = Q_{j,1}(y^{H_1(ID_j,Q_{j,1})})$, $X_1 = P_{j,1}(R_{j,1}^{H(P_{j,1})})$.
  - Compute $(h||\pi) = W \oplus H_3(V^{\frac{1}{U_{j,1}+H(P_{j,1})S_{j,1}}})$ and $(m||\omega) = F \oplus H_3(E'^{1/h})$.
  - Output $m$ if $V \stackrel{?}{=} (X)_1^{H_4(h,\pi)}$, $E' \stackrel{?}{=} g^{h(H_4(m,\omega))}$. Else, return $\perp$.

## 3.2 Our Attack

In this section, we highlight the flaw in the security reduction of the CLPRE scheme due to Srinivasan *et al.* [10]. We demonstrate that the simulation of the random oracles does not comply with the real system due to which, the adversary can distinguish the simulation of the challenger from the real system. Note that the flaw is observed in the proof for both $Type - I$ and $Type - II$ adversary and we refer to both the two types of adversaries as $\mathcal{A}$ in general. Consider that the adversary constructs a first level dummy ciphertext $C_d = (D, E, F, s)$ in the following way under a public key $PK_i$. We use $Encrypt_{fake}$ to denote this technique to construct dummy ciphertexts.

- Compute $Z$ using $Encrypt(ID_i, PK_i, m, params)$ algorithm.
- Select $u \in_R \mathbb{Z}_q^*$ and compute $D = (Z)^u$.
- Pick $r \in_R \mathbb{Z}_q^*$ and compute $E = (Z)^r$.
- Choose $F \in_R \{0,1\}^{l_0+l_1}$.
- Compute $s = u + rH_5(D, E, F) \mod q$.

Note that the computation of $F$ and $r$ in $C_d$ using $Encrypt_{fake}$ violates the definition of the $Encrypt(ID_i, PK_i, m, params)$ algorithm. But $C_d$ clears the ciphertext validity check of Eq. (1). The decryption algorithm $Decrypt(ID_i, SK_i, C_d, params)$ detects the ciphertext $C_d$ as invalid and returns $\perp$. However, the $ReEncrypt(ID_i, ID_j, C_d, RK_{i\to j}, params)$ algorithm **accepts** $C_d$ as a valid ciphertext. We use this knowledge to construct a distinguisher for the simulated environment from the real system described stepwise as follows:

1. After the *Challenge* phase, $\mathcal{A}$ generates a dummy ciphertext $C_1 = (D_1, E_1, F_1, s)$ under the target identity $PK_{ch}$ using $Encrypt_{fake}$ as shown:
   - Compute $Z_{ch}$ using $Encrypt(ID_i, PK_i, m, params)$ algorithm.
   - Select $u_1 \in_R \mathbb{Z}_q^*$ and compute $D_1 = (Z_{ch})^{u_1}$.
   - Pick $r_1 \in_R \mathbb{Z}_q^*$ and compute $E_1 = (Z_{ch})^{r_1}$.
   - Choose $F_1 \in_R \{0,1\}^{l_0+l_1}$.
   - Compute $s_1 = u_1 + r_1 H_5(D_1, E_1, F_1) \mod q$.
2. $\mathcal{A}$ generates another dummy ciphertext $C_2 = (D_2, E_2, F_2, s_2)$ in the same way described above considering random values $r_2 \in_R \mathbb{Z}_q^*$ and $F_2 \in_R \{0,1\}^{l_0+l_1}$.
3. $\mathcal{A}$ queries the re-encryption oracle $\mathcal{O}_{renc}(ID_{ch}, ID_j, C_1, RK_{ch\to j})$. As per $\mathcal{O}_{renc}$, $\mathcal{C}$ searches the $H_4$ list for a tuple of the form $(\langle m, \omega\rangle, r)$ such that $E_1 = (Z_{ch}^r)$. If no such tuple exists, $\mathcal{O}_{renc}$ outputs $\perp$. Note that, on an output $\perp$, $\mathcal{A}$ can distinguish between the simulation and

the real system, since $C_1$ is a valid ciphertext as per the definition of $ReEncrypt(ID_{ch}, ID_j, C, RK_{ch\to j}, params)$ algorithm and should produce a valid second level ciphertext $D_1$.

- If $\mathcal{O}_{renc}$ returns $\perp$, $\mathcal{A}$ aborts.
- Else, $\mathcal{O}_{renc}$ computes $D_1 = (E'_1, F_1, V_1, W_1)$ and outputs $D_1$.

4. Similarly, $\mathcal{A}$ queries the re-encryption oracle $\mathcal{O}_{renc}(ID_{ch}, ID_j, C_2, RK_{ch\to j})$. As per $\mathcal{O}_{renc}$, $\mathcal{C}$ searches the $H_4$ list for a tuple of the form $(\langle m, \omega\rangle, r)$ such that $E_2 = (Z^r_{ch})$. If no such tuple exists, $\mathcal{O}_{renc}$ outputs $\perp$. Note that, on an output $\perp$, $\mathcal{A}$ can distinguish between the simulation and the real system, since $C_1$ is a valid ciphertext as per the definition of $ReEncrypt(ID_{ch}, ID_j, C, RK_{ch\to j}, params)$ and should produce a valid second level ciphertext $D_2$.
   - If $\mathcal{O}_{renc}$ returns $\perp$, $\mathcal{A}$ aborts.
   - Else, $\mathcal{O}_{renc}$ computes $D_2 = (E'_2, F_2, V_2, W_2)$ and outputs $D_2$.
5. On receiving $D_1$ and $D_2$, $\mathcal{A}$ computes $T_1 = {E'}_1^{\frac{1}{r_1}}$ and $T_2 = {E'}_2^{\frac{1}{r_2}}$.
6. If $T_1 \overset{?}{=} T_2$ does not hold, $ReEncrypt(ID_{ch}, ID_j, C, RK_{ch\to j}, params) \neq \mathcal{O}_{renc}$, $\mathcal{A}$ learns it is not the real system and aborts. Else, if $T_1 \overset{?}{=} T_2$ holds, $\mathcal{A}$ cannot distinguish between the simulated environment and real system.

## 3.3 A Possible Fix

The flaw in the scheme can be fixed by modifying the encryption algorithm **Encrypt**$(ID_i, PK_i, m, params)$ with additional ciphertext validity checks in both **Re-Encrypt** and the **Decrypt**. The modified scheme is shown below.

- **Setup**$(1^\lambda)$: The *Setup* algorithm remains the same as in [10] described in Sect. 3.1. Add another cryptographic hash function to the existing public parameters as defined:

$$\tilde{H} : \mathbb{G}^4 \times \{0,1\}^{l_0+l_1} \to \mathbb{G}$$

  Return public parameters $params = (p, q, \mathbb{G}, g, y, \tilde{H}, H, H_1, H_2, H_3, H_4, H_5, H_6)$ and the master secret key is $msk = x$, generated as described in Sect. 3.1.
- The **PartialKeyExtract**, **UserKeyGen**, **SetPublicKey**, **Public Verify**, **SetPrivateKey**, **Re-KeyGen** algorithms are the same as described in Sect. 3.1.
- **Encrypt**$(ID_i, PK_i, m, params)$:
  - Check the validity of the public key $PK_i$ by verifying if **Public Verify**$(ID_i, PK_i) = success$. If the check fails, output $\perp$.
  - Compute $R_{i,1} = Q_{i,1}(y^{H_1(ID_i, Q_{i,1})})$, $R_{i,2} = Q_{i,2}(y^{H_1(ID_i, Q_{i,2})})$, $X = P_{i,1}(P_{i,2})^{H(P_{i,1})}$, $Y = R_{i,1}(R_{i,2})^{H(R_{i,1})}$, $\alpha = H(X)$ and set $Z = (X(Y)^\alpha)$.
  - Select $u \in_R \mathbb{Z}_q^*$ and $\omega \in_R \{0,1\}^{l_1}$. Compute $r = H_4(m, \omega)$.
  - Compute $D = (Z)^u$, $E = Z^r$.
  - Compute $\bar{D} = \tilde{H}(X, Y, D, E, F)^u$, $\bar{E} = \tilde{H}(X, Y, D, E, F)^r$.
  - Compute $F = H_3(g^r) \oplus (m\|\omega)$ and $s = u + rH_5(E, \bar{E}, F)$.
  - Return the ciphertext $C = (E, \bar{E}, F, s)$ as the first level ciphertext.

- **Re-Encrypt**$(ID_i, ID_j, C, RK_{i\to j}, params)$: On input of a re-encryption key $RK_{i\to j} = (RK^{\langle 1\rangle}_{i\to j}, V, W)$, a first level ciphertext $C = (E, \bar{E}, F, s)$ encrypted under $PK_i$, obtain a second level ciphertext $D$ under $PK_j$ as follows:
  - Compute $R_{i,1} = Q_{i,1}(y^{H_1(ID_i, Q_{i,1})})$, $R_{i,2} = Q_{i,2}(y^{H_1(ID_i, Q_{i,2})})$, $X = P_{i,1}(P_{i,2})^{H(P_{i,1})}$, $Y = R_{i,1}(R_{i,2})^{H(R_{i,1})}$, $\alpha = H(X)$ and set $Z = (X(Y)^{\alpha})$.
  - Compute $D$ and $\bar{D}$ as follows:

$$\begin{aligned} D &= (Z)^s \cdot (E^{H_5(E,\bar{E},F)})^{-1} \\ &= Z^u \cdot Z^{r\cdot H_5(E,\bar{E},F)} \cdot Z^{-r\cdot H_5(E,\bar{E},F)} \\ &= (Z)^u. \\ \bar{D} &= \tilde{H}(X,Y,D,E,F)^s \cdot (\bar{E}^{H_5(E,\bar{E},F)})^{-1} \\ &= \tilde{H}(X,Y,D,E,F)^{u+r\cdot H_5(E,\bar{E},F)} \cdot \tilde{H}(X,Y,D,E,F)^{-r\cdot H_5(E,\bar{E},F)} \\ &= \tilde{H}(X,Y,D,E,F)^u. \end{aligned}$$

  - Check the validity of the ciphertext by performing the following checks.

$$(Z)^s \stackrel{?}{=} D \cdot E^{H_5(E,\bar{E},F)} \tag{2}$$

$$\tilde{H}(X,Y,D,E,F)^s \stackrel{?}{=} \bar{D} \cdot \bar{E}^{H_5(E,\bar{E},F)} \tag{3}$$

    If the check fails, return $\perp$.
  - Else, parse $RK_{i\to j}$ as $(rk, V, W)$ compute $E' = R^{rk}$.
  - Output $D = (E', F, V, W)$ as the second level ciphertext.
- **Decrypt**$(ID_i, SK_i, C, params)$:
  - Obtain the public key $PK_i$ corresponding to $ID_i$. Check if the ciphertext is well-formed by computing the values of $D$ and $\bar{D}$ and checking if Eqs. 2 and 3 holds. If they do not hold, return $\perp$.
  - Else, compute $R_{i,1}, R_{i,2}, X, Y, \alpha, Z, K$ and retrieve $m$ as described in the *Decrypt*$(ID_i, SK_i, C, params)$ algorithm in Sect. 3.1.
- **Re-Decrypt**$(ID_j, SK_j, D, params)$: Same as described in in Sect. 3.1.

## 4 Our Unidirectional CCA-secure CLPRE Scheme

### 4.1 Our Scheme

- **Setup**$(1^{\lambda})$: Given $\lambda$ as the security parameter, choose a group $\mathbb{G}$ of prime order $q$. Let $P$ be a generator of $\mathbb{G}$. Pick $s \in_R \mathbb{Z}_q^*$ and compute $P_{pub} = sP$. Choose cryptographic hash functions:

$$\begin{aligned} \tilde{H} &: \{0,1\}^{l_{ID}} \times \mathbb{G}^2 \times \{0,1\}^{l_0+l_1} \to \mathbb{G} \\ H_1 &: \{0,1\}^{l_{ID}} \times \mathbb{G}^2 \to \mathbb{Z}_q^* \\ H_2 &: \mathbb{Z}_q^* \times \mathbb{Z}_q^* \to \mathbb{Z}_q^* \\ H_3 &: \mathbb{G} \to \mathbb{Z}_q^* \\ H_4 &: \{0,1\}^{l_0} \times \{0,1\}^{l_1} \to \mathbb{Z}_q^* \\ H_5 &: \mathbb{G}^2 \to \{0,1\}^{l_0+l_1} \\ H_6 &: \mathbb{G}^2 \times \{0,1\}^{l_0+l_1} \to \mathbb{Z}_q^* \end{aligned}$$

where $\{0,1\}^{l_0}$ is the size of the message space $\mathcal{M}$, $l_1$ is determined by the security parameter $\lambda$ and $\{0,1\}^{l_{ID}}$ is the size of the identity of a user. Return the public parameters $params = (\mathbb{G}, q, P, P_{pub}, \tilde{H}, H_1, H_2, H_3, H_4, H_5, H_6)$ and master secret key $msk = s$.

- **PartialKeyExtract**$(msk, ID_i, params)$:
  - Choose $x_i, y_i \in_R \mathbb{Z}_q^*$.
  - Compute $X_i = x_i P,\ Y_i = y_i P$.
  - Compute $q_i = H_1(ID_i, X_i, Y_i)$.
  - Compute $d_i = (x_i + q_i s) \mod q$.
  - Return the Partial Public Key $PPK_i = (X_i, Y_i, d_i)$ and the Partial Private Key $PSK_i = y_i$.
- **UserKeyGen**$(ID_i, params)$:
  - Pick $z_i \in_R \mathbb{Z}_q^*$.
  - Compute $Z_i = z_i P$.
  - Return the user private key-public key pair $(USK_i, UPK_i) = (z_i, Z_i)$.
- **SetPrivateKey**$(ID_i, PSK_i, USK_i, params)$: Set the full secret key as $SK_i = \langle z_i, y_i \rangle$.
- **SetPublicKey**$(ID_i, PPK_i, PSK_i, UPK_i, USK_i, params)$: Set the full public key as $PK_i = \langle X_i, Y_i, Z_i, d_i \rangle$.
- **PublicVerify**$(ID_i, PK_i, params)$: We additionally provide public verifiability of the public keys of each user. This is done by the following check:

$$d_i P \stackrel{?}{=} X_i + H_1(ID_i, X_i, Y_i) \cdot Ppub \tag{4}$$

If the check is satisfied, return *valid*, else return *invalid*.

*Remark 1.* Our public key verification algorithm $PublicVerify(ID_i, PK_i, params)$ ensures the validity of the public keys, since an adversary can replace the public keys with false keys of its choice.

- **Re-KeyGen**$(ID_i, ID_j, SK_i, PK_j, params)$:
  - Pick $\alpha_{ij}^{(1)}, \beta_{ij}^{(1)} \in_R \mathbb{Z}_q^*$.
  - Compute $\alpha_{ij}^{(2)}$ such that $\alpha_{ij}^{(1)} \cdot \alpha_{ij}^{(2)} = y_i \mod q$.
  - Compute $\beta_{ij}^{(2)}$ such that $\beta_{ij}^{(1)} \cdot \beta_{ij}^{(2)} = z_i \mod q$.
  - Compute $v_{ij} = H_2(\alpha_{ij}^{(2)} || \beta_{ij}^{(2)})$.
  - Compute $V_{ij} = v_{ij} \cdot Y_j$ and $W_{ij} = H_3(v_{ij} P) \oplus (\alpha_{ij}^{(2)} || \beta_{ij}^{(2)})$.
  - Return $RK_{i \to j} = (\alpha_{ij}^{(1)}, \beta_{ij}^{(1)}, V_{ij}, W_{ij})$.
- **Encrypt**$(ID_i, PK_i, m, params)$:
  - Check the validity of the public key of identity $ID_i$ by checking if **PublicVerify(**$ID_i, PK_i, params$)=*valid*.
  - If *invalid*, return $\perp$.
  - Else, pick $\sigma \in_R \{0,1\}^{l_1},\ u \in_R \mathbb{Z}_q^*$.
  - Compute $r = H_4(m, \sigma) \in \mathbb{Z}_q^*$.
  - Compute the ciphertext $C = (C_1, C_2, C_3, C_4)$ where:
    Compute $C_1 = rP \in \mathbb{G}$.

Compute $\overline{C}_1 = uP \in \mathbb{G}$.
Compute $C_2 = r\tilde{H}(ID_i, C_1, \overline{C}_1, C_3) \in \mathbb{G}$.
Compute $\overline{C}_2 = u\tilde{H}(ID_i, C_1, \overline{C}_1, C_3) \in \mathbb{G}$.
Compute $C_3 = H_5(rY_i, rZ_i) \oplus (m||\sigma) \in \{0,1\}^{l_0+l_1}$.
Compute $C_4 = u + rH_6(C_1, C_2, C_3) \in \mathbb{Z}_q^*$.

– Return $C = (C_1, C_2, C_3, C_4)$.

- **Re-Encrypt**$(ID_i, ID_j, C, RK_{i\to j}, params)$: To verify that $C$ is well-formed, compute $\overline{C}_1$ and $\overline{C}_2$ as given:

$$\begin{aligned}\overline{C}_1 &= C_4P - H_6(C_1, C_2, C_3)\cdot C_1\\ &= uP + H_6(C_1, C_2, C_3)rP - H_6(C_1, C_2, C_3)\cdot C_1\\ &= uP.\\ \overline{C}_2 &= C_4\cdot\tilde{H}(ID_i, C_1, \overline{C}_1, C_3) - H_6(C_1, C_2, C_3)\cdot C_2\\ &= (u + rH_6(C_1, C_2, C_3))\tilde{H}(ID_i, C_1, \overline{C}_1, C_3) - H_6(C_1, C_2, C_3)\cdot C_2\\ &= u\tilde{H}(ID_i, C_1, \overline{C}_1, C_3).\end{aligned}$$

We verify if the ciphertext is well-formed by performing the following checks:

$$C_4\cdot P \overset{?}{=} \overline{C}_1 + H_6(C_1, C_2, C_3)\cdot C_1 \tag{5}$$

$$C_4\cdot\tilde{H}(ID_i, C_1, \overline{C}_1, C_3) \overset{?}{=} \overline{C}_2 + H_6(C_1, C_2, C_3)\cdot C_2 \tag{6}$$

If verification is successful, do the following computation:

– Compute $D_1 = \alpha_{ij}^{(1)}\cdot C_1$.
– Compute $D_2 = \beta_{ij}^{(1)}\cdot C_1$.
– Return the re-encrypted ciphertext as $D = (D_1, D_2, D_3, D_4, D_5) = (D_1, D_2, C_3, V_{ij}, W_{ij})$.

- **Decrypt**$(ID_i, SK_i, C, params)$: Verify that $C$ is a valid ciphertext by checking if Eqs. 5 and 6 holds. If satisfied, compute $m$ using:

$$(m||\sigma) = C_3 \oplus H_5(y_i\cdot C_1, z_i\cdot C_1) \tag{7}$$

- **Re-Decrypt**$(ID_j, SK_j, D, params)$:

– Compute $(\alpha_{ij}^{(2)}||\beta_{ij}^{(2)}) = W_{ij} \oplus H_3(\frac{1}{y_j}V_{ij})$.
– Check if $V_{ij} \overset{?}{=} H_2(\alpha_{ij}^{(2)}||\beta_{ij}^{(2)})\cdot Y_j$.
– If satisfied, compute $m$ as:

$$(m||\sigma) = C_3 \oplus H_5\big(\alpha_{ij}^{(2)}\cdot D_1, \beta_{ij}^{(2)}\cdot D_2\big) \tag{8}$$

### 4.2 Correctness

Due to space constraints, the correctness of our scheme appears in the full version of the paper [8].

### 4.3 Security Proof

**First-level Ciphertext Security Against Type I Adversary**

**Theorem 1.** *Our proposed scheme is CCA-secure against Type-I adversary for the first level ciphertext under the CDH assumption and the $EUF-CMA$ security of Schnorr signature scheme [7]. If a $(t,\epsilon)IND-CLPRE-CCA$ Type-I adversary $\mathcal{A}_I$ with an advantage $\epsilon$ breaks the IND-CLPRE-CCA security of the given scheme, $\mathcal{C}$ can solve the CDH problem with advantage $\epsilon'$ within time $t'$ where:*

$$\epsilon' \geq \frac{1}{q_{H_5}}\Bigg(\frac{(1-\omega)^{1+q_{rk}}\epsilon}{e(q_{ppe}+1)} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{q_{H_6}}{2^{l_0+l_1}} - \frac{q_{\tilde{H}}}{2^{l_0+l_1}} - q_{dec}\Big(\frac{q_{H_5}/q}{1-(q_{H_4}/(2^{l_0+l_1}))} + \frac{q_{H_4}/(2^{l_0+l_1})}{1-(q_{H_5}/q)} + \frac{2}{q}\Big)\Bigg)$$

*where $\omega$ is the advantage of an attacker against the EUF-CMA security game of the Schnorr signature scheme and $e$ is the base of the natural logarithm. Time taken by $\mathcal{C}$ to solve the CDH problem is:*

$$t' \leq t + (T_q)O(1) + (T_{\mathcal{O}})t_{exp}$$

*where $T_q = q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6}$, $T_{\mathcal{O}} = 4t_{pe} + 4t_{ppe} + 4t_{ue} + 2t_{rk} + 8t_{re} + 8t_{dec} + 6t_{redec}$. We denote the time taken for exponentiation operation in group $\mathbb{G}$ as $t_{exp}$.*

*Proof.* Due to space constraints, the proof of the theorem is given in the full version of this paper [8].

**Second-level Ciphertext Security Against Type I Adversary**

**Theorem 2.** *Our proposed scheme is CCA-secure against Type-I adversary for the second level ciphertext under the CDH assumption and the $EUF-CMA$ security of the Schnorr signature scheme. If a $(t,\epsilon)IND-CLPRE-CCA$ Type-I adversary $\mathcal{A}_I$ with an advantage $\epsilon$ breaks the IND-CLPRE-CCA security of the given scheme, $\mathcal{C}$ can solve the CDH problem with advantage $\epsilon'$ within time $t'$ where:*

$$\epsilon' \geq \frac{1}{q_{H_5}}\Bigg(\frac{2(1-\omega)^{2+q_{rk}}\epsilon}{e(q_{ppe}+2)^2} - q_{dec}\Big(\frac{q_{H_5}/q}{1-(q_{H_4}/(2^{l_0+l_1}))} + \frac{q_{H_4}/(2^{l_0+l_1})}{1-(q_{H_5}/q)} + \frac{2}{q}\Big)\Bigg)$$

*where $\omega$ is the advantage of an attacker against the EUF-CMA security game of the Schnorr signature scheme and $e$ is the base of the natural logarithm. Time taken by $\mathcal{C}$ to solve the CDH problem is:*

$$t' \leq t + (T_q)O(1) + (T_{\mathcal{O}})t_{exp}$$

*where $T_q = q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6}$, $T_{\mathcal{O}} = 4t_{pe} + 4t_{ppe} + 4t_{ue} + 2t_{rk} + 8t_{re} + 8t_{dec} + 6t_{redec}$. We denote the time taken for exponentiation operation in group $\mathbb{G}$ as $t_{exp}$.*

*Proof.* Due to space constraints, the proof of the theorem is given in the full version of this paper [8].

**First-level Ciphertext Security Against Type II Adversary**

**Theorem 3.** *Our proposed scheme is CCA-secure against Type-II adversary for the first level ciphertext under the CDH assumption and the $EUF-CMA$ security of the Schnorr signature scheme. If a $(t, \epsilon) IND-CLPRE-CCA$ Type-II adversary $\mathcal{A}_{II}$ with an advantage $\epsilon$ breaks the IND-CLPRE-CCA security of the given scheme, $\mathcal{C}$ can solve the CDH problem with advantage $\epsilon'$ within time $t'$ where:*

$$\epsilon' \geq \frac{1}{q_{H_2}}\left(\frac{(1-\omega)^{1+q_{rk}}\epsilon}{e(q_{ppe}+q_{ue}+1)} - \frac{q_{H_4}}{2^{l_0+l_1}} - \frac{q_{H_6}}{2^{l_0+l_1}} - \frac{q_{\tilde{H}}}{2^{l_0+l_1}} - q_{dec}\left(\frac{q_{H_5}/q}{1-(q_{H_4}/(2^{l_0+l_1}))} + \frac{q_{H_4}/(2^{l_0+l_1})}{1-(q_{H_5}/q)} + \frac{2}{q}\right)\right)$$

*where $\omega$ is the advantage of an attacker against the EUF-CMA security game of the Schnorr signature scheme and $e$ is the base of the natural logarithm. Time taken by $\mathcal{C}$ to solve the CDH problem is:*

$$t' \leq t + (T_q)O(1) + (T_{\mathcal{O}})t_{exp}$$

*where $T_q = q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6}$, $T_{\mathcal{O}} = 4t_{pe} + 4t_{ppe} + 4t_{ue} + 2t_{rk} + 8t_{re} + 8t_{dec} + 6t_{redec}$. We denote the time taken for exponentiation operation in group $\mathbb{G}$ as $t_{exp}$.*

*Proof.* Due to space constraints, the proof of the theorem is given in the full version of this paper [8].

**Second-level Ciphertext Security Against Type II Adversary**

**Theorem 4.** *Our proposed scheme is CCA-secure against Type-II adversary for the second level ciphertext under the CDH assumption and the $EUF-CMA$ security of the Schnorr signature scheme. If a $(t, \epsilon) IND-CLPRE-CCA$ Type-II adversary $\mathcal{A}_{II}$ with an advantage $\epsilon$ breaks the IND-CLPRE-CCA security* *of the given scheme, $\mathcal{C}$ can solve the CDH problem with advantage $\epsilon'$ within time $t'$ where:*

$$Pr[E_{H_5^*}] \geq \frac{2(1-\omega)^{2+q_{rk}}}{e(q_{ppe}+q_{ue}+2)^2} - q_{dec}\left(\frac{q_{H_5}/q}{1-(q_{H_4}/(2^{l_0+l_1}))} + \frac{q_{H_4}/(2^{l_0+l_1})}{1-(q_{H_5}/q)} + \frac{2}{q}\right)$$

*where $\omega$ is the advantage of an attacker against the EUF-CMA security game of the Schnorr signature scheme and $e$ is the base of the natural logarithm. Time taken by $\mathcal{C}$ to solve the CDH problem is:*

$$t' \leq t + (T_q)O(1) + (T_{\mathcal{O}})t_{exp}$$

*where $T_q = q_{\tilde{H}} + q_{H_1} + q_{H_2} + q_{H_3} + q_{H_4} + q_{H_5} + q_{H_6}$, $T_{\mathcal{O}} = 4t_{pe} + 4t_{ppe} + 4t_{ue} + 2t_{rk} + 8t_{re} + 8t_{dec} + 6t_{redec}$. We denote the time taken for exponentiation operation in group $\mathbb{G}$ as $t_{exp}$.*

*Proof.* Due to space constraints, the proof of the theorem is given in the full version of this paper [8].

## 5 Efficiency Comparison

We give a comparison of the efficiency of our proposed CLPRE scheme with the suggested fix to [10] as described in Sect. 3.3. In Table 1, we show the computational efficiency of our scheme and the modified scheme by comparing the time taken by the different algorithms in our protocols. Note that we use $t_{exp}$ to denote the time required for exponentiation in a group. The comparison reveals that our scheme is more efficient than the existing scheme with our suggested fix.

**Table 1.** Efficiency comparison of the scheme [10] with the suggested fix with our CLPRE scheme indicates that our scheme is more efficient.

| Scheme | Modified CLPRE scheme of Srinivasan *et al.* [10] | Our CLPRE scheme |
|---|---|---|
| Setup | $t_{exp}$ | $t_{exp}$ |
| PartialKeyExtract | $3t_{exp}$ | $2t_{exp}$ |
| UserKeyGen | $2t_{exp}$ | $t_{exp}$ |
| SetPublicKey | $2t_{exp}$ | – |
| PublicVerify | $8t_{exp}$ | $2t_{exp}$ |
| Re-KeyGen | $5t_{exp}$ | $2t_{exp}$ |
| Encrypt | $10t_{exp}$ | $4t_{exp}$ |
| Re-Encrypt | $10t_{exp}$ | $6t_{exp}$ |
| Decrypt | $11t_{exp}$ | $6t_{exp}$ |
| Re-Decrypt | $6t_{exp}$ | $4t_{exp}$ |

## 6 Conclusion

Although several CLPRE schemes have been proposed in the literature, to the best of our knowledge, only one scheme [6] has reported the certificateless property without any known attacks to the scheme. The scheme is based on costly bilinear pairing operation and satisfies a weaker notion of security, termed as RCCA security. Recently, Srinivasan *et al.* [10] proposed a CLPRE scheme without resorting to bilinear pairing in the random oracle model. However, we demonstrated that their security proof is flawed by presenting a concrete attack. We then presented a unidirectional CLPRE scheme which is pairing-free and satisfies CCA-security against both the Type-I and Type-II adversaries for the first and second level ciphertexts. We remark that a potential fix to [10] is also suggested in our paper but our proposed algorithm is more efficient as noted from our efficiency comparison. Our work affirmatively resolves the problems faced by PKI-based and IB-based PRE schemes by proposing an efficient pairing-free certificateless Proxy Re-encryption scheme.

## References

1. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003). doi:10.1007/978-3-540-40061-5_29
2. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: IN NDSS (2005)
3. Ateniese, G., Kevin, F., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Tran. Inf. Syst. Secur. (TISSEC) **9**(1), 1–30 (2006)
4. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998). doi:10.1007/BFb0054122
5. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12678-9_19
6. Guo, H., Zhang, Z., Zhang, J., Chen, C.: Towards a secure certificateless proxy re-encryption scheme. In: Susilo, W., Reyhanitabar, R. (eds.) ProvSec 2013. LNCS, vol. 8209, pp. 330–346. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41227-1_19
7. Schnorr, C.-P.: Efficient signature generation by smart cards. J. Cryptol. **4**(3), 161–174 (1991)
8. Sharmila Deva Selvi, S., Paul, A., Pandu Rangan, C.: An efficient certificateless proxy re-encryption scheme without pairing. Cryptology ePrint Archive, Report 2017/768 (2017). http://eprint.iacr.org/2017/768
9. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_5
10. Srinivasan, A., Pandu Rangan, C.: Certificateless proxy re-encryption without pairing: revisited. In: Proceedings of the 3rd International Workshop on Security in Cloud Computing, SCC@ASIACCS 2015, Singapore, Republic of Singapore, 14 April 2015, pp. 41–52 (2015)

11. Sur, C., Jung, C.D., Park, Y., Rhee, K.H.: Chosen-ciphertext secure certificateless proxy re-encryption. In: De Decker, B., Schaumüller-Bichl, I. (eds.) CMS 2010. LNCS, vol. 6109, pp. 214–232. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13241-4_20
12. Yang, K., Xu, J., Zhang, Z.: Certificateless proxy re-encryption without pairings. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 67–88. Springer, Cham (2014). doi:10.1007/978-3-319-12160-4_5
13. Zheng, Y., Tang, S., Guan, C., Chen, M.-R.: Cryptanalysis of a certificateless proxy re-encryption scheme. In: 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies, Xi'an, Shaanxi, China, 9–11 September 2013, pp. 307–312 (2013)