6-2020

Secure server-aided data sharing clique with attestation

Yujue WANG
Guilin University of Electronic Technology

Hwee Hwa PANG
Singapore Management University, hhpang@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Yong DING
Guilin University of Electronic Technology

Qianhong WU
Beijing University of Aeronautics and Astronautics (Beihang University)

See next page for additional authors

Citation

WANG, Yujue; PANG, Hwee Hwa; DENG, Robert H.; DING, Yong; WU, Qianhong; QIN, Bo; and FAN, Kefeng. Secure server-aided data sharing clique with attestation. (2020). *Information Sciences*. 522, 80-98.
Available at: https://ink.library.smu.edu.sg/sis_research/5122

Author
Yujue WANG, Hwee Hwa PANG, Robert H. DENG, Yong DING, Qianhong WU, Bo QIN, and Kefeng FAN

Published in Information Sciences, Volume 522, June 2020, Pages 80-98
https://doi.org/10.1016/j.ins.2020.02.064

Secure server-aided data sharing clique with attestation

Yujue Wang [a,b], HweeHwa Pang [c], Robert H. Deng [c], Yong Ding [a,d,*], Qianhong Wu [e], Bo Qin [f], Kefeng Fan [g]

[a] Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China
[b] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[c] School of Information Systems, Singapore Management University, Singapore, 188065
[d] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518055, China
[e] School of Cyber Science and Technology, Beihang University, Beijing 100191, China
[f] School of Information, Renmin University of China, Beijing 100872, China
[g] China Electronics Standardization Institute, Beijing 100007, China

A B S T R A C T

In this paper, we consider the security issues in data sharing cliques via remote server. We present a *public key re-encryption scheme with delegated equality test on ciphertexts* (PRE-DET). The scheme allows users to share outsourced data on the server without performing decryption-then-encryption procedures, allows new users to dynamically join the clique, allows clique users to attest the message underlying a ciphertext, and enables the server to partition outsourced user data without any further help of users after being delegated. We introduce the PRE-DET framework, propose a concrete construction and formally prove its security against five types of adversaries regarding two security requirements on message confidentiality and unforgeability of attestation against the server. We also theoretically analyze and compare the proposed PRE-DET construction with related schemes in terms of ciphertext sizes and computation costs of encryption, decryption, ciphertext equality testing and re-encryption, which confirms the practicality of our construction.

Keywords:
Encryption
Re-encryption
Confidentiality
Equality test on ciphertexts
Data sharing
Data outsourcing
Data attestation

1. Introduction

Data outsourcing allows users to engage a remote storage server to hold user data, which relieves users from the overhead of managing their own storage devices. However, for privacy reasons, user data can only be stored on the remote server in ciphertext format. Recently, public key encryption with (authorized/delegated) equality test on ciphertext was proposed to not only guarantee data confidentiality at the remote server, but also enhance the functionality of the server by enabling it to compare outsourced ciphertexts. The property of ciphertext equality test in these encryption schemes has been extensively utilized in achieving controlled equijoin in outsourced relational database [32], partition of encrypted emails [18], searchable encryption and partitioning encrypted data [40], deduplication on outsourced encrypted data [6], data monitoring [31], etc. However, these schemes do not support efficient data sharing.

* Corresponding author at: Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin 541004, China.
E-mail address: stone_dingy@126.com (Y. Ding).

In this paper, we investigate data sharing cliques with the help of a remote server, where the data outsourced by users in a clique are shared only by these users. For example, for users in a workshop, the data can be shared among them to jointly perform a task. Data sharing clique raises the following requirements. First, all user data must be stored in ciphertext format to protect confidentiality and shared by all users in the clique. Second, to realize data sharing, each user does not need to directly produce ciphertext for all expected users; rather, the server will transform the ciphertext for the users in the clique without sacrificing confidentiality. Third, a user is able to add an attestation to a ciphertext with non-repudiation assurance, for instance to inform other users of the truthfulness of authenticity of the underlying message. Fourth, the delegated server is able to test whether the outsourced ciphertexts encrypt the same underlying message without decrypting them, to achieve controlled partitioning on encrypted data.

With the data sharing property, a new user joining the clique will be able to access the data of other peer users in the clique, while his/her data also become accessible by the other users. To the best of our knowledge, there is no existing solution to achieve a secure data sharing clique with all the above requirements. Specifically, existing public key encryption technology with (authorized/delegated) equality test on ciphertexts (PKEET/PKE-AET/PKE-DET) can only realize the first and fourth requirements, whereas existing public key re-encryption technology (PRE) only solves the third requirement.

1.1. Contributions

To address the confidentiality and security issues in data sharing cliques, we present a *public key re-encryption scheme with delegated equality test on ciphertexts* (PRE-DET). Our notion of PRE-DET provides more powerful functionalities than PKEET/PKE-AET/PKE-DET and PRE. In PRE-DET, to share the ciphertexts of user A with user B, user A does not need to retrieve them from the server and perform an encryption under the public key of B. Instead, users A and B will jointly generate a re-encryption key, which enables the server to transform the ciphertexts of A to that of B. All users sharing their data constitute a user clique. When some user delegates the server to perform equality test on his/her ciphertexts by issuing a token, the server is implicitly authorized to compare all data owned by all users in the same clique due to the property of data sharing.

PRE-DET allows users to dynamically join the clique by jointly issuing a re-encryption key, which demonstrates he/she is willing to share data with the other users in the clique. Users are able to attest the message underlying a ciphertext. The attestation must be publicly verifiable, so as to ensure authenticity. Moreover, in a PRE-DET scheme, the attestation cannot obstruct the re-encryption functionality (i.e., the data sharing property). In other words, all attested ciphertexts are also shared by the users in the clique, as the server is able to re-encrypt attested ciphertexts in the same way as re-encrypting ciphertexts.

We formulate the security model of PRE-DET with respect to *five* types of adversaries, representing four security requirements on message *confidentiality* and a requirement on *unforgeability* of attestation. The four message confidentiality requirements capture IND-CCA2 and OW-CCA2 security against the server with/without re-encryption key and token, respectively, whereas the attestation unforgeability requirement is defined against malicious users. We present a concrete PRE-DET construction on symmetric bilinear groups, and prove that it is secure against the five types of adversaries as formalized in the security framework in the random oracle model. Comparison with related schemes show that our PRE-DET construction is practical in applications.

1.2. Related work

Mambo and Okamoto [21] introduced public key proxy encryption, which allows the decryptor to transform his/her ciphertext to that of another decryptor without decryting the original ciphertext. They presented concrete proxy encryption constructions using the ElGamal and RSA cryptosystems. Blaze, Bleumer and Strauss [2] further studied atomic proxy cryptography such that the ciphertext/signature of some public key can be converted into ciphertext/signature of the same message under another public key. These schemes proposed in [2,21] only offer IND-CPA security on ciphertexts.

Following the seminal work of [2,21], a large number of public key re-encryption schemes have been proposed. In [4], Canetti and Hohenberger for the first time introduced an IND-CCA secure public key bidirectional proxy re-encryption scheme in the standard model under the decisional bilinear Diffie-Hellman assumption. Deng et al. [7] and Weng et al. [36] designed IND-CCA secure bidirectional proxy re-encryption schemes without using bilinear pairings, which are thus more efficient than the method in [4]. Libert and Vergnaud [17] first presented IND-CCA secure unidirectional proxy re-encryption schemes in the standard model, which was further enhanced by Seo, Yum and Lee [24]. The CCA-secure unidirectional proxy re-encryption scheme presented by Weng et al. [35] can be proved in the adaptive corruption model.

Green and Ateniese [9] first introduced the notion of identity-based proxy re-encryption (ID-PRE), which eliminates complicated certificate management in the public key setting. Then, Chu and Tzeng [5] considered ID-PRE schemes that are proved to be secure in the standard model. Multi-use and unidirectional ID-PRE schemes were investigated in [30] and [25]; the security of the former is proved in the random oracle and the latter in the standard model. In [42], Zhou et al. proposed a mechanism to allow an authorized proxy to convert a ciphertext in an identity-based broadcast encryption scheme into a ciphertext in an identity-based encryption scheme. Li et al. [13] proposed a novel solution to address the challenge problem of outsourcing computation with stronger attack model in data sharing and privacy-preserving outsourced machine learning. Different from the previous works, Li et al. [14] considered multiple devices and data sources in the attack models.

Recently, the proxy re-encryption technique has been extensively used in applications in clouds. In [41], proxy re-encryption was used to achieve secure cloud-based data sharing, and the authors analyzed a 'pitfall' in the security proof of existing proxy re-encryption schemes. Nuñez, Agudo and Lopez [22] reviewed and compared many typical proxy re-encryption schemes, and applied proxy re-encryption to secure access delegation in clouds. Shao et al. [26] designed a bidirectional proxy re-encryption to secure cloud storage against collusion attacks. Liang and Susilo [16] proposed the first searchable attribute-based proxy re-encryption system to secure electronic personal data in clouds. In [15], Liang et al. addressed the privacy issue in data sharing and conjunctive keyword searching in clouds, which also supports secure keyword update. To address the security and integrity of outsourced data in clouds, Yang et al. [39] proposed a lightweight privacy-preserving delegatable proofs of storage scheme.

The primitive of public key encryption with equality test (PKEET) was first introduced by Yang et al. [40], which supports public equality test even on ciphertexts generated under different public keys. PKEET was extended to support *authorized* or *delegated* equality test on ciphertexts (PKE-AET/PKE-DET) [19,20,28] so that only a tester with valid authorization or delegation is able to compare ciphertexts. In [29], Tang presented an all-or-nothing PKEET (AoN-PKEET), where a tester can be independently authorized by two users to test the equality of their ciphertexts. Slamanig, Spreitzer and Unterluggauer [27] designed a special case of AoN-PKEET [29], called AoN-PKEET* and constructed using the ElGamal encryption scheme [8]. The tester in AoN-PKEET* is only allowed to compare ciphertexts of the same user.

Combining the functionalities of PKEET and identity-based encryption, Ma [18] proposed an identity-based encryption scheme with equality test on ciphertexts (IBEET). Lee et al. [11] analyzed the security of [10] and presented an enhanced PKE-AET scheme. In [12], Lee et al. presented semi-generic constructions for PKEET and IBEET. Wang et al. [34] designed a scheme on asymmetric bilinear groups using the ElGamal scheme, where the confidentiality of ciphertexts and tokens are proved in the standard model. Pang and Ding [23] first researched controlled equijoin in relational databases and designed a secure solution in secret key setting, built on equality test on ciphertext fields in outsourced records. Recently, controlled equijoin in relational databases in public key setting was investigated in [32]. Note that the functionality of equality test on ciphertexts was extensively used to preform deduplication on encrypted data [6,37,38] and secure messaging services [33].

1.3. Paper organization

The remainder of this paper is organized as follows. In Section 2, we formulate the framework of PRE-DET and its security requirements. We present a PRE-DET construction in Section 3, and prove its security in Section 4. We compare the performance of our PRE-DET construction with those of existing schemes in Section 5. Finally, Section 6 concludes the paper.

2. PRE-DET Framework and security definitions

2.1. System model and security requirements

In a data sharing clique, there are two types of entities, that is, many users and a server. Users are data owners who encrypt and deposit data on the server. All the users in the clique share their outsourced data. Suppose there are only two users in the clique. When one user wants to share her data with the other, they and the server jointly generate a re-encryption key to enable the server to translate ciphertexts between them. A new user joining the clique only needs to jointly produce a re-encryption key with an existing user and the server, to start sharing her data as well as accessing the data of the other users in the clique. Users do not need to retrieve the ciphertexts from the server and encrypt the data under all public keys of the other users.

Users in the clique are able to attest an outsourced (shared/re-encrypted) data. Specifically, the user first decrypts the data, determines its truthfulness, and generates an attested ciphertext. In this way, all users in the clique can see the attestation of this data on the server without decrypting it. Note that all attested ciphertexts are also shared by all users in the same clique, which means they also support re-encryption by the server like original ciphertexts. The attestation procedure should not degrade the ciphertext security. A clique user can authorize the server to test whether a pair of original or re-encrypted or attested or re-encrypted attested ciphertexts encrypt the same underlying message. With delegation, the server can partition the outsourced data for clique users by the underlying messages.

In a data sharing clique, the server may be curious about the plaintext messages underlying the outsourced data. At some point of time, it could receive re-encryption keys and equality test tokens from the clique users. Thus, the server may lie in one of the following four status:

1. The server does not hold any re-encryption key or equality test token;
2. The server has a re-encryption key, but no equality test token;
3. The server has an equality test token, but no re-encryption key;
4. The server has both re-encryption key and equality test token.

Some user may also try to attach a fake attestation to outsourced data in the name of others, for example, to claim that some sensible data has a lower sensibility level.

2.2. PRE-DET framework

A PRE-DET scheme for secure data sharing clique consists of the following efficient procedures.

$\texttt{Setup}(1^\lambda) \to \mathsf{par}$: Given security parameter λ, the system setup procedure produces public parameter par. The public parameter par is an implicit input to all the following procedures.

$\texttt{KeyGen}(\mathsf{par}) \to (\mathsf{sk}_i, \mathsf{pk}_i)$: With public parameter par, each user $\mathcal{U}_i$ executes the key generation procedure to produce a pair of secret key sk_i and public key pk_i.

$\texttt{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j) \to \mathsf{rk}_{i\leftrightarrow j}$: With secret keys sk_i and sk_j, users $\mathcal{U}_i$ and $\mathcal{U}_j$, along with the server, jointly run the re-encryption key procedure to produce a bidirectional re-encryption key $\mathsf{rk}_{i\leftrightarrow j}$.

$\texttt{DataEnc}(\mathsf{pk}_i, m) \to C$: With public key pk_i of user $\mathcal{U}_i$, the encryption procedure produces a ciphertext C for input message m.

$\texttt{ReEnc}(\mathsf{rk}_{i\leftrightarrow j}, C_{i_l}) \to C_{j_l}$: With bidirectional re-encryption key $\mathsf{rk}_{i\leftrightarrow j}$ and ciphertext C_{i_l} of user $\mathcal{U}_i$, the server runs the re-encryption procedure to produce a ciphertext C_{j_l} of user $\mathcal{U}_j$ for the same underlying plaintext without decrypting C_{i_l}. Due to the bidirectional property of $\mathsf{rk}_{i\leftrightarrow j}$, the server may also perform $\texttt{ReEnc}(\mathsf{rk}_{i\leftrightarrow j}, C_{j_l}) \to C_{i_l}$.

$\texttt{DataDec}(\mathsf{sk}_i, C) \to m/\perp$: With secret key sk_i, user $\mathcal{U}_i$ runs the decryption procedure on ciphertext C to produce a message m, or $\perp$ that signifies an error in decryption.

$\texttt{Attest}(\mathsf{sk}_i, \mathsf{pk}_i, C) \to A$: With secret key sk_i and public key pk_i of user $\mathcal{U}_i$, $\mathcal{U}_i$ runs the attestation procedure on ciphertext C. In this procedure, user $\mathcal{U}_i$ attaches an attestation $att \in \mathbb{A}$ to the underlying plaintext and produces an attested ciphertext A.

$\texttt{AttReEnc}(\mathsf{rk}_{i\leftrightarrow j}, \mathsf{pk}_i, \mathsf{pk}_\ell, A_{i_l}) \to A_{j_l}$: With bidirectional re-encryption key $\mathsf{rk}_{i\leftrightarrow j}$, two public keys $\mathsf{pk}_i, \mathsf{pk}_\ell$ of users $\mathcal{U}_i$ and $\mathcal{U}_\ell$, and attested ciphertext A_{i_l} of user $\mathcal{U}_i$ where the attestation was added by user $\mathcal{U}_\ell$, the server runs the re-encryption procedure to produce an attested ciphertext A_{j_l} for user $\mathcal{U}_j$, so that user $\mathcal{U}_j$ can decrypt A_{j_l}. Due to the bidirectional property of $\mathsf{rk}_{i\leftrightarrow j}$, the server may also perform $\texttt{AttReEnc}(\mathsf{rk}_{i\leftrightarrow j}, \mathsf{pk}_j, \mathsf{pk}_\ell, A_{j_l}) \to A_{i_l}$.

$\texttt{AttVer}(\mathsf{pk}_i, A) \to 1/0$: With public key pk_i, any user may run the attestation verification procedure on attested ciphertext A. The procedure outputs 1 if the attestation att in A is valid under pk_i, i.e., att was originally added by user $\mathcal{U}_i$, or 0 otherwise.

$\texttt{AttDec}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, A) \to m/\perp$: With public keys pk_i, pk_j and secret key sk_j, user $\mathcal{U}_j$ runs the decryption procedure on attested ciphertext A to produce a message m, or $\perp$ that signifies an error in decryption, where the attestation in A was added by user $\mathcal{U}_i$.

$\texttt{Delegate}(\mathsf{sk}_i) \to \mathsf{tk}_i$: With secret key sk_i, user $\mathcal{U}_i$ runs the delegation procedure to produce a token tk_i to enable the server to perform equality test on ciphertexts of users in the clique.

$\texttt{EqTest}(\mathsf{tk}_i, C_{i_l}/A_{i_l}, \mathsf{tk}_j, C_{j_h}/A_{j_h}) \to 0/1$: With two tokens tk_i and tk_j respectively issued by users $\mathcal{U}_i$ and $\mathcal{U}_j$, the server runs the equality test procedure on two (attested) ciphertexts C_{i_l} (or A_{i_l}) and C_{j_h} (or A_{j_h}). The procedure outputs 1 if C_{i_l} (or A_{i_l}) and C_{j_h} (or A_{j_h}) encrypt the same plaintext; otherwise, the procedure outputs 0.

A valid attestation requires that attested ciphertexts preserve the same properties of re-encryption and equality test as ciphertexts. A PRE-DET scheme must be *sound* in the sense that: (1) Every ciphertext generated by DataEnc or re-encrypted by ReEnc is decryptable by DataDec, and every attested ciphertext generated by Attest or re-encrypted by AttReEnc is decryptable by AttDec; (2) Any two (attested) ciphertexts generated by DataEnc/Attest or re-encrypted by ReEnc/AttReEnc for the same message, must pass the equality test procedure EqTest; (3) The attestation in any attested ciphertext generated by Attest or re-encrypted by AttReEnc is publicly verifiable. Formally, the soundness of a bidirectional PRE-DET scheme can be defined as follows.

Definition 1 (Soundness)**.** A PRE-DET scheme is *sound* if, for any security parameter $\lambda \in \mathbb{N}$ and any public parameter $\mathsf{par} \leftarrow \texttt{Setup}(1^\lambda)$, the following conditions are satisfied:

1. For any secret/public key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \texttt{KeyGen}(\mathsf{par})$ and every message $m \in \mathcal{M}$, $\texttt{DataDec}(\mathsf{sk}_i, \texttt{DataEnc}(\mathsf{pk}_i, m)) = m$.
2. For any $\tau > 1$, any secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i), (\mathsf{sk}_{i+1}, \mathsf{pk}_{i+1}), \cdots, (\mathsf{sk}_{i+\tau}, \mathsf{pk}_{i+\tau}) \leftarrow \texttt{KeyGen}(\mathsf{par})$, any $\pi < \tau$, all re-encryption keys $\mathsf{rk}_{(i+\pi)\leftrightarrow(i+\pi+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{i+\pi}, \mathsf{sk}_{i+\pi+1})$, and every message $m \in \mathcal{M}$, we have
$$\texttt{DataDec}\Big(\mathsf{sk}_{i+\tau}, \texttt{ReEnc}\Big(\mathsf{rk}_{(i+\tau-1)\leftrightarrow(i+\tau)}, \cdots, \texttt{ReEnc}\Big(\mathsf{rk}_{i\leftrightarrow(i+1)}, \texttt{DataEnc}(\mathsf{pk}_i, m)\Big)\cdots\Big)\Big) = m.$$
3. For any $m_1, m_2 \in \mathcal{M}$, any τ_1, $\tau_2 > 1$, any secret/public key pairs $(\mathsf{sk}_{i-\tau_1}, \mathsf{pk}_{i-\tau_1}), \cdots, (\mathsf{sk}_i, \mathsf{pk}_i), (\mathsf{sk}_{j-\tau_2}, \mathsf{pk}_{j-\tau_2}), \cdots,$ $(\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \texttt{KeyGen}(\mathsf{par})$, any $\pi_1 < \tau_1, \pi_2 < \tau_2$, all re-encryption keys $\mathsf{rk}_{(i-\pi_1)\leftrightarrow(i-\pi_1+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{i-\pi_1}, \mathsf{sk}_{i-\pi_1+1})$, $\mathsf{rk}_{(j-\pi_2)\leftrightarrow(j-\pi_2+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{j-\pi_2}, \mathsf{sk}_{j-\pi_2+1})$, if $m_1 = m_2$, then $\texttt{EqTest}(\mathsf{tk}_i, C_1, \mathsf{tk}_j, C_2) = 1$ where
   - $C_1 \leftarrow \texttt{DataEnc}(\mathsf{pk}_i, m_1)$ or
   $$C_1 \leftarrow \texttt{ReEnc}\Big(\mathsf{rk}_{(i-1)\leftrightarrow i}, \cdots, \texttt{ReEnc}\Big(\mathsf{rk}_{(i-\tau_1)\leftrightarrow(i-\tau_1+1)}, \texttt{DataEnc}\Big(\mathsf{pk}_{i-\tau_1}, m_1\Big)\Big)\cdots\Big);$$
   - $C_2 \leftarrow \texttt{DataEnc}(\mathsf{pk}_j, m_2)$ or
   $$C_2 \leftarrow \texttt{ReEnc}\Big(\mathsf{rk}_{(j-1)\leftrightarrow j}, \cdots, \texttt{ReEnc}\Big(\mathsf{rk}_{(j-\tau_2)\leftrightarrow(j-\tau_2+1)}, \texttt{DataEnc}\Big(\mathsf{pk}_{j-\tau_2}, m_2\Big)\Big)\cdots\Big);$$
   - $\mathsf{tk}_i \leftarrow \texttt{Delegate}(\mathsf{sk}_i)$ and $\mathsf{tk}_j \leftarrow \texttt{Delegate}(\mathsf{sk}_j)$.

   Otherwise, $\texttt{EqTest}(\mathsf{tk}_i, C_1, \mathsf{tk}_j, C_2) = 0$.

4. For any secret/public key pair $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \texttt{KeyGen}(\mathsf{par})$, and $A \leftarrow \texttt{Attest}(\mathsf{sk}_i, \mathsf{pk}_i, C)$ for any well-formed ciphertext C with any $att \in \mathbb{A}$, we have $\texttt{AttDec}(\mathsf{pk}_i, \mathsf{sk}_i, A) = m$ if $\texttt{DataDec}(\mathsf{sk}_i, C) = m$.
5. For any $\tau > 1$, any secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i), (\mathsf{sk}_{i+1}, \mathsf{pk}_{i+1}), \cdots, (\mathsf{sk}_{i+\tau}, \mathsf{pk}_{i+\tau}) \leftarrow \texttt{KeyGen}(\mathsf{par})$, any $\pi < \tau$, all re-encryption keys $\mathsf{rk}_{(i+\pi)\leftrightarrow(i+\pi+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{i+\pi}, \mathsf{sk}_{i+\pi+1})$, and $A \leftarrow \texttt{Attest}(\mathsf{sk}_i, \mathsf{pk}_i, C)$ for any well-formed ciphertext C with any $att \in \mathbb{A}$, we have

$$\texttt{AttDec}\Big(\mathsf{pk}_i, \mathsf{pk}_{i+\tau}, \mathsf{sk}_{i+\tau}, \texttt{AttReEnc}\Big(\mathsf{rk}_{(i+\tau-1)\leftrightarrow(i+\tau)},\ \mathsf{pk}_{i+\tau-1}, \mathsf{pk}_i, \texttt{AttReEnc}\Big(\cdots, \texttt{AttReEnc}\Big(\mathsf{rk}_{i\leftrightarrow(i+1)}, \mathsf{pk}_i, \mathsf{pk}_i, A\Big)\cdots\Big)\Big)\Big) = m$$

if $\texttt{DataDec}(\mathsf{sk}_i, C) = m$.
6. For any two well-formed ciphertexts C_1, C_2 with respective $att_1, att_2 \in \mathbb{A}$, any τ_1, $\tau_2 > 1$, any secret/public key pairs $(\mathsf{sk}_{i-\tau_1}, \mathsf{pk}_{i-\tau_1}), \cdots, (\mathsf{sk}_i, \mathsf{pk}_i), (\mathsf{sk}_{j-\tau_2}, \mathsf{pk}_{j-\tau_2}), \cdots, (\mathsf{sk}_j, \mathsf{pk}_j) \leftarrow \texttt{KeyGen}(\mathsf{par})$, any $\pi_1 < \tau_1$, $\pi_2 < \tau_2$, all re-encryption keys $\mathsf{rk}_{(i-\pi_1)\leftrightarrow(i-\pi_1+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{i-\pi_1}, \mathsf{sk}_{i-\pi_1+1})$, $\mathsf{rk}_{(j-\pi_2)\leftrightarrow(j-\pi_2+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{j-\pi_2}, \mathsf{sk}_{j-\pi_2+1})$, if $\texttt{DataDec}(\mathsf{sk}_{i-\tau_1}, C_1) = \texttt{DataDec}(\mathsf{sk}_{j-\tau_2}, C_2) \neq \perp$, then $\texttt{EqTest}(\mathsf{tk}_i, A_1, \mathsf{tk}_j, A_2) = 1$ where

$$\begin{aligned} A_1 \leftarrow\ & \mathrm{AttReEnc}\Big(\mathrm{rk}_{(i-1)\leftrightarrow i}, \mathrm{pk}_{i-1}, \mathrm{pk}_{i-\pi_1}, \mathrm{AttReEnc}\Big(\cdots, \mathrm{AttReEnc}\Big(\mathrm{rk}_{(i-\pi_1)\leftrightarrow(i-\pi_1+1)}, \mathrm{pk}_{i-\pi_1}, \mathrm{pk}_{i-\pi_1}, \\ & \mathrm{Attest}\Big(\mathrm{sk}_{i-\pi_1}, \mathrm{pk}_{i-\pi_1}, \mathrm{ReEnc}\Big(\mathrm{rk}_{(i-\pi_1-1)\leftrightarrow(i-\pi_1)}, \cdots, \quad \mathrm{ReEnc}\Big(\mathrm{rk}_{(i-\tau_1)\leftrightarrow(i-\tau_1+1)}, C_1\Big)\cdots\Big)\Big)\Big)\cdots\Big)\Big), \\ A_2 \leftarrow\ & \mathrm{AttReEnc}\Big(\mathrm{rk}_{(j-1)\leftrightarrow j}, \mathrm{pk}_{j-1}, \mathrm{pk}_{j-\pi_2}, \mathrm{AttReEnc}\Big(\cdots, \mathrm{AttReEnc}\Big(\mathrm{rk}_{(j-\pi_2)\leftrightarrow(j-\pi_2+1)}, \mathrm{pk}_{j-\pi_2}, \mathrm{pk}_{j-\pi_2}, \\ & \mathrm{Attest}\Big(\mathrm{sk}_{j-\pi_2}, \mathrm{pk}_{j-\pi_2}, \mathrm{ReEnc}\Big(\mathrm{rk}_{(j-\pi_2-1)\leftrightarrow(j-\pi_2)}, \cdots, \quad \mathrm{ReEnc}\Big(\mathrm{rk}_{(j-\tau_2)\leftrightarrow(j-\tau_2+1)}, C_2\Big)\cdots\Big)\Big)\Big)\cdots\Big)\Big), \end{aligned}$$

$\mathsf{tk}_i \leftarrow \texttt{Delegate}(\mathsf{sk}_i)$ and $\mathsf{tk}_j \leftarrow \texttt{Delegate}(\mathsf{sk}_j)$.
Otherwise, $\texttt{EqTest}(\mathsf{tk}_i, A_1, \mathsf{tk}_j, A_2) = 0$.
7. For any $\tau \geq 1$, any secret/public key pairs $(\mathsf{sk}_i, \mathsf{pk}_i), \cdots, (\mathsf{sk}_{i+\tau}, \mathsf{pk}_{i+\tau}) \leftarrow \texttt{KeyGen}(\mathsf{par})$, any $\pi < \tau$, all re-encryption keys $\mathsf{rk}_{(i+\pi)\leftrightarrow(i+\pi+1)} \leftarrow \texttt{ReKeyGen}(\mathsf{sk}_{i+\pi}, \mathsf{sk}_{i+\pi+1})$, we have $\texttt{AttVer}(\mathsf{pk}_i, A) = 1$ where

$$A \leftarrow \mathrm{AttReEnc}\Big(\mathrm{rk}_{(i+\tau-1)\leftrightarrow(i+\tau)}, \mathrm{pk}_{i+\tau-1}, \mathrm{pk}_i, \mathrm{AttReEnc}\Big(\cdots, \mathrm{AttReEnc}\Big(\mathrm{rk}_{i\leftrightarrow(i+1)}, \mathrm{pk}_i, \mathrm{pk}_i, \mathrm{Attest}(\mathrm{sk}_i, \mathrm{pk}_i, C_i)\Big)\cdots\Big)\Big)$$

2.3. Security definitions

In this section, we define the security model of PRE-DET to capture the confidentiality requirements formalized in Section 2.1, for Type-1, 2, 3, 4 adversaries against message confidentiality, and Type-5 adversary against attestation unforgeability.

Definition 2 (PD-IND-CCA1 security against Type-1 adversary)**.** Let Γ be a PRE-DET scheme. Suppose $\mathcal{A}_1$ is a probabilistic polynomial-time (PPT) adversary who interacts with a challenger $\mathcal{C}$ to perform the following security game.

Set-up: With a security parameter λ, the challenger runs the $\mathtt{Setup}$ procedure to produce public parameter par, which is given to the adversary.

Phase 1: The adversary is able to adaptively issue the following queries.

- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: With public parameter par, the challenger runs the $\mathtt{KeyGen}$ procedure to produce a pair of secret/public keys $(\mathsf{sk}_i, \mathsf{pk}_i)$, and gives pk_i to $\mathcal{A}_1$.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: With public parameter par, the challenger runs the $\mathtt{KeyGen}$ procedure to produce a pair of secret/public keys $(\mathsf{sk}_i, \mathsf{pk}_i)$, which are given to $\mathcal{A}_1$.
- Decryption query 1 $\mathcal{O}_{dec1}$: For a query (C, pk), if pk was not generated by the $\mathtt{KeyGen}$ procedure, then the challenger returns $\perp$, otherwise the challenger returns $\texttt{DataDec}(\mathsf{sk}, C)$.
- Attestation query $\mathcal{O}_{att}$: For a query (C, pk) where C is a ciphertext under pk, if pk was not generated by the $\mathtt{KeyGen}$ procedure, then the challenger returns $\perp$, otherwise the challenger returns $\texttt{Attest}(\mathsf{sk}, \mathsf{pk}, C)$.
- Decryption query 2 $\mathcal{O}_{dec2}$: For a query $((A, \mathsf{pk}_i), \mathsf{pk}_j)$ where A is an attested ciphertext under pk_j such that the attestation was originally added by user $\mathcal{U}_i$, if either pk_i or pk_j was not generated by the $\mathtt{KeyGen}$ procedure, then the challenger returns $\perp$, otherwise the challenger returns $\texttt{AttDec}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, A)$.

Challenge: At the end of Phase 1, the adversary outputs two messages $m_0, m_1 \in_R \mathcal{M}$ and a challenge public key pk^*, where pk^* is the public key of an uncorrupted user. The challenger chooses a random value $b \in_R \{0, 1\}$, computes $C^* \leftarrow \texttt{DataEnc}(\mathsf{pk}^*, m_b)$, and gives C^* to the adversary.

Phase 2: The adversary is able to issue queries in the same way as in Phase 1, except that C^* and its attested ciphertexts cannot be submitted for decryption.

Guess: At the end of Phase 2, the adversary outputs a guess b', and succeeds in the security game if $b' = b$.

Let

$$\mathsf{Adv}^{\text{pd-ind-cca1}}_{\Gamma,\mathcal{A}_1} = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

Γ is said to offer indistinguishability under adaptive chosen ciphertext attack (PD-IND-CCA1) for ciphertext against Type-1 adversary if, for all PPT adversary $\mathcal{A}_1$, there exists a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\text{pd-ind-cca1}}_{\Gamma,\mathcal{A}_1} \leq \varepsilon(\cdot)$.

Definition 3 (PD-IND-CCA2 security against Type-2 adversary)**.** Let Γ be a PRE-DET scheme. Suppose $\mathcal{A}_2$ is a PPT adversary who interacts with a challenger $\mathcal{C}$ to perform the following security game.

Set-up: Same as in Definition 2.

Phase 1: The adversary is able to adaptively issue the following queries.

- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Same as in Definition 2.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Same as in Definition 2.
- Re-encryption key generation query $\mathcal{O}_{rkgen}$: For a queried pair $(\mathsf{pk}_i, \mathsf{pk}_j)$, where both pk_i and pk_j have been generated by the KeyGen procedure, the challenger returns $\mathsf{rk}_{i\leftrightarrow j} \leftarrow \mathtt{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j)$.

  Remark 1. As noted in [4], Section 2.1 for a bidirectional proxy re-encryption scheme, $\mathcal{O}_{rkgen}$ requires that either both $\mathcal{U}_i$ and $\mathcal{U}_j$ are corrupted users, or both are uncorrupted.

- Re-encryption query 1 $\mathcal{O}_{renc1}$: For a query $((C_i, \mathsf{pk}_i), \mathsf{pk}_j)$, where both pk_i and pk_j have been generated by the KeyGen procedure and C_i is a ciphertext under pk_i, the challenger returns a re-encrypted ciphertext $C_j = \mathtt{ReEnc}(\mathtt{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j), C_i)$.
- Attestation query $\mathcal{O}_{att}$: Same as in Definition 2.
- Re-encryption query 2 $\mathcal{O}_{renc2}$: For a query $((A_j, \mathsf{pk}_i, \mathsf{pk}_j), \mathsf{pk}_k)$, where all $\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{pk}_k$ have been generated by the KeyGen procedure, and A_j is an attested ciphertext under pk_j such that the attestation was originally added by user $\mathcal{U}_i$, the challenger returns a re-encrypted attested ciphertext $A_k = \mathtt{AttReEnc}(\mathtt{ReKeyGen}(\mathsf{sk}_j, \mathsf{sk}_k), \mathsf{pk}_j, \mathsf{pk}_i, A_j)$.
- Decryption query 1 $\mathcal{O}_{dec1}$: Same as in Definition 2.
- Decryption query 2 $\mathcal{O}_{dec2}$: Same as in Definition 2.

Challenge: At the end of Phase 1, the adversary outputs two messages $m_0, m_1 \in_R \mathcal{M}$ and a challenge public key pk^*, where pk^* is the public key of an uncorrupted user $\mathcal{U}^*$. The challenger chooses a random value $b \in {}_R\{0, 1\}$, computes $C^* \leftarrow \mathtt{DataEnc}(\mathsf{pk}^*, m_b)$, and gives C^* to the adversary.

Phase 2: The adversary is able to issue queries in the same way as in Phase 1, except that:

- Re-encryption query 1 $\mathcal{O}_{renc1}$: For a query $((C_i, \mathsf{pk}_i), \mathsf{pk}_j)$, where both pk_i and pk_j have been generated by the KeyGen procedure and C_i is a ciphertext under pk_i, if pk_j is the public key of a corrupted user $\mathcal{U}_j$ and (C_i, pk_i) is a derivative of (C^*, pk^*), then the challenger returns $\perp$; otherwise, the challenger returns a re-encrypted ciphertext $C_j = \mathtt{ReEnc}(\mathtt{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j), C_i)$.
  The definition of derivative of (C^*, pk^*) will be defined below this definition.
- Re-encryption query 2 $\mathcal{O}_{renc2}$: For a query $((A_j, \mathsf{pk}_i, \mathsf{pk}_j), \mathsf{pk}_k)$, where all $\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{pk}_k$ have been generated by the KeyGen procedure, and A_j is an attested ciphertext under pk_j such that the attestation was originally added by user $\mathcal{U}_i$, if pk_k is the public key of a corrupted user $\mathcal{U}_k$ and $(A_j, \mathsf{pk}_i, \mathsf{pk}_j)$ is a derivative of (C^*, pk^*), then the challenger returns $\perp$; otherwise, the challenger returns a re-encrypted attested ciphertext $A_k = \mathtt{AttReEnc}(\mathtt{ReKeyGen}(\mathsf{sk}_j, \mathsf{sk}_k), \mathsf{pk}_j, \mathsf{pk}_i, A_j)$.
- Decryption query 1 $\mathcal{O}_{dec1}$: Every derivative of (C^*, pk^*) cannot be submitted for decryption.
- Decryption query 2 $\mathcal{O}_{dec2}$: Every derivative of (C^*, pk^*) cannot be submitted for decryption.

Guess: At the end of Phase 2, the adversary outputs a guess b', and succeeds in the security game if $b' = b$.

Let

$$\mathsf{Adv}^{\text{pd-ind-cca2}}_{\Gamma,\mathcal{A}_2} = \left|\Pr[b' = b] - \frac{1}{2}\right|$$

Γ is said to offer indistinguishability under adaptive chosen ciphertext attack (PD-IND-CCA2) for ciphertext against Type-2 adversary if, for all PPT adversary $\mathcal{A}_2$, there exists a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\text{pd-ind-cca2}}_{\Gamma,\mathcal{A}_2} \le \varepsilon(\cdot)$.

Derivative of (C^*, pk^*) is defined as follows. For simplicity, let $\Phi \trianglelefteq \Psi$ denote that Φ is a derivative of Ψ.

- Reflexivity: $(C^*, \mathsf{pk}^*) \trianglelefteq (C^*, \mathsf{pk}^*)$.
- Transitivity: If $(C', \mathsf{pk}') \trianglelefteq (C^*, \mathsf{pk}^*)$ and $(C'', \mathsf{pk}'') \trianglelefteq (C', \mathsf{pk}')$, then $(C'', \mathsf{pk}'') \trianglelefteq (C^*, \mathsf{pk}^*)$.
- Re-encryption produces a derivative: If $C' \leftarrow \mathcal{O}_{renc}((C, \mathsf{pk}), \mathsf{pk}')$, then $(C', \mathsf{pk}') \trianglelefteq (C, \mathsf{pk})$.
- Attestation produces a derivative: If $(C', \mathsf{pk}') \trianglelefteq (C^*, \mathsf{pk}^*)$, then $(A, \mathsf{pk}') \trianglelefteq (C^*, \mathsf{pk}^*)$ where $A \leftarrow \mathtt{Attest}(\mathsf{sk}', \mathsf{pk}', C')$.
- Data sharing produces a derivative: Given $\mathsf{rk}_{i\leftrightarrow(i+1)} \leftarrow \mathcal{O}_{rkgen}(\mathsf{pk}_i, \mathsf{pk}_{i+1})$, $\mathsf{rk}_{(i+1)\leftrightarrow(i+2)} \leftarrow \mathcal{O}_{rkgen}(\mathsf{pk}_{i+1}, \mathsf{pk}_{i+2})$, $\cdots$, $\mathsf{rk}_{(i+\pi)\leftrightarrow i^*} \leftarrow \mathcal{O}_{rkgen}(\mathsf{pk}_{i+\pi}, \mathsf{pk}^*)$, where π denotes some non-negative integer, if $\mathcal{O}_{dec1}(C, \mathsf{pk}_i) \in \{m_0, m_1\}$, then $(C, \mathsf{pk}_i) \trianglelefteq (C^*, \mathsf{pk}^*)$; or if $\mathcal{O}_{dec2}((A, \mathsf{pk}_i), \mathsf{pk}_j) \in \{m_0, m_1\}$ where $j \in [i, i+\pi]$, then $(A, \mathsf{pk}_i, \mathsf{pk}_j) \trianglelefteq (C^*, \mathsf{pk}^*)$.

When the server has the equality test token, it is able to compare the ciphertexts owned by the users in the same clique, which implies the ciphertexts in this phase are distinguishable and the PRE-DET system cannot offer indistinguishability for encrypted user data under chosen plaintext/ciphertext attacks. In [40], Yang et al. have noticed that indistinguishability-based security notions are not applicable to the public key encryption schemes with equality test on ciphertexts.

Definition 4 (PD-OW-CCA3 security against Type-3 adversary)**.** Let Γ be a PRE-DET scheme. Suppose $\mathcal{A}_3$ is a PPT adversary who interacts with a challenger $\mathcal{C}$ to perform the following security game.

Set-up: Same as in Definition 2.

Phase 1: The adversary is able to adaptively issue the following queries.

- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Same as in Definition 2.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Same as in Definition 2.
- Delegation generation query $\mathcal{O}_{delgen}$: For a queried pk_i of some uncorrupted user $\mathcal{U}_i$, where pk_i has been generated by the KeyGen procedure, the challenger returns $\texttt{Delegate}(\mathsf{sk}_i)$.
- Attestation query $\mathcal{O}_{att}$: Same as in Definition 2.
- Decryption query 1 $\mathcal{O}_{dec1}$: Same as in Definition 2.
- Decryption query 2 $\mathcal{O}_{dec2}$: Same as in Definition 2.

Challenge: At the end of Phase 1, the adversary outputs a challenge public key pk^* of an uncorrupted user $\mathcal{U}^*$. The challenger picks a message $m^* \in_R \mathcal{M}$, computes $C^* \leftarrow \texttt{DataEnc}(\mathsf{pk}^*, m^*)$, and gives C^* to adversary $\mathcal{A}_3$.

Phase 2: The adversary is able to issue queries in the same way as in Phase 1, except that C^* and its attested ciphertext cannot be submitted for decryption.

Guess: At the end of Phase 2, the adversary outputs a guess m', and succeeds in the security game if $m' = m^*$.

Let

$$\mathsf{Adv}^{\text{pd-owcca3}}_{\Gamma,\mathcal{A}_3} = \Pr\left[m' = m^*\right]$$

Γ is said to offer one-wayness under adaptive chosen ciphertext attack (PD-OW-CCA3) for ciphertext against Type-3 adversary if, for all PPT adversary $\mathcal{A}_3$, there exists a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\text{pd-owcca3}}_{\Gamma,\mathcal{A}_3} \le \varepsilon(\cdot)$.

Definition 5 (PD-OW-CCA4 security against Type-4 adversary)**.** Let Γ be a PRE-DET scheme. Suppose $\mathcal{A}_4$ is a PPT adversary who interacts with a challenger $\mathcal{C}$ to perform the following security game.

Set-up: Same as in Definition 3.

Phase 1: The adversary is able to adaptively issue the following queries.

- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Same as in Definition 3.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Same as in Definition 3.
- Re-encryption key generation query $\mathcal{O}_{rkgen}$: Same as in Definition 3.
- Re-encryption query 1 $\mathcal{O}_{renc1}$: Same as in Definition 3.
- Attestation query $\mathcal{O}_{att}$: Same as in Definition 3.
- Re-encryption query 2 $\mathcal{O}_{renc2}$: Same as in Definition 3.
- Delegation generation query $\mathcal{O}_{delgen}$: For a queried pk_i of some uncorrupted user $\mathcal{U}_i$, where pk_i has been generated by the KeyGen procedure, the challenger returns $\texttt{Delegate}(\mathsf{sk}_i)$.
- Decryption query 1 $\mathcal{O}_{dec1}$: Same as in Definition 3.
- Decryption query 2 $\mathcal{O}_{dec2}$: Same as in Definition 3.

Challenge: At the end of Phase 1, the adversary outputs a challenge public key pk^* of an uncorrupted user $\mathcal{U}^*$. The challenger picks a message $m^* \in_R \mathcal{M}$, computes $C^* \leftarrow \texttt{DataEnc}(\mathsf{pk}^*, m^*)$, and gives C^* to adversary $\mathcal{A}_4$.

Phase 2: The adversary is able to issue queries in the same way as in Phase 1, except that:

- Re-encryption query 1 $\mathcal{O}_{renc1}$: Same as in Definition 3.
- Re-encryption query 2 $\mathcal{O}_{renc2}$: Same as in Definition 3.
- Decryption query 1 $\mathcal{O}_{dec1}$: Same as in Definition 3.
- Decryption query 2 $\mathcal{O}_{dec2}$: Same as in Definition 3.

Guess: At the end of Phase 2, the adversary outputs a guess m', and succeeds in the security game if $m' = m^*$.

Let

$$\mathsf{Adv}^{\text{pd-owcca4}}_{\Gamma,\mathcal{A}_4} = \Pr\left[m' = m^*\right]$$

Γ is said to offer one-wayness under adaptive chosen ciphertext attack (PD-OW-CCA4) for ciphertext against Type-4 adversary if, for all PPT adversary $\mathcal{A}_4$, there exists a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\text{pd-owcca4}}_{\Gamma,\mathcal{A}_4} \le \varepsilon(\cdot)$.

Definition 6 (PD-EUCMA security against Type-5 adversary)**.** Let Γ be a bidirectional PRE-DET scheme. Suppose $\mathcal{A}_5$ is a PPT adversary who interacts with a challenger $\mathcal{C}$ to perform the following security game.

Set-up: With a security parameter λ, the challenger runs the Setup procedure to produce public parameter par, which is given to the adversary. The challenger generates a pair of challenge key pair $(\mathsf{pk}^*, \mathsf{sk}^*)$.

Queries: The adversary is able to adaptively issue the queries as defined in Phase 1 of Definition 5.

Output: Eventually, the adversary outputs a tuple $(C^*, A^*, \mathsf{pk}_i^*)$. Adversary $\mathcal{A}_5$ wins the game if both the following conditions are satisfied:

1. (C^*, pk^*) has not been submitted in attestation queries;
2. $(A^*, \mathsf{pk}^*, \mathsf{pk}_i^*) \trianglelefteq (C^*, \mathsf{pk}^*)$, which implies $\texttt{AttDec}(\mathsf{pk}^*, \mathsf{pk}_i^*, \mathsf{sk}_i^*, A^*) = \texttt{DataDec}(\mathsf{sk}^*, C^*) \neq \perp$ and $\texttt{AttVer}(\mathsf{pk}^*, A^*) = 1$.

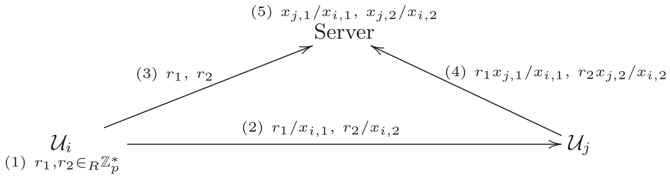

Fig. 1. Procedure of ReKeyGen.

Let

$$\mathsf{Adv}^{\mathsf{pd\text{-}eucma}}_{\Gamma,\mathcal{A}_5} = \Pr[\mathcal{A}_5 \text{ wins}]$$

Γ is said to offer existential unforgeability under adaptively chosen message attack (PD-EUCMA) for attested ciphertext against Type-5 adversary if, for all PPT adversary $\mathcal{A}_5$, there exists a negligible function $\varepsilon(\cdot)$ such that $\mathsf{Adv}^{\mathsf{pd\text{-}eucma}}_{\Gamma,\mathcal{A}_5} \le \varepsilon(\cdot)$.

3. A PRE-DET construction

Suppose $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are cyclic groups with prime order p and efficient group operations. The mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is bilinear if the following properties are satisfied:

- Bilinearity: $\forall \mu, \nu \in \mathbb{G}$, and $\forall x, y \in \mathbb{Z}_p^*$, $\hat{e}(\mu^x, \nu^y) = \hat{e}(\mu, \nu)^{xy}$;
- Non-degeneracy: $\hat{e}(g, g) \neq 1$;
- Efficiency: $\hat{e}$ is efficiently computable.

The security of our construction relies on the following complexity assumptions [1,36].

Computational Diffie-Hellman Assumption (CDH). Let $\mathbb{G} = \langle g \rangle$ be a cyclic group with prime order p. The CDH assumption states that given a tuple $(g, g^x, g^y) \in \mathbb{G}^3$, where $x, y \in_R \mathbb{Z}_p^*$, any PPT algorithm has negligible advantage $\varepsilon_{\mathsf{cdh}}$ in computing g^{xy}.

Divisible Computational Diffie-Hellman Assumption (DCDH). Let $\mathbb{G} = \langle g \rangle$ be a cyclic group with prime order p. The DDH assumption states that given a tuple $(g, g^{1/x}, g^y) \in \mathbb{G}^3$, where $x, y \in_R \mathbb{Z}_p^*$, any PPT algorithm has negligible advantage $\varepsilon_{\mathsf{dcdh}}$ in computing g^{xy}.

Bao, Deng and Zhu [1] proved that the DCDH and CDH assumptions are equivalent.

We now present a PRE-DET construction in bilinear groups.

$\mathtt{Setup}(1^\lambda)$: Choose a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are cyclic groups with prime order p. Let τ_G denote the element size in $\mathbb{G}$ and $\tau_p = \log p$. Let $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme with verification key space $\{0, 1\}^{q(\lambda)}$, where $q(\lambda)$ is a polynomial in λ. Pick two random elements $h, \hbar \in_R \mathbb{G}$ and seven cryptographic hash functions: $H_1 : \{0,1\}^{\tau_p+2\lambda+q(\lambda)} \to \mathbb{Z}_p$, $H_2 : \mathbb{G} \to \{0,1\}^{\tau_p+2\lambda}$, $H_3 : \{0,1\}^{\tau_p+\lambda+q(\lambda)} \to \mathbb{Z}_p$, $H_4 : \mathbb{G} \to \mathbb{G}$, $H_5 : \{0,1\}^{\tau_p+2\lambda+2\tau_G} \to \mathbb{Z}_p$, $H_6 : \{0,1\}^{\tau_p+\lambda+2\tau_G} \to \mathbb{Z}_p$, $H_7 : \{0,1\}^{2\tau_p+2\lambda+3\tau_G} \to \mathbb{G}$. The message space is $\mathcal{M} = \mathbb{Z}_p$. Let the attestation space be $\mathbb{A} = \mathbb{Z}_p$. The system public parameter is $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, h, \hbar, H_1, H_2, H_3, H_4, H_5, H_6, H_7, \mathsf{OS})$.

$\mathtt{KeyGen}(\mathsf{par})$: Randomly choose $x_i, y_i, z_i \in_R \mathbb{Z}_p^*$, set $\mathsf{sk}_{i,1} = x_i$, $\mathsf{sk}_{i,2} = y_i$, $\mathsf{sk}_{i,3} = z_i$, and compute $\mathsf{pk}_{i,1} = g^{x_i}$, $\mathsf{pk}_{i,2} = g^{y_i}$, $\mathsf{pk}_{i,3} = g^{z_i}$. The secret key and public key are $\mathsf{sk}_i = (\mathsf{sk}_{i,1}, \mathsf{sk}_{i,2}, \mathsf{sk}_{i,3})$ and $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \mathsf{pk}_{i,3})$, respectively.

$\mathtt{ReKeyGen}(\mathsf{sk}_i, \mathsf{sk}_j)$: On input the secret keys sk_i and sk_j, output the bidirectional re-encryption key $\mathsf{rk}_{i\leftrightarrow j} = (\mathsf{sk}_{j,1}/\mathsf{sk}_{i,1} \bmod p, \mathsf{sk}_{j,2}/\mathsf{sk}_{i,2} \bmod p)$.

Similar to [2,4], the ReKeyGen procedure can be run as follows (see Fig. 1): user $\mathcal{U}_i$ randomly picks r_1, r_2 from $\mathbb{Z}_p^*$, sends $r_1/x_{i,1}$, $r_2/x_{i,2}$ to user $\mathcal{U}_j$ and r_1, r_2 to the server. User $\mathcal{U}_j$ computes $r_1x_{j,1}/x_{i,1}$, $r_2x_{j,2}/x_{i,2}$ and gives them to the server to recover the re-encryption key $(x_{j,1}/x_{i,1}, x_{j,2}/x_{i,2})$.

$\mathtt{DataEnc}(\mathsf{pk}_i, m)$: For a given message $m \in \mathcal{M}$, randomly select $\alpha, \beta \in_R \{0, 1\}^\lambda$, and generate a ciphertext $C = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$ where

$$\begin{array}{ll}
(\mathsf{osk}, \mathsf{ovk}) \leftarrow \mathsf{OS.KGen}(1^\lambda), & \\
\theta = H_1(m\|\alpha\|\beta\|\mathsf{ovk}), & c_1 = H_2(g^\theta) \oplus (m\|\alpha\|\beta), \\
c_2 = \mathsf{pk}_{i,1}^\theta, & c_3 = h^\theta, \\
\vartheta = H_3(m\|\alpha\|\mathsf{ovk}), & c_4 = \mathsf{pk}_{i,2}^\vartheta, \\
c_5 = \hbar^\vartheta, & c_6 = g^m \cdot H_4(g^\vartheta), \\
c_7 = \mathsf{OS.Sign}(\mathsf{osk}, c_1\|c_3\|c_5\|c_6), & c_8 = \mathsf{ovk}.
\end{array}$$

$\mathtt{ReEnc}(\mathsf{rk}_{i\leftrightarrow j}, C_{i_l})$: For a given re-encryption key $\mathsf{rk}_{i\leftrightarrow j}$ and a ciphertext $C_{i_l} = (c_{i_l,1}, c_{i_l,2}, \cdots, c_{i_l,8})$ of user $\mathcal{U}_i$, check

$$\mathsf{OS.Vrfy}\left(c_{i_l,8}, c_{i_l,7}, c_{i_l,1}\|c_{i_l,3}\|c_{i_l,5}\|c_{i_l,6}\right) \stackrel{?}{=} 1, \tag{1}$$

$$\hat{e}\left(c_{i_l,2}, h\right) \stackrel{?}{=} \hat{e}\left(\mathsf{pk}_{i,1}, c_{i_l,3}\right), \tag{2}$$

$$\hat{e}\left(c_{i_l,4}, \hbar\right) \stackrel{?}{=} \hat{e}\left(\mathsf{pk}_{i,2}, c_{i_l,5}\right). \tag{3}$$

If some condition is not met, then output $\perp$ and halt; otherwise compute a ciphertext $C_{j_l} = (c_{j_l,1}, c_{j_l,2}, \cdots, c_{j_l,8})$ of user $\mathcal{U}_j$, where $c_{j_l,2} = (c_{i_l,2})^{\mathsf{rk}_{i\leftrightarrow j,1}}$, $c_{j_l,4} = (c_{i_l,4})^{\mathsf{rk}_{i\leftrightarrow j,2}}$ and $c_{j_l,t} = c_{i_l,t}$ for the other components.

`DataDec`(sk_i, C): Given a ciphertext $C = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$, check whether it satisfies equalities in (1), (2) and (3). If some condition is not met, then output $\perp$ and halt; otherwise compute

$$\begin{aligned} m\|\alpha\|\beta &= c_1 \oplus H_2\left((c_2)^{x_{i,1}^{-1}}\right), \\ \theta &= H_1(m\|\alpha\|\beta\|c_8), \\ \vartheta &= H_3(m\|\alpha\|c_8). \end{aligned} \tag{4}$$

Then verify

$$c_2 \stackrel{?}{=} g^{\mathsf{sk}_{i,1}\cdot\theta}, \tag{5}$$

$$c_4 \stackrel{?}{=} g^{\mathsf{sk}_{i,2}\cdot\vartheta}, \tag{6}$$

and

$$c_6 \stackrel{?}{=} g^m \cdot H_4\left(g^{\vartheta}\right). \tag{7}$$

If all conditions are met, then output m, otherwise output $\perp$.

`Attest`($\mathsf{sk}_i, \mathsf{pk}_i, C$): Let $m \leftarrow$ `DataDec`(sk_i, C). Let att be the attestation of m and $m\|\alpha\|\beta$ be the output of Formula (4) in decryption. Randomly select $\alpha', \beta' \in_R \{0,1\}^\lambda$ and compute an attested ciphertext $A = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$ as follows:

$$\begin{array}{ll} \theta' = H_5\left(m\|\alpha'\|\beta'\|\mathsf{pk}_{i,1}\|\mathsf{pk}_{i,2}\right), & a_1 = H_2\left(g^{\theta'}\right) \oplus \left(m\|\alpha'\|\beta'\right), \\ a_2 = \mathsf{pk}_{i,1}^{\theta'}, & a_3 = h^{\theta'}, \\ \vartheta' = H_6\left(m\|\alpha'\|\mathsf{pk}_{i,1}\|\mathsf{pk}_{i,2}\right), & a_4 = \mathsf{pk}_{i,2}^{\vartheta'}, \\ a_5 = \hbar^{\vartheta'}, & a_6 = g^m \cdot H_4\left(g^{\vartheta'}\right), \\ a_7 = H_7(a_1\|a_3\|a_5\|a_6\|att)^{\mathsf{sk}_{i,3}}, & a_8 = att. \end{array}$$

`AttReEnc`($\mathsf{rk}_{i\leftrightarrow j}, \mathsf{pk}_i, \mathsf{pk}_\ell, A_{i_l}$): Given re-encryption key $\mathsf{rk}_{i\leftrightarrow j}$ and an attested ciphertext $A_{i_l} = (a_{i_l,1}, a_{i_l,2}, \cdots, a_{i_l,8})$ of user $\mathcal{U}_i$, where the attestation was added by user $\mathcal{U}_\ell$, check

$$\hat{e}(a_{i_l,7}, g) \stackrel{?}{=} \hat{e}\left(H_7(a_{i_l,1}\|a_{i_l,3}\|a_{i_l,5}\|a_{i_l,6}\|a_{i_l,8}), \mathsf{pk}_{\ell,3}\right), \tag{8}$$

$$\hat{e}(a_{i_l,2}, h) \stackrel{?}{=} \hat{e}\left(\mathsf{pk}_{i,1}, a_{i_l,3}\right), \tag{9}$$

$$\hat{e}\left(a_{i_l,4}, \hbar\right) \stackrel{?}{=} \hat{e}\left(\mathsf{pk}_{i,2}, a_{i_l,5}\right). \tag{10}$$

If some condition is not met, then output $\perp$ and halt; otherwise compute an attested ciphertext $A_{j_l} = (a_{j_l,1}, a_{j_l,2}, \cdots, a_{j_l,8})$ of user $\mathcal{U}_j$, where $a_{j_l,2} = (a_{i_l,2})^{\mathsf{rk}_{i\leftrightarrow j,1}}$, $a_{j_l,4} = (a_{i_l,4})^{\mathsf{rk}_{i\leftrightarrow j,2}}$ and $a_{j_l,t} = a_{i_l,t}$ for the other components.

`AttVer`(pk_i, A): Note that $a_8 \in A$ is the attestation. Run the verification procedure in the same way as in Formula (8):

$$\hat{e}(a_7, g) \stackrel{?}{=} \hat{e}\left(H_7(a_1\|a_3\|a_5\|a_6\|a_8), \mathsf{pk}_{i,3}\right) \tag{11}$$

`AttDec`($\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, A$): Given an attested ciphertext $A = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$, check whether it satisfies the equalities in (11), (9) and (10) under public key pk_j. If some condition is not met, then output $\perp$ and halt; otherwise compute

$$\begin{aligned} m\|\alpha'\|\beta' &= a_1 \oplus H_2\left((a_2)^{x_{j,1}^{-1}}\right), \\ \theta' &= H_5\left(m\|\alpha'\|\beta'\|\mathsf{pk}_{i,1}\|\mathsf{pk}_{i,2}\right), \\ \vartheta' &= H_6\left(m\|\alpha'\|\mathsf{pk}_{i,1}\|\mathsf{pk}_{i,2}\right). \end{aligned} \tag{12}$$

Then verify

$$a_2 \stackrel{?}{=} g^{\mathsf{sk}_{j,1}\cdot\theta'}, \tag{13}$$

$$a_4 \stackrel{?}{=} g^{\mathsf{sk}_{j,2}\cdot\vartheta'}, \tag{14}$$

and

$$a_6 \stackrel{?}{=} g^m \cdot H_4\left(g^{\vartheta'}\right). \tag{15}$$

`Delegate`(sk_i): Output the token $\mathsf{tk}_i = \mathsf{sk}_{i,2}$.

$$
\begin{tikzcd}[column sep=small]
 & \texttt{KeyGen} \arrow[d, Rightarrow, "\mathsf{pk}"] & \texttt{ReKeyGen} \arrow[d, Rightarrow, "\mathsf{rk}"] & \texttt{ReKeyGen} \arrow[d, Rightarrow, "\mathsf{rk}"] & \texttt{KeyGen} \arrow[d, Rightarrow, "\mathsf{pk}"] & \texttt{ReKeyGen} \arrow[d, Rightarrow, "\mathsf{rk}"] & \texttt{ReKeyGen} \arrow[d, Rightarrow, "\mathsf{rk}"] \\
 \arrow[r, Rightarrow, "m"] & \texttt{DataEnc} \arrow[r, Rightarrow, "C"] & \texttt{ReEnc} \arrow[r, Rightarrow, "C_1"] & \cdots\texttt{ReEnc} \arrow[r, Rightarrow, "C_i"] & \texttt{Attest} \arrow[r, Rightarrow, "A_i"] & \texttt{AttReEnc} \arrow[r, Rightarrow, "A_{i+1}"] & \texttt{AttReEnc}\cdots
\end{tikzcd}
$$

Fig. 2. Usage of the re-encryption and attestation procedures.

Remark 2. Combining `Delegate` with `ReKeyGen` achieves a more powerful delegation clique. Suppose there is a chain of re-encryption keys $\mathsf{rk}_{(i-\tau_1)\leftrightarrow(i-\tau_1+1)}, \cdots, \mathsf{rk}_{(i-1)\leftrightarrow i}, \mathsf{rk}_{i\leftrightarrow(i+1)}, \cdots, \mathsf{rk}_{(i+\tau_2-1)\leftrightarrow(i+\tau_2)}$ for some positive τ_1 and τ_2. Once given tk_i, the server also gets $\mathsf{tk}_{i-\tau_1}, \cdots, \mathsf{tk}_{i-1}, \mathsf{tk}_{i+1}, \cdots, \mathsf{tk}_{i+\tau_2}$. In this way, the server is allowed to compare ciphertexts of a clique of users $\mathcal{U}_{i-\tau_1}, \cdots, \mathcal{U}_{i-1}, \mathcal{U}_i, \mathcal{U}_{i+1}, \cdots, \mathcal{U}_{i+\tau_2}$.

$\texttt{EqTest}(\mathsf{tk}_i, C_{i_l}/A_{i_l}, \mathsf{tk}_j, C_{j_h}/A_{j_h})$: For two (re-encrypted) ciphertexts C_{i_l} and C_{j_h} under the public keys pk_i and pk_j, respectively, check whether they encrypt the same message (i.e., $m_{i_l} = m_{j_h}$) as follows:

$$\frac{c_{i_l,6}}{H_4\left(\left(c_{i_l,4}\right)^{\mathsf{tk}_i^{-1}}\right)} \stackrel{?}{=} \frac{c_{j_h,6}}{H_4\left(\left(c_{j_h,4}\right)^{\mathsf{tk}_j^{-1}}\right)} \tag{16}$$

The equality test for (re-encrypted) attested ciphertext pair (A_{i_l}, A_{j_h}) can be performed similarly.

Remark 3. The EqTest procedure in our construction also supports equality test on (re-encrypted) ciphertexts and (re-encrypted) attested ciphertexts.

Fig. 2 depicts how the procedures `ReEnc`, `Attest` and `AttReEnc` are invoked in achieving a secure data sharing clique for a message m.

Soundness. For validating the soundness of the proposed scheme, the straightforward steps can be omitted. For decryption of a ciphertext, we need only to show the following equality holds:

$$c_1 \oplus H_2\left((c_2)^{x_{i,1}^{-1}}\right) = \left(H_2\left(g^{\theta}\right) \oplus (m\|\alpha\|\beta)\right) \oplus H_2\left(\left(\mathsf{pk}_{i,1}^{\theta}\right)^{x_{i,1}^{-1}}\right) = m\|\alpha\|\beta$$

For equality test, we have

$$\frac{c_{i_l,6}}{H_4\left(\left(c_{i_l,4}\right)^{\mathsf{tk}_i^{-1}}\right)} = \frac{g^{m_{i_l}} \cdot H_4\left(g^{\vartheta_{i_l}}\right)}{H_4\left(\left(\mathsf{pk}_{i,2}^{\vartheta_{i_l}}\right)^{x_{i,2}^{-1}}\right)} = \frac{g^{m_{i_l}} \cdot H_4\left(g^{\vartheta_{i_l}}\right)}{H_4\left(g^{\vartheta_{i_l}}\right)} = g^{m_{i_l}}$$

and similarly

$$\frac{c_{j_h,6}}{H_4\left(\left(c_{j_h,4}\right)^{\mathsf{tk}_j^{-1}}\right)} = \frac{g^{m_{j_h}} \cdot H_4\left(g^{\vartheta_{j_h}}\right)}{H_4\left(\left(\mathsf{pk}_{j,2}^{\vartheta_{j_h}}\right)^{x_{j,2}^{-1}}\right)} = \frac{g^{m_{j_h}} \cdot H_4\left(g^{\vartheta_{j_h}}\right)}{H_4\left(g^{\vartheta_{j_h}}\right)} = g^{m_{j_h}}$$

Thus, $m_{i_l} = m_{j_h}$ if and only if Equality (16) holds.

4. Security analysis

Theorem 1. *Suppose the DCDH assumption holds in group $\mathbb{G}$. The proposed PRE-DET scheme offers PD-IND-CCA1 security for ciphertext against Type-1 adversary in the random oracle model.*

The proof can be captured as a special case of Theorem 2 without two types of re-encryption queries, which is thus omitted here. Specifically, if there is a Type-1 PPT adversary $\mathcal{A}_1$ that has non-negligible advantage ε in attacking the PD-IND-CCA1 security for ciphertext in the PRE-DET scheme, then one can construct an algorithm $\mathcal{I}$ to solve the DCDH problem with non-negligible probability $\varepsilon_{\mathsf{dcdh}}$ such that

$$\varepsilon_{\mathsf{dcdh}} \geq \frac{1}{q_{H_2}}\left(2\varepsilon - \frac{q_{H_1}+q_{H_5}}{4^{\lambda}} - \frac{q_{H_3}+q_{H_6}}{2^{\lambda}} - \frac{q_{H_4}+3q_D}{p} - \frac{q_D}{p\cdot 4^{\lambda}} - (q_{H_1}+q_{H_3})\cdot\rho - \Pr[\mathcal{A}_1 \textit{ breaks } \mathsf{OS}]\right)$$

where $\mathcal{A}_1$ is able to issue at most q_D decryption queries to $\mathcal{O}_{dec1}$, and at most q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} and q_{H_6} hash queries to H_1, H_2, H_3, H_4, H_5 and H_6, respectively. $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a strong one-time signature scheme, where KGen has super-logarithmic minimum entropy and maximum probability ρ of outputting a given verification key.

Theorem 2. *Suppose the DCDH assumption holds in group $\mathbb{G}$. The proposed PRE-DET scheme offers PD-IND-CCA2 security for ciphertext against Type-2 adversary in the random oracle model.*

The following proof follows the standard framework established in [4,36].

Proof. Let $\mathcal{A}_2$ be a Type-2 PPT adversary that has non-negligible advantage ε in attacking the PD-IND-CCA2 security for ciphertext in the PRE-DET scheme. Suppose $\mathcal{A}_2$ issues at most q_D decryption queries to $\mathcal{O}_{dec1}$, at most q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} and q_{H_6} hash queries to H_1, H_2, H_3, H_4, H_5 and H_6, respectively, at most q_{R_1} re-encryption queries to $\mathcal{O}_{renc1}$, and at most q_{R_2} re-encryption queries to $\mathcal{O}_{renc2}$. Let $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme, where KGen has super-logarithmic minimum entropy and maximum probability ρ of outputting a given verification key. We show that if such an adversary $\mathcal{A}_2$ exists, then one can construct an algorithm $\mathcal{I}$ to solve the DCDH problem with non-negligible probability $\varepsilon_{\mathsf{dcdh}}$.

Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be cycle groups with prime order p and bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. At first, algorithm $\mathcal{I}$ is given a DCDH instance $(g, g^{1/u}, g^v) \in \mathbb{G}^3$. The goal of $\mathcal{I}$ is to compute g^{uv}. Algorithm $\mathcal{I}$ simulates the challenger and interacts with adversary $\mathcal{A}_2$ as follows.

Set-up: Algorithm $\mathcal{I}$ randomly picks $\tilde{a}, \tilde{b} \in_R \mathbb{Z}_p^*$, computes $h = g^{\tilde{a}/u}$ and $\hbar = g^{\tilde{b}}$, and sets the system public parameter as $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, h, \hbar, H_1, H_2, \cdots, H_7, \mathsf{OS})$.

Algorithm $\mathcal{I}$ runs $(\mathsf{osk}^*, \mathsf{ovk}^*) \leftarrow \mathsf{OS.KGen}(1^\lambda)$ and records the key pair.

Phase 1: The adversary adaptively makes the following queries.

- H_1 hash query $\mathcal{O}_{H_1}$: For answering $\mathcal{O}_{H_1}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_1$ which is initially empty. For an input tuple $(m, \alpha, \beta, \mathsf{ovk})$, if there exists an entry $(m, \alpha, \beta, \mathsf{ovk}, \theta) \in \mathcal{L}_1$, then $\mathcal{O}_{H_1}$ responds with θ; otherwise, a random value $\theta \in_R \mathbb{Z}_p^*$ is picked and returned, and $\mathcal{L}_1$ is updated as $\mathcal{L}_1 \cup (m, \alpha, \beta, \mathsf{ovk}, \theta)$.
- H_2 hash query $\mathcal{O}_{H_2}$: For answering $\mathcal{O}_{H_2}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_2$ which is initially empty. For an input element T, if there exists an entry $(T, \Theta) \in \mathcal{L}_2$, then $\mathcal{O}_{H_2}$ responds with Θ; otherwise, a random value $\Theta \in_R \{0,1\}^{\tau_p + 2\lambda}$ is picked and returned, and $\mathcal{L}_2$ is updated as $\mathcal{L}_2 \cup (T, \Theta)$.
- H_3 hash query $\mathcal{O}_{H_3}$: For answering $\mathcal{O}_{H_3}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_3$ which is initially empty. For an input tuple $(m, \alpha, \mathsf{ovk})$, if there exists an entry $(m, \alpha, \mathsf{ovk}, \vartheta) \in \mathcal{L}_3$, then $\mathcal{O}_{H_3}$ responds with ϑ; otherwise, a random value $\vartheta \in_R \mathbb{Z}_p^*$ is picked and returned, and $\mathcal{L}_3$ is updated as $\mathcal{L}_3 \cup (m, \alpha, \mathsf{ovk}, \vartheta)$.
- H_4 hash query $\mathcal{O}_{H_4}$: For answering $\mathcal{O}_{H_4}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_4$ which is initially empty. For an input element $T \in \mathbb{G}$, if there exists an entry $(T, \Delta) \in \mathcal{L}_4$, then $\mathcal{O}_{H_4}$ responds with Δ; otherwise, a random value $\Delta \in_R \mathbb{G}$ is picked and returned, and $\mathcal{L}_4$ is updated as $\mathcal{L}_4 \cup (T, \Delta)$.
- H_5 hash query $\mathcal{O}_{H_5}$: For answering $\mathcal{O}_{H_5}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_5$ which is initially empty. For an input tuple $(m, \alpha, \beta, \mathsf{pk}_1, \mathsf{pk}_2)$, if there exists an entry $(m, \alpha, \beta, \mathsf{pk}_1, \mathsf{pk}_2, \theta') \in \mathcal{L}_5$, then $\mathcal{O}_{H_5}$ responds with θ'; otherwise, a random value $\theta' \in_R \mathbb{Z}_p^*$ is picked and returned, and $\mathcal{L}_5$ is updated as $\mathcal{L}_5 \cup (m, \alpha, \beta, \mathsf{pk}_1, \mathsf{pk}_2, \theta')$.
- H_6 hash query $\mathcal{O}_{H_6}$: For answering $\mathcal{O}_{H_6}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_6$ which is initially empty. For an input tuple $(m, \alpha, \mathsf{pk}_1, \mathsf{pk}_2)$, if there exists an entry $(m, \alpha, \mathsf{pk}_1, \mathsf{pk}_2, \vartheta') \in \mathcal{L}_6$, then $\mathcal{O}_{H_6}$ responds with ϑ'; otherwise, a random value $\vartheta' \in_R \mathbb{Z}_p^*$ is picked and returned, and $\mathcal{L}_6$ is updated as $\mathcal{L}_6 \cup (m, \alpha, \mathsf{pk}_1, \mathsf{pk}_2, \vartheta')$.
- H_7 hash query $\mathcal{O}_{H_7}$: For answering $\mathcal{O}_{H_7}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_7$ which is initially empty. For an input tuple $(a_1, a_3, a_5, a_6, att)$, if there exists an entry $(a_1, a_3, a_5, a_6, att, \Lambda) \in \mathcal{L}_7$, then $\mathcal{O}_{H_7}$ responds with Λ; otherwise, a random value $\Lambda \in_R \mathbb{G}$ is picked and returned, and $\mathcal{L}_7$ is updated as $\mathcal{L}_7 \cup (a_1, a_3, a_5, a_6, att, \Lambda)$.
- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Algorithm $\mathcal{I}$ randomly picks $x_{i,1}, x_{i,2}, x_{i,3} \in_R \mathbb{Z}_p^*$, sets $\mathsf{sk}_{i,2} = x_{i,2}$, $\mathsf{sk}_{i,3} = x_{i,3}$, and computes $\mathsf{pk}_{i,1} = (g^{1/u})^{x_{i,1}} = g^{x_{i,1}/u}$, $\mathsf{pk}_{i,2} = g^{x_{i,2}}$ and $\mathsf{pk}_{i,3} = g^{x_{i,3}}$. Next, algorithm $\mathcal{I}$ gives $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \mathsf{pk}_{i,3})$ to $\mathcal{A}_2$ and adds $(i, x_{i,1}, \mathsf{pk}_{i,1}, x_{i,2}, \mathsf{pk}_{i,2}, x_{i,3}, \mathsf{pk}_{i,3}, 0)$ to the list $\mathcal{L}_{key}$, where '0' denotes that pk_i is an uncorrupted public key.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Algorithm $\mathcal{I}$ randomly picks $x_{j,1}, x_{j,2}, x_{j,3} \in_R \mathbb{Z}_p^*$, sets $\mathsf{sk}_{j,1} = x_{j,1}$, $\mathsf{sk}_{j,2} = x_{j,2}$ and $\mathsf{sk}_{j,3} = x_{j,3}$, and computes $\mathsf{pk}_{j,1} = g^{x_{j,1}}$, $\mathsf{pk}_{j,2} = g^{x_{j,2}}$ and $\mathsf{pk}_{j,3} = g^{x_{j,3}}$. Next, algorithm $\mathcal{I}$ gives $(\mathsf{sk}_j, \mathsf{pk}_j)$ to $\mathcal{A}_2$ and adds $(j, \mathsf{sk}_{j,1}, \mathsf{pk}_{j,1}, \mathsf{sk}_{j,2}, \mathsf{pk}_{j,2}, \mathsf{sk}_{j,3}, \mathsf{pk}_{j,3}, 1)$ to the list $\mathcal{L}_{key}$, where '1' denotes that pk_j is a corrupted public key.
- Re-encryption key generation query $\mathcal{O}_{rkgen}$: For a queried pair $(\mathsf{pk}_i, \mathsf{pk}_j)$, if one of $\mathcal{U}_i$ and $\mathcal{U}_j$ is corrupted while the other is uncorrupted, then returns $\perp$. Otherwise, algorithm $\mathcal{I}$ outputs $\mathsf{rk}_{i\leftrightarrow j} = (x_{j,1}/x_{i,1} \bmod p, x_{j,2}/x_{i,2} \bmod p)$.
- Re-encryption query $\mathcal{O}_{renc1}$: For a query $((C_i, \mathsf{pk}_i), \mathsf{pk}_j)$, algorithm $\mathcal{I}$ checks whether $C_i = (c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8)$ satisfies the equalities in (1), (2) and (3). If some condition is not met, which means the ciphertext is not well-formed, then algorithm $\mathcal{I}$ returns $\perp$ and halts. Otherwise, algorithm $\mathcal{I}$ works as follows:
  - If both users $\mathcal{U}_i$ and $\mathcal{U}_j$ are either corrupted or uncorrupted, then algorithm $\mathcal{I}$ computes $\mathsf{rk}_{i\leftrightarrow j} = (x_{j,1}/x_{i,1} \bmod p, x_{j,2}/x_{i,2} \bmod p)$ and returns $\mathtt{ReEnc}(\mathsf{rk}_{i\leftrightarrow j}, C_i)$.
  - If one of $\mathcal{U}_i$ and $\mathcal{U}_j$ is corrupted and the other is uncorrupted, then algorithm $\mathcal{I}$ searches $\mathcal{L}_1$ for a tuple $(m, \alpha, \beta, \mathsf{ovk}, \theta)$ such that $\mathsf{pk}_{i,1}^{\theta} = c_2$ and $h^\theta = c_3$. If no such tuple can be found, algorithm $\mathcal{I}$ returns $\perp$; otherwise, it retrieves $(m, \alpha, \mathsf{ovk}, \vartheta)$ from $\mathcal{L}_3$. If $\mathsf{pk}_{i,2}^{\vartheta} = c_4$ and $\hbar^\vartheta = c_5$, then it computes $c_2' = \mathsf{pk}_{j,1}^{\theta}$ and $c_4' = \mathsf{pk}_{j,2}^{\vartheta}$, and returns $(c_1, c_2', c_3, c_4', c_5, c_6, c_7, c_8)$; otherwise it returns $\perp$.
- Attestation query $\mathcal{O}_{att}$: For a query (C_i, pk_i), algorithm $\mathcal{I}$ performs the oracle $\mathcal{O}_{dec1}$ with input (C_i, pk_i). If $\mathcal{O}_{dec1}$ outputs $\perp$, then algorithm $\mathcal{I}$ returns $\perp$ and halts. Otherwise, letting $(m, \alpha, \beta, c_8, \theta) \in \mathcal{L}_1$ be the retrieved tuple in $\mathcal{O}_{dec1}$, algorithm $\mathcal{I}$ runs the oracle $\mathcal{O}_{H_5}$ with input $(m, \alpha, \beta, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ to get θ', runs $\mathcal{O}_{H_2}$ with input $g^{\theta'}$ to get Θ', runs $\mathcal{O}_{H_6}$ with input $(m, \alpha, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ to get ϑ', and runs $\mathcal{O}_{H_4}$ with input $g^{\vartheta'}$ to get Δ'. Then algorithm $\mathcal{I}$ computes

$a_1 = \Theta' \oplus (m\|\alpha\|\beta)$, $a_2 = \mathsf{pk}_{i,1}^{\theta'}$, $a_3 = h^{\theta'}$, $a_4 = \mathsf{pk}_{i,2}^{\vartheta'}$, $a_5 = \hbar^{\vartheta'}$ and $a_6 = g^m \cdot \Delta'$, chooses $a_8 = att \in \mathbb{Z}_p$, runs $\mathcal{O}_{H_7}$ with input $(a_1, a_3, a_5, a_6, a_8)$ to get Λ, computes $a_7 = \Lambda^{\mathsf{sk}_{i,3}}$, and returns $A_i = (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$.

- Re-encryption query $\mathcal{O}_{renc2}$: For a query $((A_j, \mathsf{pk}_i, \mathsf{pk}_j), \mathsf{pk}_k)$, algorithm $\mathcal{I}$ checks whether $(A_j, \mathsf{pk}_i, \mathsf{pk}_j)$ satisfies the equalities in (8), (9) and (10). If some condition is not met, which means the attested ciphertext is not well-formed, then algorithm $\mathcal{I}$ returns $\perp$ and halts. Otherwise, algorithm $\mathcal{I}$ works as follows:
  - If both users $\mathcal{U}_j$ and $\mathcal{U}_k$ are either corrupted or uncorrupted, then algorithm $\mathcal{I}$ computes $\mathsf{rk}_{j\leftrightarrow j} = (x_{k,1}/x_{j,1} \bmod p, x_{k,2}/x_{j,2} \bmod p)$ and returns $\mathtt{AttReEnc}(\mathsf{rk}_{j\leftrightarrow k}, A_j)$.
  - If one of $\mathcal{U}_j$ and $\mathcal{U}_k$ is corrupted and the other is uncorrupted, then algorithm $\mathcal{I}$ searches $\mathcal{L}_5$ for a tuple $(m, \alpha, \beta, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \theta')$ such that $\mathsf{pk}_{j,1}^{\theta'} = a_2$ and $h^{\theta'} = a_3$. If no such tuple can be found, then algorithm $\mathcal{I}$ returns $\perp$; otherwise, it retrieves $(m, \alpha, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \vartheta')$ from $\mathcal{L}_6$. If $\mathsf{pk}_{j,2}^{\vartheta'} = a_4$ and $\hbar^{\vartheta'} = a_5$, then algorithm $\mathcal{I}$ computes $a_2' = \mathsf{pk}_{k,1}^{\theta'}$ and $a_4' = \mathsf{pk}_{k,2}^{\vartheta'}$, and returns $(a_1, a_2', a_3, a_4', a_5, a_6, a_7, a_8)$; otherwise it returns $\perp$.
- Decryption query $\mathcal{O}_{dec1}$: For input ciphertext (C_i, pk_i), algorithm $\mathcal{I}$ retrieves $(i, x_{i,1}, \mathsf{pk}_{i,1}, x_{i,2}, \mathsf{pk}_{i,2}, x_{i,3}, \mathsf{pk}_{i,3}, \varsigma)$ from $\mathcal{L}_{key}$. If $\varsigma = 1$, then algorithm $\mathcal{I}$ returns $\mathtt{DataDec}(\mathsf{sk}_i, C)$. Otherwise, it checks if the queried ciphertext satisfies the equalities in (1), (2) and (3). If at least one condition is not met, then algorithm $\mathcal{I}$ returns $\perp$. Otherwise, it searches lists $\mathcal{L}_1$ and $\mathcal{L}_2$ for tuples $(m, \alpha, \beta, c_8, \theta) \in \mathcal{L}_1$ and $(T, \Theta) \in \mathcal{L}_2$ such that $\Theta \oplus (m\|\alpha\|\beta) = c_1$, $\mathsf{pk}_{i,1}^{\theta} = c_2$, $T = g^{\theta}$ and $c_3 = h^{\theta}$. If such tuples exist, algorithm $\mathcal{I}$ retrieves $(m, \alpha, c_8, \vartheta) \in \mathcal{L}_3$ and $(g^{\vartheta}, \Delta) \in \mathcal{L}_4$, and checks if both $\mathsf{pk}_{i,2}^{\vartheta} = c_4$ and $c_6 = g^m \cdot \Delta$ hold. If so, algorithm $\mathcal{I}$ gives m to the adversary $\mathcal{A}_2$; otherwise, it returns $\perp$.
- Decryption query $\mathcal{O}_{dec2}$: For a query $(A_j, \mathsf{pk}_i, \mathsf{pk}_j)$, algorithm $\mathcal{I}$ retrieves $(j, x_{j,1}, \mathsf{pk}_{j,1}, x_{j,2}, \mathsf{pk}_{j,2}, x_{j,3}, \mathsf{pk}_{j,3}, \varsigma)$ from $\mathcal{L}_{key}$. If $\varsigma = 1$, then algorithm $\mathcal{I}$ returns $\mathtt{AttDec}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, A_j)$. Otherwise, it checks if the queried attested ciphertext satisfies the equalities in (8), (9) and (10). If at least one condition is not met, then algorithm $\mathcal{I}$ returns $\perp$. Otherwise, it searches lists $\mathcal{L}_5$ and $\mathcal{L}_2$ for tuples $(m, \alpha, \beta, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \theta') \in \mathcal{L}_5$ and $(T, \Theta) \in \mathcal{L}_2$ such that $\Theta \oplus (m\|\alpha\|\beta) = a_1$, $\mathsf{pk}_{j,1}^{\theta'} = a_2$, $T = g^{\theta'}$ and $a_3 = h^{\theta'}$. If such tuples exist, then algorithm $\mathcal{I}$ retrieves $(m, \alpha, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \vartheta') \in \mathcal{L}_6$ and $(g^{\vartheta'}, \Delta') \in \mathcal{L}_4$, and checks whether both $\mathsf{pk}_{j,2}^{\vartheta'} = a_4$ and $a_6 = g^m \cdot \Delta'$ hold. If so, algorithm $\mathcal{I}$ gives m to the adversary $\mathcal{A}_2$; otherwise, it returns $\perp$.

Challenge: Adversary $\mathcal{A}_2$ outputs two random messages m_0 and m_1 of the same length and a challenge public key pk^*. Then, algorithm $\mathcal{I}$ generates the challenge ciphertext $C^* = (c_1^*, c_2^*, \cdots, c_8^*)$ as follows:

- Retrieve $(i^*, x_1^*, \mathsf{pk}_1^*, x_2^*, \mathsf{pk}_2^*, x_3^*, \mathsf{pk}_3^*, \varsigma^*)$ from $\mathcal{L}_{key}$. Note that $\varsigma^* = 0$, which implies $\mathsf{pk}_1^* = g^{x_1^*/u}$, $\mathsf{pk}_2^* = g^{x_2^*}$ and $\mathsf{pk}_3^* = g^{x_3^*}$.
- Randomly pick $\alpha^*, \beta^* \in_R \{0,1\}^{\lambda}$, $\vartheta^* \in_R \mathbb{Z}_p^*$, $b \in_R \{0,1\}$, $S \in_R \{0,1\}^{\tau_p + 2\lambda}$ and $U \in_R \mathbb{G}$, compute $c_1^* = S$, $c_2^* = (g^v)^{x_1^*}$, $c_3^* = (g^v)^{\tilde{a}}$, $c_4^* = (\mathsf{pk}_2^*)^{\vartheta^*}$, $c_5^* = \hbar^{\vartheta^*}$, $c_6^* = g^{m_b} \cdot U$, $c_7^* = \mathsf{OS.Sign}(\mathsf{osk}^*, c_1^*\|c_3^*\|c_5^*\|c_6^*)$ and $c_8^* = \mathsf{ovk}^*$. This process implicitly defines $\theta^* = H_1(m_b\|\alpha^*\|\beta^*\|\mathsf{ovk}^*) = uv$, $H_2(g^{\theta^*}) = (m_b\|\alpha^*\|\beta^*) \oplus S$, $H_3(m_b\|\alpha^*\|\mathsf{ovk}^*) = \vartheta^*$ and $H_4(g^{\vartheta^*}) = U$.

Then, algorithm $\mathcal{I}$ returns the challenge ciphertext C^*.

Phase 2: The adversary can continue to make queries except that the derivatives of C^* cannot be submitted for decryption and re-encryption queries.

Guess: Eventually, adversary $\mathcal{A}_2$ returns a guess b'. Algorithm $\mathcal{I}$ randomly picks a pair (T, Θ) from the list $\mathcal{H}_2$ and outputs T as the solution to the given DCDH problem instance.

Analysis. The setup and key generation responses are perfectly simulated, where the parameters and keys are distributed in the same way as in the proposed PRE-DET scheme. As long as adversary $\mathcal{A}_2$ does not submit $(m_b, \alpha^*, \beta^*, \mathsf{ovk}^*)$ to $\mathcal{O}_{H_1}$, g^{uv} to $\mathcal{O}_{H_2}$, $(m_b, \alpha^*, \mathsf{ovk}^*)$ to $\mathcal{O}_{H_3}$, g^{ϑ^*} to $\mathcal{O}_{H_4}$, $(m_b, \alpha^*, \beta^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ to $\mathcal{O}_{H_5}$, nor $(m_b, \alpha^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ to $\mathcal{O}_{H_6}$, the simulation of the random oracles are perfect. Let $\mathtt{EvtH}_1^*$, $\mathtt{EvtH}_2^*$, $\mathtt{EvtH}_3^*$, $\mathtt{EvtH}_4^*$, $\mathtt{EvtH}_5^*$ and $\mathtt{EvtH}_6^*$ respectively denote the events that $(m_b, \alpha^*, \beta^*, \mathsf{ovk}^*)$ was submitted to $\mathcal{O}_{H_1}$, g^{uv} was submitted to $\mathcal{O}_{H_2}$, $(m_b, \alpha^*, \mathsf{ovk}^*)$ was submitted to $\mathcal{O}_{H_3}$, g^{ϑ^*} was submitted to $\mathcal{O}_{H_4}$, $(m_b, \alpha^*, \beta^*, *, *)$ was submitted to $\mathcal{O}_{H_5}$, and $(m_b, \alpha^*, *, *)$ was submitted to $\mathcal{O}_{H_6}$.

The challenge ciphertext of message m_b is identically distributed as in the PRE-DET scheme. Since H_1, H_2 and H_3 are random oracles, it can be seen that $c_1^* = H_2(g^{uv}) \oplus (m_b\|\alpha^*\|\beta^*) = H_2(g^{\theta^*}) \oplus (m_b\|\alpha^*\|\beta^*)$, $c_2^* = (g^v)^{x_1^*} = (g^{x_1^*/u})^{uv} = (\mathsf{pk}_1^*)^{\theta^*}$, $c_3^* = (g^v)^{\tilde{a}} = (g^{\tilde{a}/u})^{uv} = h^{\theta^*}$, and all other components directly follow the proposed scheme. Thus, adversary $\mathcal{A}_2$ would guess $b' = b$ with the same advantage as in a real execution of the PRE-DET scheme.

The decryption responses by $\mathcal{O}_{dec1}$ are also perfect, except that algorithm $\mathcal{I}$ cannot always answer decryption queries with $c_8 = \mathsf{ovk}^*$ and may reject some valid ciphertexts. First, in Phase 1, adversary $\mathcal{A}_2$ has a $(q_{H_1} + q_{H_3}) \cdot \rho$ chance of querying oracle $\mathcal{O}_{dec1}$ with a component $c_8 = \mathsf{ovk}^*$. Second, in Phase 2, if the adversary queries $\mathcal{O}_{dec1}$ on a well-formed ciphertext C such that $c_8 = \mathsf{ovk}^*$ and C is not a derivative of C^*, then $\mathcal{A}_2$ breaks the one-time signature scheme OS, which means the adversary's chance of submitting such queries equals to $\Pr[\mathcal{A}_2\ \mathit{breaks}\ \mathsf{OS}]$. Third, consider a well-formed ciphertext C is submitted for decryption but it is generated without querying $(m\|\alpha\|\beta\|\mathsf{ovk})$ to H_1, g^{θ} to H_2, $(m\|\alpha\|\mathsf{ovk})$ to H_3 and g^{ϑ} to H_4, where $\theta = H_1(m\|\alpha\|\beta\|\mathsf{ovk})$ and $\vartheta = H_3(m\|\alpha\|\mathsf{ovk})$. Let $\mathtt{Wform}$ denote the event that C is a well-formed ciphertext, and let $\mathtt{EvtH}_1$, $\mathtt{EvtH}_2$, $\mathtt{EvtH}_3$, $\mathtt{EvtH}_4$ respectively denote the events that $(m\|\alpha\|\beta\|\mathsf{ovk})$ was queried to H_1, g^{θ} was queried to H_2, $(m\|\alpha\|\mathsf{ovk})$ was queried to H_3, and g^{ϑ} was queried to H_4. Thus,

$$\begin{aligned}
&\Pr[\mathtt{Wform} \mid \neg\mathtt{EvtH}_1 \vee \neg\mathtt{EvtH}_2 \vee \neg\mathtt{EvtH}_3 \vee \neg\mathtt{EvtH}_4] \\
&\quad \le \Pr[\mathtt{Wform} \mid \neg\mathtt{EvtH}_1] + \Pr[\mathtt{Wform} \mid \neg\mathtt{EvtH}_2] + \Pr[\mathtt{Wform} \mid \neg\mathtt{EvtH}_3] + \Pr[\mathtt{Wform} \mid \neg\mathtt{EvtH}_4]
\end{aligned}$$

$$\leq \frac{1}{p} + \frac{1}{2^{\tau_p+2\lambda}} + \frac{1}{p} + \frac{1}{p} = \frac{3}{p} + \frac{1}{p \cdot 4^\lambda} \quad (17)$$

Let $\texttt{DecErr}$ denote the event that the above defined cases happen in decryption queries to $\mathcal{O}_{dec1}$. Thus,

$$\Pr[\texttt{DecErr}] \leq (q_{H_1} + q_{H_3}) \cdot \rho + \Pr[\mathcal{A}_2 \textit{ breaks } \mathsf{OS}] + \frac{3q_D}{p} + \frac{q_D}{p \cdot 4^\lambda}$$

The responses to re-encryption queries $\mathcal{O}_{renc1}$ are perfect, as long as no well-formed ciphertexts are submitted which are produced without querying to H_1, H_2, H_3 and H_4. Let $\texttt{ReErr1}$ denote the event that such ciphertexts are queried to $\mathcal{O}_{renc1}$. Since both H_1 and H_3 are random oracles,

$$\Pr[\texttt{ReErr1}] \leq \frac{q_{R_1}}{p} + \frac{q_{R_1}}{p} = \frac{2q_{R_1}}{p}.$$

Similarly, the responses to re-encryption queries $\mathcal{O}_{renc2}$ are perfect, as long as no well-formed attested ciphertexts are submitted which are produced without querying to H_2, H_4, H_5, H_6 and H_7. Let $\texttt{ReErr2}$ denote the event that such ciphertexts are queried to $\mathcal{O}_{renc2}$. Since both H_5 and H_6 are random oracles, we know

$$\Pr[\texttt{ReErr2}] \leq \frac{q_{R_2}}{p} + \frac{q_{R_2}}{p} = \frac{2q_{R_2}}{p}.$$

Let $\texttt{Good}$ denote the event $\texttt{EvtH}_1^* \vee \texttt{EvtH}_2^* \vee \texttt{EvtH}_3^* \vee \texttt{EvtH}_4^* \vee \texttt{EvtH}_5^* \vee \texttt{EvtH}_6^* \vee \texttt{DecErr} \vee \texttt{ReErr1} \vee \texttt{ReErr2}$. If $\texttt{Good}$ does not happen, then adversary $\mathcal{A}_2$ can get no advantage in guessing $b' = b$, that is, $\Pr[b' = b | \neg\texttt{Good}] = 1/2$. Thus, according to Theorem 1,

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \frac{1}{2}\Pr[\texttt{Good}]$$

We have

$$\begin{aligned}
\varepsilon &= \left|\Pr[b' = b] - \frac{1}{2}\right| \\
&\leq \frac{1}{2}\Pr[\texttt{Good}] \\
&= \frac{1}{2}\Pr\left[\texttt{EvtH}_1^* \vee \texttt{EvtH}_2^* \vee \texttt{EvtH}_3^* \vee \texttt{EvtH}_4^* \vee \texttt{EvtH}_5^* \vee \texttt{EvtH}_6^* \vee \texttt{DecErr} \vee \texttt{ReErr1} \vee \texttt{ReErr2}\right] \\
&\leq \frac{1}{2}(\Pr[\texttt{EvtH}_1^*] + \Pr[\texttt{EvtH}_2^*] + \Pr[\texttt{EvtH}_3^*] + \Pr[\texttt{EvtH}_4^*] + \Pr[\texttt{EvtH}_5^*] + \Pr[\texttt{EvtH}_6^*] + \Pr[\texttt{DecErr}] \\
&\quad + \Pr[\texttt{ReErr1}] + \Pr[\texttt{ReErr2}])
\end{aligned}$$

As α^* and β^* are randomly chosen from $\{0, 1\}^\lambda$, we have $\Pr[\texttt{EvtH}_1^*] \leq \frac{q_{H_1}}{4^\lambda}$, $\Pr[\texttt{EvtH}_3^*] \leq \frac{q_{H_3}}{2^\lambda}$, $\Pr[\texttt{EvtH}_4^*] \leq \frac{q_{H_4}}{p}$, $\Pr[\texttt{EvtH}_5^*] \leq \frac{q_{H_5}}{4^\lambda}$ and $\Pr[\texttt{EvtH}_6^*] \leq \frac{q_{H_6}}{2^\lambda}$. Thus,

$$\begin{aligned}
\mathrm{EvtH}_2^* \quad &\geq \quad 2\varepsilon - (\Pr[\mathrm{EvtH}_1^*] + \Pr[\mathrm{EvtH}_3^*] + \Pr[\mathrm{EvtH}_4^*] + \Pr[\mathrm{EvtH}_5^*] + \Pr[\mathrm{EvtH}_6^*] + \Pr[\mathrm{DecErr}] \\
&\qquad + \ \Pr[\mathrm{ReErr1}] + \Pr[\mathrm{ReErr2}]) \\
&\geq \quad 2\varepsilon - \frac{q_{H_1} + q_{H_5}}{4^\lambda} - \frac{q_{H_3} + q_{H_6}}{2^\lambda} - \frac{q_{H_4} + 3q_D + 2q_{R_1} + 2q_{R_2}}{p} - \frac{q_D}{p \cdot 4^\lambda} - \left(q_{H_1} + q_{H_3}\right) \cdot \rho - Pr[\mathscr{A}_2 \textit{ breaks OS}]
\end{aligned}$$

Therefore, if event $\texttt{EvtH}_2^*$ happens, then algorithm $\mathcal{I}$ can solve the given DCDH instance with advantage

$$\varepsilon_{\mathsf{dcdh}} \geq \frac{1}{q_{H_2}}\left(2\varepsilon - \frac{q_{H_1} + q_{H_5}}{4^\lambda} - \frac{q_{H_3} + q_{H_6}}{2^\lambda} - \frac{q_{H_4} + 3q_D + 2q_{R_1} + 2q_{R_2}}{p} - \frac{q_D}{p \cdot 4^\lambda} - (q_{H_1} + q_{H_3}) \cdot \rho - \Pr[\mathcal{A}_2 \textit{ breaks } \mathsf{OS}]\right)$$

This concludes Theorem 2. □

Theorem 3. *Suppose the DCDH assumption holds in group $\mathbb{G}$. The proposed PRE-DET scheme offers PD-OW-CCA3 security for ciphertext against Type-3 adversary in the random oracle model.*

The proof can be captured as a special case of Theorem 4 without two types of re-encryption queries, which is thus omitted here. Specifically, if there is a Type-3 PPT adversary $\mathcal{A}_3$ that has non-negligible advantage ε in attacking the PD-OW-CCA3 security for ciphertext in the PRE-DET scheme, then one can construct an algorithm $\mathcal{I}$ to solve the DCDH problem with non-negligible probability $\varepsilon_{\mathsf{dcdh}}$ such that

$$\varepsilon_{\mathsf{dcdh}} \geq \frac{1}{q_{H_2}}\left(\varepsilon - \frac{q_{H_1} + q_{H_5}}{4^\lambda} - \frac{q_{H_3} + q_{H_6}}{2^\lambda} - \frac{q_{H_4} + 3q_D}{p} - \frac{q_D}{p \cdot 4^\lambda} - (q_{H_1} + q_{H_3}) \cdot \rho - \Pr[\mathcal{A}_3 \textit{ breaks } \mathsf{OS}]\right)$$

where $\mathcal{A}_3$ is able to issue at most q_D decryption queries to $\mathcal{O}_{dec1}$, and at most q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} and q_{H_6} hash queries to H_1, H_2, H_3, H_4, H_5 and H_6, respectively. $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ is a strong one-time signature scheme, where KGen has super-logarithmic minimum entropy and maximum probability ρ of outputting a given verification key.

Theorem 4. *Suppose the DCDH assumption holds in group $\mathbb{G}$. The proposed PRE-DET scheme offers PD-OW-CCA4 security for ciphertext against Type-4 adversary in the random oracle model.*

The following proof follows the standard framework established in [4,29,36].

Proof. Let $\mathcal{A}_4$ be a Type-4 PPT adversary that has non-negligible advantage ε in attacking the PD-OW-CCA4 security for ciphertext in the PRE-DET scheme. Suppose $\mathcal{A}_4$ issues at most q_D decryption queries to $\mathcal{O}_{dec1}$, at most q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5} and q_{H_6} hash queries to H_1, H_2, H_3, H_4, H_5 and H_6, respectively, at most q_{R_1} re-encryption queries to $\mathcal{O}_{renc1}$, and at most q_{R_2} re-encryption queries to $\mathcal{O}_{renc2}$. Let $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme, where KGen has super-logarithmic minimum entropy and maximum probability ρ of outputting a given verification key. We show that if such an adversary $\mathcal{A}_4$ exists, then one can construct an algorithm $\mathcal{I}$ to solve the DCDH problem with non-negligible probability $\varepsilon_{\mathsf{dcdh}}$.

Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be cycle groups with prime order p and bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. At first, algorithm $\mathcal{I}$ is given a DCDH instance $(g, g^{1/u}, g^{v}) \in \mathbb{G}^3$. The goal of $\mathcal{I}$ is to compute g^{uv}. Algorithm $\mathcal{I}$ simulates the challenger and interacts with adversary $\mathcal{A}_4$ as follows.

Set-up: Algorithm $\mathcal{I}$ randomly picks $\tilde{a}, \tilde{b} \in_R \mathbb{Z}_p^*$, computes $h = g^{\tilde{a}/u}$ and $\hbar = g^{\tilde{b}}$, and sets the system public parameter as $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, h, \hbar, H_1, H_2, \cdots, H_7, \mathsf{OS})$.

Algorithm $\mathcal{I}$ runs $(\mathsf{osk}^*, \mathsf{ovk}^*) \leftarrow \mathsf{OS.KGen}(1^{\lambda})$ and records the key pair.

Phase 1: The adversary adaptively makes the following queries.

- H_1 hash query $\mathcal{O}_{H_1}$: Same as in the proof of Theorem 2.
- H_2 hash query $\mathcal{O}_{H_2}$: Same as in the proof of Theorem 2.
- H_3 hash query $\mathcal{O}_{H_3}$: Same as in the proof of Theorem 2.
- H_4 hash query $\mathcal{O}_{H_4}$: Same as in the proof of Theorem 2.
- H_5 hash query $\mathcal{O}_{H_5}$: Same as in the proof of Theorem 2.
- H_6 hash query $\mathcal{O}_{H_6}$: Same as in the proof of Theorem 2.
- H_7 hash query $\mathcal{O}_{H_7}$: Same as in the proof of Theorem 2.
- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Same as in the proof of Theorem 2.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Same as in the proof of Theorem 2.
- Re-encryption key generation query $\mathcal{O}_{rkgen}$: Same as in the proof of Theorem 2.
- Re-encryption query $\mathcal{O}_{renc1}$: Same as in the proof of Theorem 2.
- Attestation query $\mathcal{O}_{att}$: Same as in the proof of Theorem 2.
- Re-encryption query $\mathcal{O}_{renc2}$: Same as in the proof of Theorem 2.
- Delegation generation query $\mathcal{O}_{delgen}$: For a query pk_i, algorithm $\mathcal{I}$ outputs $x_{i,2}$.
- Decryption query $\mathcal{O}_{dec1}$: Same as in the proof of Theorem 2.
- Decryption query $\mathcal{O}_{dec2}$: Same as in the proof of Theorem 2.

Challenge: Adversary $\mathcal{A}_4$ outputs a challenge public key pk^*. Then, algorithm $\mathcal{I}$ picks a message $m^* \in_R \mathbb{Z}_p$ and generates the challenge ciphertext $C^* = (c_1^*, c_2^*, \cdots, c_8^*)$ as follows:

- Retrieve $(i^*, x_1^*, \mathsf{pk}_1^*, x_2^*, \mathsf{pk}_2^*, x_3^*, \mathsf{pk}_3^*, \varsigma^*)$ from $\mathcal{L}_{key}$. Note that $\varsigma^* = 0$, which implies $\mathsf{pk}_1^* = g^{x_1^*/u}$, $\mathsf{pk}_2^* = g^{x_2^*}$ and $\mathsf{pk}_3^* = g^{x_3^*}$.
- Randomly pick $\alpha^*, \beta^* \in_R \{0, 1\}^{\lambda}$, $\vartheta^* \in_R \mathbb{Z}_p^*$, $S \in_R \{0,1\}^{\tau_p + 2\lambda}$ and $U \in_R \mathbb{G}$, compute $c_1^* = S$, $c_2^* = (g^v)^{x_1^*}$, $c_3^* = (g^v)^{\tilde{a}}$, $c_4^* = (\mathsf{pk}_2^*)^{\vartheta^*}$, $c_5^* = \hbar^{\vartheta^*}$, $c_6^* = g^{m^*} \cdot U$, $c_7^* = \mathsf{OS.Sign}(\mathsf{osk}^*, c_1^* \| c_3^* \| c_5^* \| c_6^*)$ and $c_8^* = \mathsf{ovk}^*$. This process implicitly defines $\theta^* = H_1(m^* \| \alpha^* \| \beta^* \| \mathsf{ovk}^*) = uv$, $H_2(g^{\theta^*}) = (m^* \| \alpha^* \| \beta^*) \oplus S$, $H_3(m^* \| \alpha^* \| \mathsf{ovk}^*) = \vartheta^*$ and $H_4(g^{\vartheta^*}) = U$.

Then, algorithm $\mathcal{I}$ returns the challenge ciphertext C^*.

Phase 2: The adversary can continue to make queries except that the derivatives of C^* cannot be submitted for decryption and re-encryption queries.

Guess: Eventually, adversary $\mathcal{A}_4$ outputs a guess m'. Algorithm $\mathcal{I}$ randomly picks a pair (T, Θ) from the list $\mathcal{H}_2$ and outputs T as the solution to the given DCDH problem instance.

Analysis. The setup and key generation responses are perfectly simulated, where the parameters and keys are distributed in the same way as in the proposed PRE-DET scheme. As long as adversary $\mathcal{A}_4$ does not submit $(m^*, \alpha^*, \beta^*, \mathsf{ovk}^*)$ to $\mathcal{O}_{H_1}$, g^{uv} to $\mathcal{O}_{H_2}$, $(m^*, \alpha^*, \mathsf{ovk}^*)$ to $\mathcal{O}_{H_3}$, g^{ϑ^*} to $\mathcal{O}_{H_4}$, $(m^*, \alpha^*, \beta^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ to $\mathcal{O}_{H_5}$, nor $(m^*, \alpha^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ to $\mathcal{O}_{H_6}$, the simulation of the random oracles are perfect. Let $\mathtt{EvtH}_1^*$, $\mathtt{EvtH}_2^*$, $\mathtt{EvtH}_3^*$, $\mathtt{EvtH}_4^*$, $\mathtt{EvtH}_5^*$ and $\mathtt{EvtH}_6^*$ respectively denote the events that $(m^*, \alpha^*, \beta^*, \mathsf{ovk}^*)$ was submitted to $\mathcal{O}_{H_1}$, g^{uv} was submitted to $\mathcal{O}_{H_2}$, $(m^*, \alpha^*, \mathsf{ovk}^*)$ was submitted to $\mathcal{O}_{H_3}$, g^{ϑ^*} was submitted to $\mathcal{O}_{H_4}$, $(m^*, \alpha^*, \beta^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ was submitted to $\mathcal{O}_{H_5}$, and $(m^*, \alpha^*, \mathsf{pk}_1^*, \mathsf{pk}_2^*)$ was submitted to $\mathcal{O}_{H_6}$.

The challenge ciphertext of message m^* is identically distributed as in the PRE-DET scheme. Since H_1, H_2, H_3 and H_4 are random oracles, it can be seen that $c_1^* = H_2(g^{uv}) \oplus (m^* \| \alpha^* \| \beta^*) = H_2(g^{\theta^*}) \oplus (m^* \| \alpha^* \| \beta^*)$, $c_2^* = (g^v)^{x_1^*} = (g^{x_1^*/u})^{uv} = (\mathsf{pk}_1^*)^{\theta^*}$, $c_3^* = (g^v)^{\tilde{a}} = (g^{\tilde{a}/u})^{uv} = h^{\theta^*}$, and all other components directly follow the proposed scheme. Thus, adversary $\mathcal{A}_4$ would guess $m' = m^*$ with the same advantage as in a real execution of the PRE-DET scheme.

The decryption responses by $\mathcal{O}_{dec1}$ are also perfect, except that algorithm $\mathcal{I}$ cannot always answer decryption queries with $c_8 = \mathsf{ovk}^*$ and may reject some valid ciphertexts. First, in Phase 1, adversary $\mathcal{A}_4$ has a $(q_{H_1} + q_{H_3}) \cdot \rho$ chance in querying oracle $\mathcal{O}_{dec1}$ with a component $c_8 = \mathsf{ovk}^*$. Second, in Phase 2, if the adversary queries $\mathcal{O}_{dec1}$ on a well-formed ciphertext C such that $c_8 = \mathsf{ovk}^*$, and C is not a derivative of C^*, then $\mathcal{A}_4$ breaks the one-time signature scheme OS, which means the adversary's chance of submitting such queries equals to $\Pr[\mathcal{A}_4 \textit{ breaks } \mathsf{OS}]$. Third, consider a well-formed ciphertext C is submitted for decryption but it is generated without querying $(m\|\alpha\|\beta\|\mathsf{ovk})$ to H_1, g^θ to H_2, $(m\|\alpha\|\mathsf{ovk})$ to H_3 and g^ϑ to H_4, where $\theta = H_1(m\|\alpha\|\beta\|\mathsf{ovk})$ and $\vartheta = H_3(m\|\alpha\|\mathsf{ovk})$. Let $\texttt{Wform}$ denote the event that C is a well-formed ciphertext, and let $\texttt{EvtH}_1$, $\texttt{EvtH}_2$, $\texttt{EvtH}_3$, $\texttt{EvtH}_4$ respectively denote the events that $(m\|\alpha\|\beta\|\mathsf{ovk})$ was queried to H_1, g^θ was queried to H_2, $(m\|\alpha\|\mathsf{ovk})$ was queried to H_3, and g^ϑ was queried to H_4. Let $\texttt{DecErr}$ denote the event that the above defined cases happen in decryption queries to $\mathcal{O}_{dec1}$. Thus,

$$\Pr[\texttt{DecErr}] \le (q_{H_1} + q_{H_3}) \cdot \rho + \Pr[\mathcal{A}_4 \textit{ breaks } \mathsf{OS}] + \frac{3q_D}{p} + \frac{q_D}{p \cdot 4^\lambda}$$

The responses to re-encryption queries $\mathcal{O}_{renc1}$ are perfect, as long as no well-formed ciphertexts are submitted which are produced without querying to H_1, H_2, H_3 and H_4. Let $\texttt{ReErr1}$ denote the event that such ciphertexts are queried to $\mathcal{O}_{renc1}$. Since both H_1 and H_3 are random oracles,

$$\Pr[\texttt{ReErr1}] \le \frac{q_{R_1}}{p} + \frac{q_{R_1}}{p} = \frac{2q_{R_1}}{p}.$$

Similarly, the responses to re-encryption queries $\mathcal{O}_{renc2}$ are perfect, as long as no well-formed attested ciphertexts are submitted which are produced without querying to H_2, H_4, H_5, H_6 and H_7. Let $\texttt{ReErr2}$ denote the event that such ciphertexts are queried to $\mathcal{O}_{renc2}$. Since both H_5 and H_6 are random oracles,

$$\Pr[\texttt{ReErr2}] \le \frac{q_{R_2}}{p} + \frac{q_{R_2}}{p} = \frac{2q_{R_2}}{p}.$$

Let $\texttt{Good}$ denote the event $\texttt{EvtH}_1^* \vee \texttt{EvtH}_2^* \vee \texttt{EvtH}_3^* \vee \texttt{EvtH}_4^* \vee \texttt{EvtH}_5^* \vee \texttt{EvtH}_6^* \vee \texttt{DecErr} \vee \texttt{ReErr1} \vee \texttt{ReErr2}$. If $\texttt{Good}$ does not happen, then adversary $\mathcal{A}_4$ can get no advantage in guessing $m' = m^*$. Thus,

$$\begin{aligned}
\varepsilon &= \Pr[m' = m^*] \\
&\le \Pr[\texttt{Good}] \\
&= \Pr[\texttt{EvtH}_1^* \vee \texttt{EvtH}_2^* \vee \texttt{EvtH}_3^* \vee \texttt{EvtH}_4^* \vee \texttt{EvtH}_5^* \vee \texttt{EvtH}_6^* \vee \texttt{DecErr} \vee \texttt{ReErr1} \vee \texttt{ReErr2}] \\
&\le \Pr[\texttt{EvtH}_1^*] + \Pr[\texttt{EvtH}_2^*] + \Pr[\texttt{EvtH}_3^*] + \Pr[\texttt{EvtH}_4^*] + \Pr[\texttt{EvtH}_5^*] + \Pr[\texttt{EvtH}_6^*] \\
&\quad + \Pr[\texttt{DecErr}] + \Pr[\texttt{ReErr1}] + \Pr[\texttt{ReErr2}]
\end{aligned}$$

As α^* and β^* are randomly chosen from $\{0, 1\}^\lambda$, we have $\Pr[\texttt{EvtH}_1^*] \le \frac{q_{H_1}}{4^\lambda}$, $\Pr[\texttt{EvtH}_3^*] \le \frac{q_{H_3}}{2^\lambda}$, $\Pr[\texttt{EvtH}_4^*] \le \frac{q_{H_4}}{p}$, $\Pr[\texttt{EvtH}_5^*] \le \frac{q_{H_5}}{4^\lambda}$ and $\Pr[\texttt{EvtH}_6^*] \le \frac{q_{H_6}}{2^\lambda}$. Thus,

$$\begin{aligned}
\mathrm{EvtH}_2^* \quad &\ge \varepsilon - (\Pr[\mathrm{EvtH}_1^*] + \Pr[\mathrm{EvtH}_3^*] + \Pr[\mathrm{EvtH}_4^*] + \Pr[\mathrm{EvtH}_5^*] + \Pr[\mathrm{EvtH}_6^*] + \Pr[\mathrm{DecErr}] \\
&\quad + \Pr[\mathrm{ReErr1}] + \Pr[\mathrm{ReErr2}]) \\
&\ge \varepsilon - \frac{q_{H_1} + q_{H_5}}{4^\lambda} - \frac{q_{H_3} + q_{H_6}}{2^\lambda} - \frac{q_{H_4} + 3q_D + 2q_{R_1} + 2q_{R_2}}{p} - \frac{q_D}{p \cdot 4^\lambda} - \left(q_{H_1} + q_{H_3}\right) \cdot \rho \qquad - Pr[\mathscr{A}_4 \textit{ breaks } OS]
\end{aligned}$$

Therefore, if event $\texttt{EvtH}_2^*$ happens, then algorithm $\mathcal{I}$ can solve the given DCDH instance with advantage

$$\varepsilon_{\mathsf{dcdh}} \ge \frac{1}{q_{H_2}} \left(\varepsilon - \frac{q_{H_1} + q_{H_5}}{4^\lambda} - \frac{q_{H_3} + q_{H_6}}{2^\lambda} - \frac{q_{H_4} + 3q_D + 2q_{R_1} + 2q_{R_2}}{p} - \frac{q_D}{p \cdot 4^\lambda} - (q_{H_1} + q_{H_3}) \cdot \rho - \Pr[\mathcal{A}_4 \textit{ breaks } \mathsf{OS}] \right)$$

This concludes Theorem 4. □

Theorem 5. *Suppose the CDH assumption holds in group $\mathbb{G}$. The proposed PRE-DET scheme offers PD-EUCMA security for attested ciphertext against Type-5 adversary in the random oracle model.*

The proof for Theorem 5 follows the standard framework established in [3].

Proof. Let $\mathcal{A}_5$ be a Type-5 PPT adversary that has non-negligible advantage ε in attacking the PD-EUCMA security for attested ciphertext in the PRE-DET scheme. Suppose $\mathcal{A}_5$ issues at most q_A attestation queries, and at most q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_5}, q_{H_6} and q_{H_7} hash queries. Let $\mathsf{OS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a strong one-time signature scheme. We show that if such an adversary $\mathcal{A}_5$ exists, then one can construct an algorithm $\mathcal{I}$ to solve the CDH problem with non-negligible probability $\varepsilon_{\mathsf{cdh}}$.

Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be cycle groups with prime order p and bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. At first, algorithm $\mathcal{I}$ is given a CDH instance $(g, g^u, g^v) \in \mathbb{G}^3$. The goal of $\mathcal{I}$ is to compute g^{uv}. Algorithm $\mathcal{I}$ simulates the challenger and interacts with adversary $\mathcal{A}_5$ as follows.

Set-up: Algorithm $\mathcal{I}$ randomly picks $\tilde{a}, \tilde{b} \in_R \mathbb{Z}_p^*$, computes $h = g^{\tilde{a}}$ and $\hbar = g^{\tilde{b}}$, and sets the system public parameter as $\mathsf{par} = (\mathbb{G}, \mathbb{G}_T, \hat{e}, p, g, h, \hbar, H_1, H_2, \cdots, H_7, \mathsf{OS})$. Algorithm $\mathcal{I}$ randomly picks $x_1^*, x_2^* \in_R \mathbb{Z}_p^*$, sets $\mathsf{sk}_1^* = x_1^*$, $\mathsf{sk}_2^* = x_2^*$, $\mathsf{pk}_3^* = g^u$ which implies $\mathsf{sk}_3^* = u$, and computes $\mathsf{pk}_1^* = g^{x_1^*}$, $\mathsf{pk}_2 = g^{x_2^*}$. Next, algorithm $\mathcal{I}$ publishes $\mathsf{pk}^* = (\mathsf{pk}_1^*, \mathsf{pk}_2^*, \mathsf{pk}_3^*)$ to $\mathcal{A}_5$ and adds

$(0, x_1^*, \mathsf{pk}_1^*, x_2^*, \mathsf{pk}_2^*, \top, \mathsf{pk}_3^*, 0)$ to the list $\mathcal{L}_{key}$, where $\top$ denotes an unknown value and the last entry '0' denotes that pk^* is an uncorrupted public key.

Queries: Adversary $\mathcal{A}$ can adaptively submit the following queries.

- H_1 hash query $\mathcal{O}_{H_1}$: Same as in the proof of Theorem 4.
- H_2 hash query $\mathcal{O}_{H_2}$: Same as in the proof of Theorem 4.
- H_3 hash query $\mathcal{O}_{H_3}$: Same as in the proof of Theorem 4.
- H_4 hash query $\mathcal{O}_{H_4}$: Same as in the proof of Theorem 4.
- H_5 hash query $\mathcal{O}_{H_5}$: Same as in the proof of Theorem 4.
- H_6 hash query $\mathcal{O}_{H_6}$: Same as in the proof of Theorem 4.
- H_7 hash query $\mathcal{O}_{H_7}$: For answering $\mathcal{O}_{H_7}$ queries, algorithm $\mathcal{I}$ maintains a list $\mathcal{L}_7$ which is initially empty. For an input tuple $(a_1, a_3, a_5, a_6, att)$, if there exists an entry $(a_1, a_3, a_5, a_6, att, \mu, \nu, \Lambda) \in \mathcal{L}_7$, then $\mathcal{O}_{H_7}$ responds with Λ; otherwise, $\mathcal{I}$ picks a random coin $\mu \in_R \{0, 1\}$ such that $\Pr[\mu = 0] = \frac{1}{q_A+1}$, picks a random value $\nu \in_R \mathbb{Z}_p^*$, computes $\Lambda = (g^v)^{1-\mu} g^\nu \in \mathbb{G}$, returns Λ and appends $(a_1, a_3, a_5, a_6, att, \mu, \nu, \Lambda)$ to $\mathcal{L}_7$.
- Uncorrupted key generation query $\mathcal{O}_{ukgen}$: Algorithm $\mathcal{I}$ randomly picks $x_{i,1}, x_{i,2}, x_{i,3} \in_R \mathbb{Z}_p^*$, sets $\mathsf{sk}_{i,1} = x_{i,1}$, $\mathsf{sk}_{i,2} = x_{i,2}$, $\mathsf{sk}_{i,3} = x_{i,3}$, and computes $\mathsf{pk}_{i,1} = g^{x_{i,1}}$, $\mathsf{pk}_{i,2} = g^{x_{i,2}}$ and $\mathsf{pk}_{i,3} = g^{x_{i,3}}$. Next, algorithm $\mathcal{I}$ gives $\mathsf{pk}_i = (\mathsf{pk}_{i,1}, \mathsf{pk}_{i,2}, \mathsf{pk}_{i,3})$ to $\mathcal{A}_5$ and adds $(i, x_{i,1}, \mathsf{pk}_{i,1}, x_{i,2}, \mathsf{pk}_{i,2}, x_{i,3}, \mathsf{pk}_{i,3}, 0)$ to the list $\mathcal{L}_{key}$, where '0' denotes that pk_i is an uncorrupted public key.
- Corrupted key generation query $\mathcal{O}_{ckgen}$: Same as in the proof of Theorem 4.
- Re-encryption key generation query $\mathcal{O}_{rkgen}$: For a queried pair $(\mathsf{pk}_i, \mathsf{pk}_j)$, algorithm $\mathcal{I}$ outputs $\mathsf{rk}_{i\leftrightarrow j} = (x_{j,1}/x_{i,1} \bmod p, x_{j,2}/x_{i,2} \bmod p)$.
- Attestation query $\mathcal{O}_{att}$: For a query (C_i, pk_i), algorithm $\mathcal{I}$ performs the oracle $\mathcal{O}_{dec1}$ with input (C_i, pk_i). If $\mathcal{O}_{dec1}$ outputs $\perp$, then algorithm $\mathcal{I}$ returns $\perp$ and halts. Otherwise, letting $(m_i, \alpha_i, \beta_i, c_{i,8}, \theta_i) \in \mathcal{L}_1$ be the retrieved tuple in $\mathcal{O}_{dec1}$, algorithm $\mathcal{I}$ runs the oracle $\mathcal{O}_{H_5}$ with input $(m_i, \alpha_i, \beta_i, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ to get θ_i', runs $\mathcal{O}_{H_2}$ with input $g^{\theta_i'}$ to get Θ_i', runs $\mathcal{O}_{H_6}$ with input $(m_i, \alpha_i, \mathsf{pk}_{i,1}, \mathsf{pk}_{i,2})$ to get ϑ_i', and runs $\mathcal{O}_{H_4}$ with input $g^{\vartheta_i'}$ to get Δ_i'. Then algorithm $\mathcal{I}$ computes $a_{i,1} = \Theta_i' \oplus (m_i \| \alpha_i \| \beta_i)$, $a_{i,2} = \mathsf{pk}_{i,1}^{\theta_i'}$, $a_{i,3} = h^{\theta_i'}$, $a_{i,4} = \mathsf{pk}_{i,2}^{\vartheta_i'}$, $a_{i,5} = \hbar^{\vartheta_i'}$ and $a_{i,6} = g^{m_i} \cdot \Delta_i'$, chooses $a_{i,8} = att_i \in \mathbb{Z}_p$, and runs $\mathcal{O}_{H_7}$ with input $(a_{i,1}, a_{i,3}, a_{i,5}, a_{i,6}, a_{i,8})$. Let $(a_{i,1}, a_{i,3}, a_{i,5}, a_{i,6}, att_i, \mu_i, \nu_i, \Lambda_i)$ be the corresponding entry in list $\mathcal{L}_7$. To compute $a_{i,7}$, there are two cases to consider:

  Case 1: $\mathsf{pk}_i \neq \mathsf{pk}^*$. Algorithm $\mathcal{I}$ computes $a_{i,7} = \Lambda_i^{\mathsf{sk}_{i,3}}$.

  Case 2: $\mathsf{pk}_i = \mathsf{pk}^*$. If $\mu_i = 0$, then algorithm $\mathcal{I}$ reports failure and aborts the game. Otherwise, algorithm $\mathcal{I}$ computes $a_{i,7} = (g^u)^{\nu_i}$, where $H_7(a_{i,1} \| a_{i,3} \| a_{i,5} \| a_{i,6} \| a_{i,8}) = g^{\nu_i} \in \mathbb{G}$. Note that the attested ciphertexts are perfectly simulated in adversary $\mathcal{A}$'s view when the abortion case does not occur.

  At last, algorithm $\mathcal{I}$ returns $A_i = (a_{i,1}, a_{i,2}, a_{i,3}, a_{i,4}, a_{i,5}, a_{i,6}, a_{i,7}, a_{i,8})$.
- Delegation generation query $\mathcal{O}_{delgen}$: Same as in the proof of Theorem 4.
- Decryption query $\mathcal{O}_{dec1}$: For input ciphertext (C_i, pk_i), algorithm $\mathcal{I}$ returns $\mathtt{DataDec}(\mathsf{sk}_i, C)$.
- Decryption query $\mathcal{O}_{dec2}$: For a query $(A_j, \mathsf{pk}_i, \mathsf{pk}_j)$, algorithm $\mathcal{I}$ returns $\mathtt{AttDec}(\mathsf{pk}_i, \mathsf{pk}_j, \mathsf{sk}_j, A_j)$.

Output: Eventually, adversary $\mathcal{A}$ outputs a tuple $(C^*, A^*, \mathsf{pk}_i^*)$ such that C^* is a well-formed ciphertext under pk^* and every derivative of (C^*, pk^*) has not been queried to $\mathcal{O}_{att}$. Assume $(A^*, \mathsf{pk}^*, \mathsf{pk}_i^*)$ is a valid derivative of (C^*, pk^*); otherwise, $\mathcal{I}$ reports failure and aborts the game. In the random oracle model, $(a_1^*, a_3^*, a_5^*, a_6^*, a_8^*)$ should have been queried to $\mathcal{O}_{H_7}$.

Algorithm $\mathcal{I}$ retrieves the tuple $(a_1^*, a_3^*, a_5^*, a_6^*, a_8^*, \mu^*, \nu^*, \Lambda^*)$ from the list $\mathcal{L}_7$. If $\mu^* = 1$, then $\mathcal{I}$ reports failure and aborts the game. Otherwise, i.e., $\mu^* = 0$, we know $H_7(a_1^* \| a_3^* \| a_5^* \| a_6^* \| a_8^*) = \Lambda^* = g^v \cdot g^{\nu^*} \in \mathbb{G}$. Therefore, $a_7^* = g^{uv} \cdot g^{u\nu^*}$. Next, algorithm $\mathcal{I}$ computes $g^{uv} = a_7^*/(g^u)^{\nu^*}$.

To analyze the probability of solving the given CDH instance, we define three events:

- Let $\mathtt{Evt}_1$ be the event that algorithm $\mathcal{I}$ does not abort in responding to attestation queries.
- Let $\mathtt{Evt}_2$ be the event that $(A^*, \mathsf{pk}^*, \mathsf{pk}_i^*)$ is a valid forged derivative of (C^*, pk^*).
- Let $\mathtt{Evt}_3$ be the event that $\mu^* = 1$.

As discussed in [3], we know

$$\Pr[\mathtt{Evt}_1] = \left(1 - \frac{1}{q_A+1}\right)^{q_A} \geq \frac{1}{e}, \quad \Pr[\mathtt{Evt}_2 | \mathtt{Evt}_1] \geq \varepsilon, \quad \Pr[\mathtt{Evt}_3 | \mathtt{Evt}_2 \cap \mathtt{Evt}_1] = \frac{1}{q_A+1}$$

where e denotes the base of the natural logarithm. Therefore, algorithm $\mathcal{I}$ can correctly solve the given CDH problem with the following probability:

$$\Pr[\mathcal{I}_{success}] = \Pr[\mathtt{Evt}_1 \cap \mathtt{Evt}_2 \cap \mathtt{Evt}_3] = \Pr[\mathtt{Evt}_1] \cdot \Pr[\mathtt{Evt}_2 | \mathtt{Evt}_1] \cdot \Pr[\mathtt{Evt}_3 | \mathtt{Evt}_2 \cap \mathtt{Evt}_1] \geq \frac{\varepsilon}{e(q_A+1)}$$

This completes the proof of Theorem 5. $\square$

Table 1
Comparison with related encryption schemes.

Scheme	Ciphertext size	Computation cost		
		Encryption	Decryption	Equality test
Yang et al. [40]	$3\tau_G + \tau_p$	$3\delta_G$	$3\delta_G$	$2\delta_{\hat{e}}$
Tang [28]	$\tau_G + 2\tau_{G_1} + \tau_p + \tau_M + \lambda$	$2\delta_G + 2\delta_{G_1}$	$2\delta_G$	$4\delta_{a\hat{e}}$
Tang [29]	$3\tau_G + \tau_p + \tau_M + \lambda$	$5\delta_G$	$2\delta_G$	$4\delta_G$
Lee et al. [11]	$3\tau_G + \tau_p$	$4\delta_G$	$3\delta_G$	$2\delta_G + 2\delta_{\hat{e}}$
Ma et al. [19]	$5\tau_G + \tau_p$	$6\delta_G$	$5\delta_G$	$2\delta_G + 2\delta_{\hat{e}}$
Ma [18]	$5\tau_G + \tau_p$	$6\delta_G + 2\delta_{\hat{e}}$	$2\delta_G + 2\delta_{\hat{e}}$	$4\delta_{\hat{e}}$
Ma et al. [20]	$\tau_{G_1} + 3\tau_{G_2} + \tau_p$	$\delta_{G_1} + 3\delta_{G_2}$ $+\delta_{G_T} + \delta_{\hat{de}}$	$\delta_{G_1} + 2\delta_{G_2}$ $+\delta_{G_T} + \delta_{\hat{de}}$	$4\delta_{\hat{de}}$
Wang and Pang [32]	$5\tau_G + \tau_p$	$8\delta_G + \delta_{\hat{e}}$	$3\delta_G + 4\delta_{\hat{e}}$	$2\delta_G + 4\delta_{\hat{e}}$
Slamanig, Spreitzer and Unterluggauer [27]	$4\tau_{G_1} + 3\tau_p$	$6\delta_{G_1}$	$5\delta_{G_1}$	$2\delta_{\hat{de}}$
Wang et al. [34]	$4\tau_{G_1}$	$4\delta_{G_1}$	$3\delta_{G_1}$	$2\delta_{\hat{de}}$
Pang and Ding [23]	$7\tau_G + \tau_{G_T}$	$7\delta_G + \delta_{\hat{e}}$	–	$2\delta_G + 5\delta_{\hat{e}}$
This paper	(a) $5\tau_G + \tau_p + 2\lambda + \tau_{os} + q(\lambda)$	$7\delta_G + \delta_{os}$	$5\delta_G + 4\delta_{\hat{e}} + \delta_{ov}$	$2\delta_G$
	(b) $6\tau_G + 2\tau_p + 2\lambda$	$13\delta_G + 4\delta_{\hat{e}} + \delta_{ov}$	$5\delta_G + 6\delta_{\hat{e}}$	$2\delta_G$

5. Analysis and comparison

In this section, we analyze and compare our PRE-DET construction with existing encryption techniques. Table 1 summarizes the comparison in terms of ciphertext size and computation costs of encryption, decryption and equality test. In the comparison, we focus mainly on resource-intensive computations including exponentiation and bilinear mapping, whereas all lightweight computations such as addition and hash evaluation are omitted.

In Table 1, we let τ_G denote the element size in group $\mathbb{G}$, and δ_G and $\delta_{\hat{e}}$ respectively represent the evaluation costs of an exponentiation in $\mathbb{G}$ and a bilinear map $\hat{e}(\cdot,\cdot)$ for a symmetric bilinear map $\hat{e}:\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$. Similarly, for an asymmetric bilinear map $\hat{e}:\mathbb{G}_1\times\mathbb{G}_2\to\mathbb{G}_T$, we let τ_{G_1} and τ_{G_2} respectively denote the element sizes in $\mathbb{G}_1$ and $\mathbb{G}_2$, whereas δ_{G_1}, δ_{G_2} and $\delta_{a\hat{e}}$ respectively represent the evaluation costs of an exponentiation in $\mathbb{G}_1$ and $\mathbb{G}_2$ and a bilinear map $\hat{e}(\cdot,\cdot)$. Also, we use τ_p and τ_{G_T} to respectively denote the element sizes in $\mathbb{Z}_p$ and $\mathbb{G}_T$ for both types of bilinear maps, and δ_{G_T} to denote the cost of an exponentiation in $\mathbb{G}_T$. For Tang's schemes [28,29], we let τ_{G} and τ_M respectively represent the size of an ordinary multiplicative cyclic group G and message space $\mathcal{M}$, whereas δ_{G} denotes the computation cost of an exponentiation in G. For the one-time signature scheme OS employed in our PRE-DET scheme, we let τ_{os} denote its signature size, and δ_{os} and δ_{ov} respectively represent the computation costs of OS.Sign and OS.Vrfy.

The efficiency of ciphertexts and attested ciphertexts of our PRE-DET scheme are given in lines (a) and (b), respectively. From the table, we see that only our PRE-DET scheme supports ciphertext re-encryption. Also, our PRE-DET construction allows the user to add attestation to ciphertext, without affecting the functionality of equality test.

Our PRE-DET construction can be implemented using the Pairing Based Cryptography Library (PBC, http://crypto.stanford.edu/pbc/). When executed on a system with Intel(R) Core(TM) i5-5200U CPU at 2.20GHz, 8.00GB RAM and running Windows 7, and chosen the elliptic curve of Type A ($y^2 = x^3 + x$) such that p is a 160-bit prime and $\tau_G = 256$, we obtained the benchmark where $\delta_{\hat{e}} = 2.4$ ms, $\delta_G = 2.7$ ms and $\delta_{G_T} = 0.6$ ms. With this benchmark, it is easy to estimate the rough running time of every procedure of our PRE-DET construction.

6. Conclusion

Motivated by the need to support partitioning and attestation on encrypted data in a secure data sharing clique, we introduced the notion of public key re-encryption with delegated equality test on ciphertexts (PRE-DET). We formalized the PRE-DET framework and its security model with respect to five types of adversaries, four for message confidentiality and one for attestation unforgeability. We then proposed a concrete PRE-DET construction in symmetric bilinear groups and formally proved its security in the formal security model. An analysis and comparison with related schemes showed the practicality of our PRE-DET construction.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yujue Wang: Conceptualization, Writing - original draft. **HweeHwa Pang:** Conceptualization, Methodology, Writing - original draft, Project administration. **Robert H. Deng:** Writing - original draft, Supervision, Project administration. **Yong Ding:** Writing - review & editing, Validation, Project administration. **Qianhong Wu:** Writing - review & editing, Supervision, Project administration. **Bo Qin:** Writing - review & editing, Project administration. **Kefeng Fan:** Writing - review & editing.

Acknowledgments

This research is supported by the Singapore National Research Foundation under NCR Award Number NRF2014NCR-NCR001-012. This article is also supported in part by the National Key R&D Program of China through project 2017YFB0802500, the National Natural Science Foundation of China under projects 61862012, 61772150, 61972019, 61932011, 61772538, 61672083, 61532021, 91646203, 61962012, and 61902123, the Guangxi Key R&D Program under project AB17195025, the Guangxi Natural Science Foundation under grants 2018GXNSFDA281054, 2018GXNSFAA281232, 2019GXNSFFA245015 and AD19245048, the National Cryptography Development Fund of China under projects MMJJ20170217 and MMJJ20170106, the foundation of Science and Technology on Information Assurance Laboratory through project 61421120305162112006, and the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004.

References

[1] F. Bao, R.H. Deng, H. Zhu, Variations of Diffie-Hellman Problem, in: S. Qing, D. Gollmann, J. Zhou (Eds.), Information and Communications Security: 5th International Conference, ICICS 2003, Huhehaote, China, October 10–13, 2003. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 301–312, doi:10.1007/978-3-540-39927-8_28.

[2] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography, in: K. Nyberg (Ed.), Advances in Cryptology — EUROCRYPT'98, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 127–144.

[3] D. Boneh, B. Lynn, H. Shacham, Short signatures from the weil pairing, J. Cryptol. 17 (4) (2004) 297–319.

[4] R. Canetti, S. Hohenberger, Chosen-ciphertext secure proxy re-encryption, in: Proceedings of the 14th ACM Conference on Computer and Communications Security, in: CCS '07, ACM, New York, NY, USA, 2007, pp. 185–194, doi:10.1145/1315245.1315269.

[5] C.-K. Chu, W.-G. Tzeng, Identity-based proxy re-encryption without random oracles, in: Proceedings of the 10th International Conference on Information Security, in: ISC'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 189–202.

[6] H. Cui, R. ... upporting secure deduplication of encrypted data in cloud, IEEE Trans. Big Data 5 (3) (2019) 330–342,

[7] R.H. Den ... ecure proxy re-encryption without pairings, in: M.K. Franklin, L.C.K. Hui, D.S. Wong (Eds.), Cryptolog ... erg, Berlin, Heidelberg, 2008, pp. 1–17.

[8] T. Elgam ... cheme based on discrete logarithms, IEEE Trans. Inf. Theory 31 (4) (1985) 469–472, doi:10.1109/TIT.1985.1057074.

[9] M. Green, G. Ateniese, Identity-based proxy re-encryption, in: Proceedings of the 5th International Conference on Applied Cryptography and Network Security, in: ACNS'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 288–306, doi:10.1007/978-3-540-72738-5_19.

[10] K. Huang, R. Tso, Y.-C. Chen, S.M.M. Rahman, A. Almogren, A. Alamri, PKE-AET: public key encryption with authorized equality test, Comput. J. 58 (10) (2015) 2686–2697.

[11] H.T. Lee, S. Ling, J.H. Seo, H. Wang, CCA2 attack and modification of huang et al.'s public key encryption with authorized equality test, Comput. J. (2016), doi:10.1093/comjnl/bxw033.

[12] H.T. Lee, S. Ling, J.H. Seo, H. Wang, Semi-generic construction of public key encryption and identity-based encryption with equality test, Inf. Sci. 373 (2016) 419–440, doi:10.1016/j.ins.2016.09.013.

[13] J. Li, Y. Zhang, X. Chen, Y. Xiang, Secure attribute-based data sharing for resource-limited users in cloud computing, Comput. Secur. 72 (2018) 1–12, doi:10.1016/j.cose.2017.08.007.

[14] T. Li, J. Li, Z. Liu, P. Li, C. Jia, Differentially private naive bayes learning over multiple data sources, Inf. Sci. 444 (2018) 89–104, doi:10.1016/j.ins.2018.02.056.

[15] K. Liang, C. Su, J. Chen, J.K. Liu, Efficient multi-function data sharing and searching mechanism for cloud-based encrypted data, in: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, in: ASIA CCS '16, ACM, New York, NY, USA, 2016, pp. 83–94, doi:10.1145/2897845.2897865.

[16] K. Liang, W. Susilo, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 10 (9) (2015) 1981–1992, doi:10.1109/TIFS.2015.2442215.

[17] B. Libert, D. Vergnaud, Unidirectional chosen-ciphertext secure proxy re-encryption, IEEE Trans. Inf. Theor. 57 (3) (2011) 1786–1802, doi:10.1109/TIT.2011.2104470.

[18] S. Ma, Identity-based encryption with outsourced equality test in cloud computing, Inf. Sci. 328 (2016) 389–402, doi:10.1016/j.ins.2015.08.053.

[19] S. Ma, Q. Huang, M. Zhang, B. Yang, Efficient public key encryption with equality test supporting flexible authorization, IEEE Trans. Inf. Forensics Secur. 10 (3) (2015) 458–470.

[20] S. Ma, M. Zhang, Q. Huang, B. Yang, Public key encryption with delegated equality test in a multi-user setting, Comput. J. 58 (4) (2015) 986–1002.

[21] M. Mambo, E. Okamoto, Proxy cryptosystems: delegation of the power to decrypt ciphertexts, IEICE Trans. Fundam. Electron.Commun. Comput. Sci. E80–A (1) (1997) 54–63.

[22] D. Nuñez, I. Agudo, J. Lopez, Proxy re-encryption: analysis of constructions and its application to secure access delegation, J. Netw. Comput. Appl. 87 (2017) 193–209, doi:10.1016/j.jnca.2017.03.005.

[23] H. Pang, X. Ding, Privacy-preserving ad-hoc equi-join on outsourced data, ACM Trans. Database Syst. 39 (3) (2014) 23:1–23:40, doi:10.1145/2629501.

[24] J.W. Seo, D.H. Yum, P.J. Lee, Comments on "unidirectional chosen-ciphertext secure proxy re-encryption", IEEE Trans. Inf. Theor. 59 (5) (2013), doi:10.1109/TIT.2012.2236606. 3256–3256.

[25] J. Shao, Z. Cao, Multi-use unidirectional identity-based proxy re-encryption from hierarchical identity-based encryption, Inf. Sci. 206 (2012) 83–95, doi:10.1016/j.ins.2012.04.013.

[26] J. Shao, R. Lu, X. Lin, K. Liang, Secure bidirectional proxy re-encryption for cryptographic cloud storage, Pervasive Mob. Comput. 28 (2016) 113–121, doi:10.1016/j.pmcj.2015.06.016.

[27] D. Slamanig, R. Spreitzer, T. Unterluggauer, Adding controllable linkability to pairing-based group signatures for free, in: S.S.M. Chow, J. Camenisch, L.C.K. Hui, S.M. Yiu (Eds.), Information Security: 17th International Conference, ISC 2014, Hong Kong, China, October 12–14, 2014. Proceedings, Springer International Publishing, Cham, 2014, pp. 388–400, doi:10.1007/978-3-319-13257-0_23.

[28] Q. Tang, Towards public key encryption scheme supporting equality test with fine-grained authorization, in: U. Parampalli, P. Hawkes (Eds.), Proceedings of Information Security and Privacy: 16th Australasian Conference, ACISP 2011, LNCS, vol. 6812, Springer, Heidelberg, 2011, pp. 389–406.

[29] Q. Tang, Public key encryption supporting plaintext equality test and user-specified authorization, Sec. Commun. Netw. 5 (12) (2012) 1351–1362.

[30] H. Wang, Z. Cao, L. Wang, Multi-use and unidirectional identity-based proxy re-encryption schemes, Inf. Sci. 180 (20) (2010) 4042–4059, doi:10.1016/j.ins.2010.06.029.

[31] Y. Wang, Y. Ding, Q. Wu, Y. Wei, B. Qin, H. Wang, Privacy-preserving cloud-based road condition monitoring with source authentication in vanets, IEEE Trans. Inf. Forensics Secur. 14 (7) (2019) 1779–1790.

[32] Y. Wang, H. Pang, Probabilistic public key encryption for controlled equijoin in relational databases, Comput. J. 60 (4) (2017) 600–612, doi:10.1093/comjnl/bxw083.

[33] Y. Wang, H. Pang, R.H. Deng, Y. Ding, Q. Wu, B. Qin, Securing messaging services through efficient signcryption with designated equality test, Inf. Sci. 490 (2019) 146–165, doi:10.1016/j.ins.2019.03.039.
[34] Y. Wang, H. Pang, N.H. Tran, R.H. Deng, CCA Secure encryption supporting authorized equality test on ciphertexts in standard model and its applications, Inf. Sci. 414 (2017) 289–305, doi:10.1016/j.ins.2017.06.008.
[35] J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen, F. Bao, Cca-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles, Sci. China Inf. Sci. 53 (3) (2010) 593–606, doi:10.1007/s11432-010-0047-3.
[36] J. Weng, R.H. Deng, S. Liu, K. Chen, Chosen-ciphertext secure bidirectional proxy re-encryption schemes without pairings, Inf. Sci. 180 (24) (2010) 5077–5089, doi:10.1016/j.ins.2010.08.017.
[37] Z. Yan, W. Ding, X. Yu, H. Zhu, R.H. Deng, Deduplication on encrypted big data in cloud, IEEE Trans. Big Data 2 (2) (2016) 138–150, doi:10.1109/TBDATA.2016.2587659.
[38] Z. Yan, M. Wang, Y. Li, A.V. Vasilakos, Encrypted data management with deduplication in cloud computing, IEEE Cloud Comput. 3 (2) (2016) 28–35, doi:10.1109/MCC.2016.29.
[39] A. Yang, J. Xu, J. Weng, J. Zhou, D.S. Wong, Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage, IEEE Trans. Cloud Comput. (2018), doi:10.1109/TCC.2018.2851256.
[40] G. Yang, C.H. Tan, Q. Huang, D.S. Wong, Probabilistic public key encryption with equality test, in: J. Pieprzyk (Ed.), Topics in Cryptology - CT-RSA 2010, LNCS, vol. 5985, Springer, Heidelberg, 2010, pp. 119–131, doi:10.1007/978-3-642-11925-5_9.
[41] J. Zhang, Z. Zhang, Secure and efficient data-sharing in clouds, Concurrency Comput. 27 (8) (2015) 2125–2143, doi:10.1002/cpe.3395.
[42] Y. Zhou, H. Deng, Q. Wu, B. Qin, J. Liu, Y. Ding, Identity-based proxy re-encryption version 2, Future Gener. Comput. Syst. 62 (C) (2016) 128–139, doi:10.1016/j.future.2015.09.027.