Achieving key privacy without losing CCA security in proxy re-encryption

Jun Shao a,*, Peng Liu b, Yuan Zhou c

a College of Computer and Information Engineering, Zhejiang Gongshang University, PR China
b College of Information Sciences and Technology, Pennsylvania State University, United States
c National Network Emergency Response Technical Team/Coordination Center, PR China

1. Introduction

There are many applications requiring that the ciphertext under one public key can be transformed into another ciphertext under another public key with the same message. During the transformation process, the corresponding plaintext would not be revealed. Let’s consider the following scenario.

A group of members share an account of an outsourcing storage service, and every member can use this account to upload/download files to/from the server. In order to protect the secrecy of the files, the files should be encrypted before being sent to the server. When the owner of the files wants to share them to other group member, he/she needs to delegate his/her decryption rights on the encrypted files to the intended group member. Such delegation can be done via the server. That is, the server transforms the files encrypted under the owner’s public key to other files encrypted by the intended group member’s public key. However, the server cannot access the files. In other words, the outsourcing storage service does not require the full trustiness of the server.

To solve the above problem, Blaze et al. (1998) proposed the concept of proxy re-encryption (PRE). In such a scheme, a semi-trusted proxy with specific information (a.k.a., re-encryption key) can transform a ciphertext under Alice’s (delegator’s) public key into another ciphertext of the same plaintext under Bob’s (delegatee’s) public key. However, the proxy cannot learn anything about the plaintext. Two methods are given in Blaze et al. (1998) to classify PRE schemes. One method is according to the direction of transformation. If the re-encryption key allows the proxy to transform from Alice to Bob, and vice versa, the PRE scheme is bidirectional; otherwise, it is unidirectional. The other method is according to the times of transformation allowed. If the ciphertext can be transformed from Alice to Bob, then from Bob to Charlie, and so on, the PRE scheme is multi-use; otherwise, it is single-use.

According to the security requirements of applications, there are two security notions for PRE (Ateniese et al., 2005, 2006, 2009; Canetti and Hohenberger, 2007):

• Indistinguishability of encryptions: the adversary cannot get the plaintext, if he is not the intended receiver (including the delegator and delegatees). In the security model, an adversary cannot effectively distinguish between the encryption of two messages of his choosing. Like public key encryption (PKE), there are three levels of this indistinguishability, i.e., chosen-plaintext (CPA) security, replayable chosen-ciphertext (RCCA) security, and chosen-ciphertext (CCA) security (ordered by the adversary’s ability).

• Indistinguishability of keys (Key privacy): the adversary cannot identify the delegatee even if he holds the re-encryption key. In the security model, an adversary cannot effectively distinguish between the real re-encryption key of the challenge delegation (from an uncorrupted user to an uncorrupted user) of his choosing and a random key. All the applications of PRE can benefit from this property, i.e., the recipient of the ciphertext can keep his/her identity secret.
property is highly desirable for many encrypted communication scenarios (Ateniese et al., 2009).

To the best of our knowledge, though there are many PRE schemes proposed (Blaze et al., 1998; Ateniese et al., 2005, 2006, 2009; Canetti and Hohenberger, 2007; Libert and Vergnaud, 2008; Shao and Cao, 2009), no existing scheme holds CCA security and key privacy simultaneously in the standard model.

On the other hand, there are many applications requiring CCA-secure PRE scheme with key privacy. Let’s still consider the above outsourcing storage service. The following chosen ciphertext attack may be launched: an adversary might obtain a “decryption oracle” by faking encrypted files, sending them to the owner of the files, and then hoping that he/she responds with, “Did you share the following to me? [Decrypted attachment.]” Furthermore, in such an environment, it is also highly desired that the server cannot extract a list of “Who was sharing files privately with whom” or a list of “Who is using the re-encryption service”. Because these two lists could not only hurt the secrecy of users’ content, but also may hurt the secrecy of the content of files (the adversary may only focus on decrypting encrypted files related to a specified group member). The high-level description of the whole scenario is given in Fig. 1.

In this paper, we will propose the first such PRE scheme in the standard model, which is an open problem left by Ateniese et al. (2009). Our construction demands a new Diffie–Hellman related intractability assumption in bilinear map groups. Note that our proposal is single-use, while it is enough for the above encrypted email setting.

1.1. Related work

Besides the encrypted email forwarding, PRE can be used in many applications, including simplification of key distribution (Blaze et al., 1998), distributed file systems (Ateniese et al., 2005, 2006), security in publish/subscribe systems (Khurana and Koleva, 2006), multicast (Chiu et al., 2005), secure certified email mailing lists (Khurana et al., 2005; Khurana and Hahm, 2006), interoperable architecture of DRM (Taban et al., 2006), access control (Talmy and Dobzinski, 2006), and privacy for public transportation (Heydt-Benjamin et al., 2005). Hence, since the introduction of proxy re-encryption by Blaze et al. (1998), there have been many papers (Blaze et al., 1998; Ateniese et al., 2005, 2006, 2009; Green and Ateniese, 2007; Hohenberger et al., 2007; Canetti and Hohenberger, 2007; Weng et al., 2008; Shao and Cao, 2009) that have proposed different PRE schemes with different security properties.

The first CPA-secure PRE scheme was proposed by Blaze et al. (1998) based on ElGamal encryption (ElGamal, 1985). Later, by using key sharing technique, Green and Ateniese (2007), and Weng et al. (2008) proposed two efficient single-use unidirectional PRE schemes, respectively. The first CCA-secure multi-use bidirectional PRE scheme in the standard model was proposed by Canetti and Hohenberger (2007).

Nevertheless, none of the above schemes are collusion resistant. Based on public key encryption with double trapdoors (strong and weak private keys), Ateniese et al. (2005, 2006) proposed the first collusion resistant PRE schemes. However, their schemes are only CPA-secure. Recently, Libert and Vergnaud (2008) proposed the first RCCA-secure and collusion resistant PRE scheme in the standard model, and Shao and Cao (2009), and Chow et al. (2010) proposed CCA-secure and collusion resistant PRE schemes in the random oracle model. These three schemes are all single-use and unidirectional.

However, as mentioned by Ateniese et al. (2009), none of the above schemes is key-private. Based on the scheme in (Ateniese

---

Footnote:

1 Collusion resistance means that the delegatee (delegator) cannot get the private key of the delegator (delegatee) by colluding with the proxy.
et al., 2005, 2006), Ateniese et al. (2009) proposed the first key-private PRE scheme. However, their scheme is only CPA-secure. And they left how to achieve key privacy without losing CCA security in proxy re-encryption as an open problem.

In this paper, we attempt to take this challenge.

1.2. Our contribution

In this paper, we propose the first CCA-secure and key-private PRE scheme in the standard model, which is an open problem left by Ateniese et al. (2009). Furthermore, the key privacy of our proposal is proved in a revised security model, where multiple re-encryption keys can correspond to one delegation. As mentioned by Ateniese et al. (2009), Ateniese et al.’s scheme is not key-private in the revised security model.

In the rest of the paper, we first introduce some basic knowledge we use in this paper. In Section 3, we propose our construction and give the security proofs. Finally, we draw the conclusion.

2. Definitions and security models

In this section, we review some basic knowledge we will use later, including the definitions and security models for one-time symmetric key encryption (SKE), one-time signature (SIG), and proxy re-encryption (PRE), and the definitions of the 5-Extended Decision Bilinear Diffie–Hellman (5-EDBDH) assumption and the Decision Diffie–Hellman (DDH) assumption.

2.1. One-time symmetric-key encryption

Definition 1. A one-time symmetric-key encryption scheme SKE contains three algorithms SKE.KeyGen, SKE.Enc and SKE.Dec (Cramer and Shoup, 2003; Kurosawa and Desmedt, 2004).

- SKE.KeyGen(1^λ) → k. On input the security parameter λ, SKE.KeyGen outputs a key k.
- SKE.Enc(k, m) → C. On input a key k and a message m, SKE.Enc outputs a ciphertext C.
- SKE.Dec(k, C) → m. On input a key k and a ciphertext C, SKE.Dec outputs a message m or the special symbol ⊥.

Correctness. The correctness property is that for any message m in the message space and any key k ← SKE.KeyGen(1^λ), the following condition must hold: SKE.Dec(k, SKE.Enc(k, m)) = m.

2.1.1. An example of CCA-secure SKE

In this paper, we will use the following CCA-secure SKE, which is originally from Cramer and Shoup (2003).

- SKE.KeyGen: Choose an ℓ-bit string k.
- SKE.Enc: To encrypt a message m, the encryptor does the following steps.
  - Compute f = PRBG(k|0), where PRBG is a pseudo-random bit generator which stretches (ℓ + 1)-bit strings to strings of |m|-bit length.
  - Compute e = f ⊕ m and a = AXUH(1, e), where AXUH is a hash function, suitable for message authentication, which is keyed by an (ℓ + 1)-bit string and hashes arbitrary bit string to ℓ-bit strings.
  - Output the ciphertext C = (e, a).
- SKE.Dec: To decrypt a message m, the decryptor uses k to do the following steps. Check if a = AXUH(k|1, e). If it does not hold, then output ⊥; otherwise, output m by m = e ⊕ PRBG(k|0).

According to the result in Cramer and Shoup (2003) and Kurosawa and Desmedt (2004), we have the following theorem.

Theorem 1. The above SKE is CCA-secure, if AXUH and PRBG are secure.

Lemma 1. If AXUH is collision-resistant, then for a given C = (e, a), the probability to find two different keys k₀, k₁ such that SKE.Dec(k₀, C) ̸= ⊥(b ∈ {0, 1}) is negligible.

Proof. Assume that there are two different secret keys, say k₀, k₁, to make SKE.Dec(k, C) ̸= ⊥, then we have that AXUH[k₀]\|1, e) = AXUH[k₁]\|1, e) = a.

It is easy to see that the above equation violates the assumption that AXUH is collision-resistant.

Hence, we obtain this lemma. □

2.2. One-time signature scheme

Definition 2. A one-time signature scheme SIG contains three algorithms SIG, SIG.S and SIG.V (Canetti et al., 2004).

- SIG.Γ(1^λ) → (svk, ssk). On input the security parameter λ, SIG.Γ outputs a key pair (svk, ssk).
- SIG.(ssk, m) → S. On input a signing key ssk and a message m, SIG.S outputs a signature S on m.
- SIG.V(svk, m, S) → 1 or 0. On input a verifying key svk, a message m and a signature S, SIG.V outputs 1 if S is a signature of m under svk; otherwise, it outputs 0.

Correctness. The correctness property is that for any message m ≠ m’ in the message space and any key pair (svk, ssk) ← SIG.Γ(1^λ), the following conditions must hold:
SIG.V(svk, m, SIG.(ssk, m)) = 1, and SIG.V(svk, m, SIG.(ssk, m’)) = 0.

2.3. Single-use unidirectional proxy re-encryption

2.3.1. Definition of single-use unidirectional proxy re-encryption

Definition 3. A single-use unidirectional proxy re-encryption scheme PRE is a tuple of p.p.t. algorithms (PRE.KeyGen, PRE.Enc, PRE.Dec, PRE.ReEnc, PRE.ReKeyGen).

- PRE.KeyGen, PRE.Enc, PRE.Dec. Identical to those in the traditional public key encryption.
- PRE.ReKeyGen(sk₁, pk₂) → rkpk₁, pk₂. On input a private key sk₁ and a public key pk₂, the re-encryption key generation algorithm PRE.ReKeyGen outputs a unidirectional re-encryption key rkpk₁, pk₂.
- PRE.ReEnc(rkpk₁, pk₂, C₁) → C₂. On input a re-encryption key rkpk₁, pk₂ and a ciphertext C₁, the re-encryption algorithm PRE.ReEnc outputs a re-encrypted ciphertext C₂ or ⊥.

Correctness. A correct single-use unidirectional proxy re-encryption scheme should satisfy the following two requirements: PRE.Dec(skk, PRE.Enc(pk, m)) = m, and PRE.Dec(skk, PRE.ReEnc(PRE.ReKeyGen(skk, pk₂), C₁)) = m, where (pk, skk), (pk’, skk’) ← PRE.KeyGen(1^λ), and C is the ciphertext of message m under pk from algorithm PRE.Enc.

Remark 1 (Two types of ciphertexts). In almost all existing single-use unidirectional proxy re-encryption schemes, there are two types of ciphertexts: the first-level ciphertext, which cannot be re-encrypted and can be only decrypted by the delegator; and the second-level ciphertext, which can be re-encrypted into the first-level ciphertext and can be decrypted by the delegator and the
collaboration of the proxy and the delegatee. In this paper, we call the second-level and first level ciphertext as original and re-encrypted ciphertext, respectively.

### 2.3.2. Indistinguishability of encryptions under chosen-ciphertext attack for single-use unidirectional proxy re-encryption

The IE-CCA security for single-use unidirectional PRE is defined by the following chosen-ciphertext attack game played between a challenger \( C \) and an adversary \( A \). Note that we work in the static corruption model, where the adversary must decide the corrupted users before the game starts. Furthermore, we assume that the public keys input into the oracles by the adversary are all from \( \mathcal{O}_{pk} \). Since we have two types of ciphersystems, there are two situations.

#### 2.3.2.1. The challenge original ciphertext. Phase 1:

The adversary \( A \) issues queries \( q_1, \ldots, q_n \), where query \( q_i \) is one of:

- **Public key generation oracle \( \mathcal{O}_{pk} \):** On input an index \( i \), the challenger takes a security parameter \( \lambda \) and responds by running algorithm \( \text{PRE. KeyGen}(1^\lambda) \) to generate a key pair \((pk_i, sk_i)\), gives \( pk_i \) to \( A \) and records \((pk_i, sk_i)\) in table \( T_x \).

- **Secret key generation oracle \( \mathcal{O}_{sk} \):** On input \( pk \) by \( A \), if \( pk \) is corrupted, the challenger searches \( pk \) in the table \( T_x \) and returns \( sk \); otherwise, the challenger returns \( \bot \).

- **Re-encryption key generation oracle \( \mathcal{O}_{rk} \):** On input \((pk, pk', C)\) by \( A \), the challenger returns the re-encryption key \( rk_{pk,pk'} = \text{PRE.ReKeyGen}(sk, pk') \), where \( sk \) is the private key corresponding to \( pk \).

- **Re-encryption oracle \( \mathcal{O}_{re} \):** On input \((pk, pk', C)\) by \( A \), the re-encrypted ciphertext \( C' = \text{PRE.ReEnc}(\text{PRE.ReKeyGen}(sk, pk'), C) \) is returned by the challenger, where \( sk \) is the private key corresponding to \( pk \).

- **Decryption oracle \( \mathcal{O}_{dec} \):** On input \((pk, C)\), the challenger returns \( \text{PRE.Dec}(sk, C) \), where \( sk \) is the private key corresponding to \( pk \).

These queries may be asked adaptively, that is, each query \( q_i \) may depend on the replies to \( q_1, \ldots, q_{i-1} \).

**Challenge:** Once the adversary \( A \) decides that Phase 1 is over, it outputs two equal length plaintexts \( m_0, m_1 \) from the message space, and a public key \( pk \) on which it wishes to challenge. There are two restrictions on the public key \( pk \): (i) \( pk \) is an uncorrupted public key; (ii) if \( pk \) is uncorrupted, \( C' = \text{PRE.Enc}(pk, m) \) holds for \( m \neq \bot \).

**Phase 2:** The adversary \( A \) issues more queries \( q_{n+1}, \ldots, q_n \) where query \( q_n \) is one of:

- **\( \mathcal{O}_{rk} \), \( \mathcal{O}_{sk} \):** The challenger responds as in Phase 1.

- **\( \mathcal{O}_{re} \):** on input \((pk, pk', C)\) by \( A \), if \( pk = pk' \) and \( pk \) is a corrupted public key, the challenger outputs \( \bot \); otherwise, the challenger responds as in Phase 1.

- **\( \mathcal{O}_{re} \):** on input \((pk, pk', C)\) by \( A \), if \((pk, C) = (pk', C')\), and \( pk \) is a corrupted public key, the challenger outputs \( \bot \); otherwise, the challenger responds as in Phase 1.

- **\( \mathcal{O}_{dec} \):** On input \((pk, C)\), if \((pk, C) \) is not a derivative\(^3\) of \((pk', C')\), the challenger outputs \( \bot \); otherwise, the challenger responds as in Phase 1.

These queries may be also asked adaptively.

**Guess:** Finally, the adversary \( A \) outputs a guess \( b' \in \{0, 1\} \) and wins the game if \( b = b' \).

The advantage \( \text{Adv}_{\text{PRE}}^{\text{IE-CCA-0}}(\lambda) \) is defined as \( |\text{Pr}[b = b'] - 1/2| \).

The scheme \( \text{PRE} \) is said to be IE-CCA-0 secure if all efficient adversaries \( A \), the advantage \( \text{Adv}^{\text{IE-CCA-0}}_{\text{PRE}}(\lambda) \) is negligible.

#### 2.3.2.2. The challenge re-encrypted ciphertext. Phase 1:

Identical to that in the challenge original ciphertext case.

**Challenge:** Once the adversary \( A \) decides that Phase 1 is over, it outputs two equal length plaintexts \( m_0, m_1 \) from the message space, a public key \( pk \) and an uncorrupted public key \( pk' \) on which it wishes to challenge. \( C' = \text{PRE.ReEnc}(rk, \text{PRE.Enc}(pk, m)) \), where \( rk \) is a re-encryption key from \( pk \) to \( pk' \). It sends \( C' \) as the challenge to \( A \).

**Phase 2:** Almost the same as that in Phase 1, except the following.

- **\( \mathcal{O}_{dec} \):** On input \((pk, C)\), if \((pk, C) = (pk', C')\), the challenger outputs \( \bot \); otherwise, the challenger responds as in Phase 1.

**Guess:** Identical to that in the challenge original ciphertext case.

The advantage \( \text{Adv}^{\text{IE-CCA-0}}_{\text{PRE}}(\lambda) \) is defined as \( |\text{Pr}[b = b'] - 1/2| \).

The scheme \( \text{PRE} \) is said to be IE-CCA-0 secure if all efficient adversaries \( A \), the advantage \( \text{Adv}^{\text{IE-CCA-0}}_{\text{PRE}}(\lambda) \) is negligible.

**Remark 2.** As mentioned by Libert and Vergnaud (2008), the security of collusion resistance is implied by the IE-CCA-R security.

#### 2.3.3. Indistinguishability of keys under chosen-ciphertext attack for single-use unidirectional proxy re-encryption

The IK-CCA security for single-use unidirectional PRE is defined by the same method of the IE-CCA security for single-use unidirectional PRE.

**Phase 1:** There exist public key generation oracle \( \mathcal{O}_{pk} \), private key generation oracle \( \mathcal{O}_{sk} \), re-encryption key generation oracle \( \mathcal{O}_{rk} \), and decryption oracle \( \mathcal{O}_{dec} \) as those in Phase 1 of the IE-CCA-O game for single-use unidirectional PRE. There is no re-encryption oracle in the IK-CCA-O game for single-use unidirectional PRE, since the adversary can obtain any re-encryption key he wants, and he can use the obtained re-encryption key to generate the corresponding re-encrypted ciphertexts.

**Challenge:** Once the adversary \( A \) decides that Phase 1 is over, it outputs two public keys \( pk_i, pk_j \) on which it wishes to challenge, where \( pk_i \) and \( pk_j \) are two uncorrupted public keys. The challenger picks a random bit \( b \in \{0, 1\} \). If \( b = 0 \), then it sets \( rk_{pk_i, pk_j} \) as a random key from the re-encryption key space; otherwise, it sets \( rk_{pk_i, pk_j} = \text{PRE.ReKeyGen}(sk_i, pk_j) \), where \( sk_i \) is the corresponding private key of \( pk_i \). At last, the challenger sends \( rk_{pk_i, pk_j} \) as the challenge to \( A \).

**Phase 2:** It runs almost the same as that in Phase 1, but with the following restrictions.

\(^1\) \((pk', C')\) is a derivative of itself.

\(^2\) If \( A \) has queried \( \mathcal{O}_{sk} \) on input \((pk, pk', C)\) and obtained \((pk', C')\), then \((pk', C')\) is a derivative of \((pk, C)\).

\(^3\) If \( A \) has queried \( \mathcal{O}_{sk} \) on input \((pk, pk', C)\), and \( C' = \text{PRE.ReEnc}(\mathcal{O}_{sk}(pk, pk'), C) \), then \((pk', C')\) is a derivative of \((pk, C)\).
Let our re-encryption previous re-encrypted key.

\[ C \]

2.4.1. The advantage \( Adv^{\text{pre}}_{\text{pre}}(\lambda) \) is defined as \( |\Pr[b = b'] - 1/2| \). The scheme PRE is said to be IK-CCA secure if all efficient adversaries \( A \), the advantage \( Adv^{\text{pre}}_{\text{pre}}(\lambda) \) is negligible.

Remark 3 (Restrictions). The restrictions in \( O_{\text{dec}} \) in Phase 2 of the IK-CCA game for single-use unidirectional PRE are reasonable, since if not, the adversary can trivially decide whether the challenge re-encryption key is a random value or a real re-encryption key as follows. The adversary first encrypts a message \( m \) with \( pk \) to get a ciphertext \( C \), and then it re-encrypts \( C \) with the challenge re-encryption key to get another ciphertext \( C' \). At last, the adversary queries the decryption oracle with \( (pk, C') \). If the result message equals to \( m \), then the challenge re-encryption key is a real one; otherwise, it is a random value.

The other question on the restrictions is whether it is too weak, since it is not easy for the decryptor to decide whether a re-encrypted ciphertext is computed by a specific re-encryption key. However, it is possible to make the decryptor have the ability: If every re-encrypted ciphertext computed from the same re-encryption key contains the same value, which can be obtained by the decryptor but not the proxy.

Remark 4 (CPA security, differences between ours and previous one). If we remove the decryption oracle from the IK-CCA game for single-use unidirectional PRE, we can get the IK-CPA game for single-use unidirectional PRE. Compared to the IK-CPA game by Ateniese et al. (2009), our IK-CPA game has two differences.

- Our IK-CPA game allows multiple re-encryption keys per one delegation, while it allows only one re-encryption key per one delegation in Ateniese et al. (2009).
- Our IK-CPA game allows the adversary to get the re-encryption key of any delegation he wants, including the re-encryption key of the delegation corresponding to the challenge re-encryption key, while it disallows this query in Ateniese et al. (2009).

2.4. Bilinear forms

2.4.1. Bilinear groups

1. \( G \) and \( G_r \) are two (multiplicative) cyclic groups of prime order \( p \);
2. \( g \) is a generator of \( G \);
3. \( e \) is a bilinear map \( e : G \times G \to G_r \).

Let \( G \) and \( G_r \) be two groups as above. An admissible bilinear map is a map \( e : G \times G \to G_r \) with the following properties:

1. \( \text{Bilinearity:} \) For all \( P, Q, R \in G \), \( e(P, Q, R) = e(P, Q) \cdot e(Q, R) \) and \( e(P, Q, R) = e(P, R) \cdot e(P, Q) \).
2. \( \text{Non-degeneracy:} \) If \( e(P, Q) = 1 \) for all \( Q \in G \), then \( P = O \), where \( O \) is a point at infinity.

We say that \( G \) is a bilinear group if the group action in \( G \) can be computed efficiently and there exists a group \( G_r \) and an efficiently computable bilinear map as above. We denote \( \text{BSetup} \) as an algorithm that, on input the security parameter \( 1^{\lambda} \), outputs the parameters for a bilinear map as \( (p, g, G, G_r, e) \), where \( p \in \Theta(2^{\lambda}) \).

2.4.2. 5-Extended Decision Bilinear Diffie–Hellman (5-EDBDH) assumption

Let \( \text{BSetup}(1^{\lambda}) \rightarrow (p, g, G, G_r, e) \), where \( \langle g \rangle = G \). For all p.p.t. adversaries \( A \), there exists a negligible function \( \varepsilon \) such that the following probability is less than or equal to \( 1/2 + \varepsilon(\lambda) \):

\[
\Pr[a, b, c, d \leftarrow Z_p^r; x_1 \leftarrow e(g, g)^{ad}; x_0 \leftarrow e(g, g)^{cd}; z \leftarrow \{0, 1\}; z' \leftarrow A(g, g^a, g^b, g^c, g^{a^2}, g^{c^2}, \langle g \rangle, x_1, x_0) : z = z']
\]

The input of the 5-EDBDH problem does not contain \( g^{a^2}, g^{c^2} \) or \( g^b, g^c \); hence, the bilinear map cannot be used to solve the problem. In fact, the 5-EDBDH problem is a specific case of the decision \( (P, Q, f) \)-Diffie–Hellman problem, which was studied and proven-hard in Chapter 2 of Goh (2007).

Note that compared to the DBDH problem, the new problem has 5 additional inputs \( (g^{a^2}, g^{c^2}, g^b, g^c, g^{a^2}, g^{c^2}) \), hence we name the new problem as 5-EDBDH.

In the security proofs (see Section 3.3), \( (g^{a^2}, g^{c^2}, g^b, g^c) \) are used to simulate the re-encryption key generation, \( g^b \) is used to simulate \( b' \), and \( g^{a^2} \) is used to simulate \( x_1 \).

2.4.3. Decision Diffie–Hellman (DDH) assumption

Let \( \text{BSetup}(1^{\lambda}) \rightarrow (p, g, G, G_r, e) \), where \( \langle P \rangle = G_r \). For all p.p.t. adversaries \( A \), there exists a negligible function \( \varepsilon \) such that the following probability is less than or equal to \( 1/2 + \varepsilon(\lambda) \):

\[
\Pr[a, b, c, d \leftarrow Z_p^r; x_0 \leftarrow \langle P \rangle^i; x_1 \leftarrow \langle P \rangle^{i^2}; i = 1, 2, \ldots; \varepsilon \geq 2; z \leftarrow \{0, 1\}; z' \leftarrow A(P, P^a, \ldots, P^a, x_1, \ldots, x_0) : z = z']
\]

3. Our proposal

In this section, we first propose a new PRE scheme, and then prove its CCA security and key privacy in the standard model one by one.

3.1. Intuition behind the construction

To achieve key privacy without losing CCA security in proxy re-encryption is not an easy task, since the CPA-to-CCA translations, such as the Fujisaki–Okamoto transformation (Fujisaki and Okamoto, 1999), cannot be applied directly (Ateniese et al., 2009). In this paper, to solve the problem, we start from the following observations by Ateniese et al. (2009).

- The re-encryption key generation algorithm of any key-private PRE scheme should be probabilistic and key-private, if it allows multiple re-encryption keys corresponding to one delegation.
- The re-encryption algorithm of any key-private PRE scheme should be probabilistic.

The re-encryption key in our proposal is generated by the similar method in Ateniese et al. (2005, 2006), and Libert and Vergnaud (2008), which is deterministic. To transform the re-encryption key generation algorithm from deterministic to probabilistic, we introduce a random number into the re-encryption key generation algorithm. That is, one component in the re-encryption key is \( g^{(x+y)r} \), where \( r \) is a random number, and \( x \) and \( y \) are the private keys of delegate and delegatee, respectively. This random number \( r \) makes the re-encryption key generation algorithm probabilistic. Furthermore, \( r \) is encrypted by a key-private public key encryption scheme with the delegatee's
public key. Hence, from $g^{(x/y)} \cdot r$, r’s ciphertext), the adversary cannot identify the delegatee or the delegator. At last, besides $g^{(x/y)} \cdot r$, r’s ciphertext), the re-encryption key contains a delegatee’s “pseudo” public key, which will be explained soon.

To make the re-encryption algorithm probabilistic, the proxy first uses $g^{(x/y)} \cdot r$ to process the original ciphertext, and then uses the key-private public key encryption scheme to encrypt (r’s ciphertext and the processed original ciphertext) with a delegatee’s “pseudo” public key. The resulting ciphertext is treated as the re-encrypted ciphertext.

The key-private public key encryption scheme used in the re-encryption key generation algorithm and the re-encryption algorithm is Cramer–Shoup scheme (Cramer and Shoup, 1998).

Following the ideas in Canetti and Hohenberger (2007) and Shao and Cao (2009), we obtain the CCA security by providing public validity check of original ciphertexts and validity check of re-encrypted ciphertexts. In this paper, we make use of strongly unforgeable one-time signature and CCA-secure one-time symmetric key encryption to achieve these two validity checks.

3.1.1. “Pseudo” public key

The “pseudo” public key plays an important role in our proposal. It is not the real public key, but the private key can be used to decrypt all ciphertexts under one of these two public keys. For example, in ElGamal encryption (ElGamal, 1985), the real public key is $(g, g^x)$, the “pseudo” public key could be $(h, h^x)$, where $g, h$ is random numbers in the underlying group, and $x$ can be used to decrypt all ciphertexts under $(g, g^x)$ and $(h, h^x)$. Due to the DDH assumption, given $(g, g^x, h, h^x)$, it is hard to decide $\log g^x = \log h^x$.

For different re-encryption keys, it requires different delegatee’s “pseudo” public keys for different proxies. If not, the adversary can easily check the challenge re-encryption key’s validity by comparing the “pseudo” public key in the challenge re-encryption key with that in the real re-encryption key. Note that we allow multiple re-encryption keys are associated with a delegation, and allow the adversary to obtain a real re-encryption key corresponding to the challenge delegation.

Moreover, $(g^{(x/y)} \cdot r$, r’s ciphertext) and the “pseudo” public key should be bound tightly. If not, the adversary can check the challenge re-encryption key’s validity by the following method. Get a re-encrypted ciphertext by using the “pseudo” public key and another $(g^{(x/y)} \cdot r$, r’s ciphertext), and check whether the plaintext is equal to the output of the decryption oracle. To solve the problem, we insert r into the “pseudo” public key.

With the above ideas, we propose the first PRE scheme which achieves key privacy without losing CCA security. We prove the two security notions (CCA security and key privacy) in the standard model based on the 5-Extended Decision Bilinear Diffie–Hellman assumption and Decision Diffie–Hellman assumption, respectively.

3.2. The description of our proposal

**PRE.Setup**: The system parameters are

$$(g, h, h^x, g_2, g_3, P_1, P_2, p, e, G, G_r, SIG, SKE, F(\cdot), F(\cdot), F(\cdot), H_1(\cdot), H_2(\cdot), H_3(\cdot))$$

where $(g, p, G, G_r, e) \leftarrow $ BSetup($\lambda$), $(h, h^x, h^y)$ are random numbers from $G$, $P_1$ and $P_2$ are random numbers from $G_r$, $SIG = (G, S, \{\})$ is a strongly unforgeable one-time signature scheme, $SKE$ is a CCA-secure one-time symmetric key encryption scheme (the scheme in Section 2.1.1), $F(\cdot), F(\cdot), F(\cdot)$ are pseudo-random bit generators (PRBG), and $H_1(\cdot) (i = 1, 2, 3), H_2(\cdot) (i = 1, 2), H_3(\cdot) (i = 1, 2)$ are hash functions,

$$F(\cdot) : (0, 1)^* \rightarrow (0, 1)^*$$

$$F(\cdot) : (0, 1)^n \rightarrow G \times G \times C \times G_r^* \times S \times G$$

We denote $\ell_1$ and $\ell_2$ as the bit lengths of the private key of SKE and the verifying key of SIG, respectively. $C$ is the ciphertext space of SKE. According to the results in Shi et al. (2007) and Kurosawa and Desmedt (2004), p, $\ell_1$ and $\ell_2$ are 160-bit, each element in $G$ and $G_r$ needs 512 bit storage, and the bit length of ciphertext of SKE will be the bit length of message plus 160bit.

**PRE.KeyGen**: The user sets his public key as

$pk = (X_1, X_2, Z_1, Z_2, Z_3) = (g^x, h^{1/2}, P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2)$

for random

$sk = (x, x_1, x_2, x_3, x_4, x_5) \in Z_p^6$.

If the values $(X_2, Z_1, Z_2, Z_3)$ are presented, which signifies that the user is willing to accept delegations.

**PRE.ReKeyGen**: On input a public key $pk = (X_1, X_2, Z_1, Z_2, Z_3) = (g^y, h^{1/2}, P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2)$, and a private key $sk = (x, x_1, X_2, x_3, x_4, x_5)$, it outputs a unidirectional re-encryption key

$rk_{pk, sk} = (rk_{sk, pk}^1, rk_{sk, pk}^2, rk_{sk, pk}^3, rk_{sk, pk}^4, rk_{sk, pk}^5, rk_{sk, pk}^6)$

which is computed as follows.

1. Choose random numbers $r, X, Y \in Z_p$.
2. Set $rk_{sk, pk}^1 = (X_1)^{x^r} \cdot g^{y^r}$.
3. Set $rk_{sk, pk}^2 = (Q_1, Q_2, T_1, T_2, T_3) = (p_1^2, p_2^2, Z_1^2, Z_2^2, Z_3^2)$.
4. Compute $rk_{sk, pk}^3 = (A, B, C, D, E)$ as follows.

$\bar{A} = p_1^2, B = p_1^2, C = p_2^2, D = p_2^2, E = p_2^2, Z = (Z_1^2, Z_2^2, Z_3^2)$

Note that $(A, B, C, D, E)$ is computed almost the same as that in Cramer–Shoup scheme with the public key $(P_1, P_2, Z_1^2, Z_2^2, Z_3^2)$, except that $C$ is computed as $F((Z_1^2) \cdot E \cdot m)$ instead of $(Z_1^2) \cdot m$.

**PRE.Enc**: On input a public key $pk = (X_1, X_2, Z_1, Z_2, Z_3) = (g^x, h^{1/2}, P_1, P_2, P_1, P_2, P_1, P_2, P_1, P_2)$ and a message $m \in \{0, 1\}^n$, the encryptor does the following steps:

1. Choose a random number $k \in Z_p$.
2. Run the key generation algorithm $SIG = (G, \{\})$ to get a signing and verifying key pair $(ssk, svk)$.
3. Set

$A = svk, B = g^x, C = g^{(x+y)} \cdot g_3^y, D = h^{1/2}, E = SKE.Enc(F(A) || D || (C) || (X_1, h^y)), S = SIG.Ssk((B, C, D, E))$.

4. Output the ciphertext $K = (A, B, C, D, E, S)$.

**PRE.ReEnc**: On input a re-encryption key $rk_{sk, sk} = (rk_{sk, pk}^1, rk_{sk, pk}^2, rk_{sk, pk}^3, rk_{sk, pk}^4, rk_{sk, pk}^5, rk_{sk, pk}^6)$ and a ciphertext $K =$
\((A, B, C, D, E, S)\) for public key \(pk_1\). The proxy does the following steps.

- **If** \(e(B, g^{h(A)}) = e(g, C), e(B, h') = e(g, D), \) and \(\text{SIG}.V(A, (B, C, D, E), S) = 1\) all hold, go to the next step; otherwise, output \(\perp\) and halts.
- Choose a random number \(k \leftarrow Z_p\).
- Set \(c = \text{A}||\text{C}||\text{D}||e(\text{A}||\text{B}||\text{C}||\text{D}).\)
- Compute
  \[
  B' = e(B, r_{pk_1}^{(1)}), \quad A = Q_1 k, \quad B = Q_2 k, \\
  \hat{C} = F(T_3 k) \oplus (B' || c), \quad \hat{a} = H_5(A||B||C||D), \quad \hat{D} = T_1 k_1 + k_2 a.
  \]
- Note that \((A, \hat{B}, \hat{C}, \hat{D})\) is computed almost the same as that in Cramer–Shoup scheme with the public key \((P_1, P_2, (Z_1, H_2^{(r)}), (Z_2, H_2^{(r)}), Z_3)\), except that \(\hat{C}\) is computed as \(F(T_3 k) \oplus m\) instead of \(T_3 k \cdot m\).
- Output the re-encrypted ciphertext \(\hat{K} = (A, \hat{B}, \hat{C}, \hat{D}).\)

**Theorem 2.** Our proposal is IE-CCA-O secure in the standard model under assumptions that the 5-EDDBH problem and DDH problem are hard, that \(\text{SKE}\) is CCA-secure, that \(\text{SIG}\) is strongly unforgeable, that \(H_1, H_2, H_3\) are target collision resistant, and that \(F, \hat{f}, \hat{F}\) are secure PRBCs. In particular, we have that

\[
\text{Adv}_{\text{IE-CCA-O}}(\lambda) \leq q_{pk}^{-1} \cdot \left( \epsilon_{5-EDDBH} + \epsilon_{SIG} + 2\epsilon_{\text{CA-SKE}} + 2\epsilon_{\text{PRBC}} \right)
\]

\[
+ q_{pd}^{-1} \cdot \left\langle \epsilon_{\text{Re-Dec}} + \epsilon_{\text{Dec}} \right\rangle \cdot \left( \frac{4 + q_{ed}^{(1)} + q_{ed}}{q} + \epsilon_{\text{Dec}} \right) + q_{re} \cdot \left( \epsilon_{\text{Re-Dec}}^{(1)} + \epsilon_{\text{PRBC}}^{(1)} \right) \]

where \(q_{pk}\) is the amount of queries to the public key generation oracle for uncorrupted public keys, \(q_{ed}\) is the amount of queries to the re-encryption oracle and the decryption oracle, \(q_{re}\) is the amount of queries to the re-encryption oracle and the re-encryption key generation oracle, \(q_{ed}\) is the amount of queries to the re-encryption oracle, \(q_{pd}\) is the amount of queries to the re-encryption key generation oracle, \(\epsilon_{5-EDDBH}\) is the probability that the adversary solves the 5-EDDBH problem, \(\epsilon_{\text{SIG}}\) is the probability that the adversary breaks the strong unforgeability security of \(\text{SIG}\), \(\hat{b}\) is the probability that any given verification key is output by \(\text{SIG}\), \(\epsilon_{\text{Re-Dec}}^{(1)} + \epsilon_{\text{Dec}}^{(1)}\) are the probabilities that the adversary breaks the target collision resistance of \(H_1, H_2, H_3\), \(\epsilon_{\text{SKE}}\) is the probability that the adversary breaks the CCA security of \(\text{SKE}\), and \(\epsilon_{\text{PRBC}}, \epsilon_{\text{PRBC}}^{(1)}\) are the probabilities that the adversary breaks the randomness of \(F, \hat{f}, \hat{F}\), respectively.

**Proof.** We incrementally define a sequence of games starting at the real attack (Game \(G_0\)), and ending up at Game \(G_0\), which clearly shows that the adversary cannot break the system. We define \(E\) to be the event that \(b = b'\) in Game \(G_0\), where \(b\) is the bit involved in the challenge phase, and \(b'\) is the output of \(A\) in the guess phase.

**Game \(G_0\).** This game corresponds to the real attack (the IE-CCA-O game for single-use unidirectional PRE). By definition,

\[
Pr[E_0] = \frac{1}{2} = \text{Adv}_{\text{IE-CCA-O}}(\lambda)\]

**Game \(G_1\).** In this game, we modify the public key oracle and the challenge phase as follows.

- **\(\hat{c}_{pk}\):** On input an index \(i\), if it is an uncorrupted public key, the challenger decides whether it is the challenge public key. If yes, the challenger marks the public key as \(\hat{pk}\). The other steps in this oracle are the same as those in Game \(G_0\).

- **Challenge:** On input \((pk', m_0, m_1)\) by the adversary, if \(pk' \neq \hat{pk}\), the challenger reports “failure” and aborts; otherwise, the challenger does the same steps as those in Game \(G_0\).

It is easy to see that only if the challenger guesses the correct challenge public key, Game \(G_1\) and Game \(G_0\) are indistinguishable. The probability that the challenger guesses the correct challenge public key is \(1/q_{pk}\) at least. Hence, we have

\[
Pr[E_1] \cdot \frac{1}{2} \geq \frac{1}{q_{pk}} \cdot Pr[E_0] \cdot \frac{1}{2}
\]

**Game \(G_2\).** In this game, given a 5-EDDBH input \((g, g^{h'}, g^{h'}, g^{h'}, g^{h'}, g^{h'}, g^{h'}, g^{h'}, \alpha, \beta)\), we modify the setup phase and the challenge phase as follows.

**Setup:** Set \(h = g^{\alpha}, h' = g^{\alpha''}, h'' = g^{\alpha''}, g_2 = g^{\beta}, \) and \(g_3 = (g^{1/2} g_2)/(g^{1/2} g_2)\), \(H_{\text{ske}}(skk') = h_{\text{ske}}(\beta, g_2, H_2(\text{ske})), \) where \(\alpha, \beta, \) and \(\omega\) are three random numbers from \(Z_p\) and \(\text{SIG}(A^r) \rightarrow (skk', svk')\). The other steps are the same as that in Game \(G_1\).

**Challenge:** The challenger sets \(A^r = svk'\), and computes \(S\) by using \(skk'\). The other steps are the same as that in Game \(G_1\).
Since \((g, g^a, g^b, g^c, g^{x_a}, g^{x_b}, g^{x_c}, g^{x_D}, \alpha_1, \alpha_2, w)\) are random, the above setup oracle and the challenge phase are the same as that in Game G₁. Hence, we have that

\[
\Pr[E₂] = \Pr[E₁] \tag{3}
\]

**Game G₂.** In this game, we modify the public key generation oracle, the re-encryption oracle, the decryption oracle and the challenge phase as follows.

**Phase 1:**

- **O\(_{pk}\):** the challenger chooses random numbers \((X₁₁, X₁₂, X₂₁, X₂₂, X₃₄, X₅₃)\) from \(Z_p^*\) and sets \(Z₁ = (P₁)^{X₁₁}_1, P₂)^{X₁₂}_2, P₃)^{X₃₄}_3, P₄)^{X₅₃}_5\).
  - If it is a corrupted public key, set \((X₁₁, X₁₂) = (g^{x₁}, (g^{x₁})^{1/3}) = (g^{x₁}, h^{x₁/3})\).
  - If it is an uncorrupted public key, the guessed public key, set \((X₁₁, X₁₂) = ((g^{x₁})^{1/3}, (g^{x₁})^{1/3}) = (g^{x₁}, h^{1/3}(x₁/3))\); otherwise, set \((X₁₁, X₁₂) = ((g^{x₁})^{1/3}, (g^{x₁})^{1/3}) = (g^{x₁}, h^{1/3}(x₁/3))\).
  - At last, the challenger records \((X₁₁, X₁₂, X₂₁, X₂₂, X₃₄, X₅₃)\) into the table \(T₁\).

- **O\(_{pk}\):** On input \((X₁₁, X₁₂, Z₁, Z₂, Z₃)\) by the adversary, if it is a corrupted public key, the challenger searches tuple \((X₁₁, X₁₂, X₂₁, X₂₂, X₃₄, X₅₃)\), and returns the corresponding \((X₁₁, X₁₂, X₂₁, X₂₂, X₃₄, X₅₃)\) to the adversary.

- **O\(_{pk}\):** On input \((pₖ₁, pₖ₂)\), where \(pₖ₁ = (X₁₁, X₁₂, Z₁, Z₂, Z₃)\) and \(pₖ₂ = (X₁₁, X₁₂, Z₁, Z₂, Z₃)\), and \(pₖ₁, pₖ₂\) are both from \(O\(_{pk}\), the challenger first searches the tuples corresponding to \(pₖ₁, pₖ₂\) in the table \(T₁\) and gets the values of \(x₁\) and \(x₂\), and then he chooses a random number \(r\) from \(Z_p^*\) and computes \(r^{(2)}_{pₖ₁, pₖ₂} = (Q₁, Q₂, T₁, T₂, T₃)\) and \(r^{(3)}_{pₖ₁, pₖ₂} = (\bar{A}, \bar{B}, \bar{C}, \bar{D})\) as the real execution.
  - If \(pₖ₁, pₖ₂\) are both corrupted, the challenger computes \(r^{(2)}_{pₖ₁, pₖ₂} = (g^{x₁})^{1/3} = h^{x₁/3} \cdot r\).
  - If \(pₖ₁\) is the guessed challenge public key and \(pₖ₂\) is uncorrupted, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₁})^{1/3} = h^{x₁/3} \cdot r\).
  - If \(pₖ₁\) is uncorrupted and \(pₖ₂\) is the guessed challenge public key, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₂})^{1/3} = h^{x₂/3} \cdot r\).
  - If \(pₖ₁\) and \(pₖ₂\) are both uncorrupted and neither of them is the guessed challenge public key, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₁})^{1/3} = h^{x₁/3} \cdot r\).
  - If \(pₖ₁\) is corrupted, and \(pₖ₂\) is uncorrupted but not the guessed challenge public key, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₂})^{1/3} = h^{x₂/3} \cdot r\).
  - If \(pₖ₁\) is corrupted and \(pₖ₂\) is the guessed challenge public key, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₂})^{1/3} = h^{x₂/3} \cdot r\).
  - If \(pₖ₁\) is corrupted and \(pₖ₂\) is the guessed challenge public key, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₁})^{1/3} = h^{x₁/3} \cdot r\).
  - If \(pₖ₁\) is corrupted but not the guessed challenge public key, and \(pₖ₂\) is corrupted, the challenger computes \(r^{(1)}_{pₖ₁, pₖ₂} = (g^{x₁})^{1/3} = h^{x₁/3} \cdot r\).

- **O\(_{c}\):** On input \((pₖ₁, pₖ₂, K)\), where \(pₖ₁ = (X₁₁, X₁₂, Z₁, Z₂, Z₃)\), \(pₖ₂ = (X₁₁, X₁₂, Z₁, Z₂, Z₃)\), \(K = (A, B, C, D, E, S)\), and \(pₖ₁, pₖ₂\) are both from \(O\(_{pk}\) the challenger first checks the validity of \(K\) as the real execution. If no, he outputs \(⊥\) and halts; otherwise, the challenger gets the corresponding \(x₁\) and \(x₂\) to \(pₖ₁, pₖ₂\) in the table \(T₁\), respectively.
  - If \(pₖ₁\) is the guessed challenge public key, and \(pₖ₂\) is corrupted, compute \(h^k = (C/(D^{(2)})^{(3)})^{1/3}(h₁(h₂)-h₂(h₃)(svk^n))\). And then choose a random numbers \(r\) and compute \(B = e((g^{x₁})^{1/3}, h^k)\). At last, the challenger computes \(Q₁, Q₂, T₁, T₂, T₃, \bar{A}, \bar{B}, \bar{C}, \bar{D}\), \(A, B, C, D\) as the real execution. Note that
    \[
    \begin{align*}
    \left(\frac{C}{D^{(2)}}\right)^{1/3}(h₁(h₂)-h₂(h₃)(svk^n)) & = (g^{(h₂(h₁))} \cdot h^k)^{1/3}(h₁(h₂)-h₂(h₃)(svk^n)) \\
    & = (h₁(h₂) \cdot h^k)^{1/3}(h₁(h₂)-h₂(h₃)(svk^n)) \\
    & = (h₁(h₂) \cdot h^k)^{1/3}(h₁(h₂)-h₂(h₃)(svk^n)) \\
    & = (h^k \cdot h₁(h₂)-h₂(h₃)(svk^n)) = (h₁(h₂)-h₂(h₃)(svk^n)) = h^k.
    \end{align*}
    \]
  - Otherwise, the challenger calls \(O_{c}\) itself to get a corresponding re-encryption key, and uses it to get the corresponding re-encrypted ciphertext of \((pₖ₁, K)\).

**Challenge:** The adversary outputs two same length messages \(m₀\) and \(m₁\) and a challenge public key \(pₖ^*\, if \(pₖ^*\) is not the one the challenger guessed in Phase 1, the challenger reports “failure” and aborts; otherwise, he chooses a random bit \(b\) and computes

- \(A' = svk^b\), \(B' = g^b\), \(D' = h^{b}\), \(E' = \text{SKE.Enc}(F(A')\mid C'\mid D')(T^{(2)})\), \(m₃ = \text{SIG.Enc}(K', (B', C', D', E'))\).

Note that \(e(B', g^{(H₂(A'))}) = e(g^b, C')\) and \(e(B', h') = e(g^b, D')\) always hold, since \(g^{(H₂(A'))} = g^{h₁(h₂)h₃}(svk^n) = h^{h₂/b} = h^k\) and \(h' = h^{k}\).

**Phase 2:** Almost the same as Phase 1, but with the restrictions listed in the IE-CCA-O game for single-use unidirectional PRE in Section 2.3.

The keys responses and the challenge phase are indistinguishable from those in Game G₂. The decryption oracle and re-encryption oracle are also indistinguishable from those in Game G₂, except that the challenger cannot always answer them when \(H₂(A) = H₂(svk^n), \) Since the target collision resistance of \(H₂\) we have \(H₂(A) = H₂(svk^n) \rightarrow A = svk^n\). On the one hand, before the challenge ciphertext is given, the adversary could query with a ciphertext where \(H₂(A) = H₂(svk^n) \) with probability \(q₃ \cdot (1 + \epsilon₁²)\). On the other hand, after the challenge is given, the adversary could query with a well-formed ciphertext where \(A = svk^n\) and yet the ciphertext is not the challenge ciphertext \(\epsilon₃\). If \(A = svk^n\) then the ciphertext
must not be identical to the challenge ciphertext. If the ciphertext is well-formed, then $S$ is a valid forgery against $\text{STG}$.

Furthermore, when the challenger receives a 5-EDBDH instance as input, its challenge ciphertext is a perfectly distributed, proper encryption of message $m_B$.

As a result, we have that

$$|\Pr[E_3] - \Pr[E_2]| \leq \epsilon_{-\text{EDBDH}} + \epsilon_{\text{SKE}} + q_{rd} \cdot (\delta + \epsilon_{\text{CTR}}^B) + \epsilon_{\text{CCA}}^C + \epsilon_{\text{PRBG}}^F \quad (4)$$

**Game $G_4$.** In this game, we modify the oracles related to uncorrupted public keys by using the IE-CCA security oracles of Cramer–Shoup scheme (Cramer and Shoup, 1998). In particular, we use the setup oracle of Cramer–Shoup scheme to get the values of $(P_1, P_2, Z_1, Z_2, Z_3)$ in the uncorrupted public keys, and use the decryption oracle of Cramer–Shoup scheme to decrypt $(A, B, C, D)$ and $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$ under the uncorrupted public keys.

Since the setup and decryption oracles of Cramer–Shoup scheme are perfect, we have that

$$\Pr[E_4] = \Pr[E_3] \quad (5)$$

**Game $G_5$.** In this game, we continue to modify the re-encryption oracle in Phase 2 as follows.

- $\mathcal{O}_{\text{re}}$ in **Phase 2**: On input $(pk_k, K)$, if $(pk_k, K)$ is the challenge original ciphertext, and $pk_k$ is uncorrupted, the challenger gets $B'$ as that in Game $G_4$, and uses $pk_k$ to encrypt $B'|A|C|D|E|\overline{A}|\overline{B}|\overline{C}|\overline{D}$ instead of $B'|A|C|D|E|\overline{A}|\overline{B}|\overline{C}|\overline{D}$ to get $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$, where $B'|A|C|D|E|\overline{A}|\overline{B}|\overline{C}|\overline{D} \neq B'|A|C|D|E|\overline{A}|\overline{B}|\overline{C}|\overline{D}$, but they are with the same bit length and same format. The other steps in this oracle are the same as that in Game $G_4$.

Since $(pk_k, (A, B, C, D))$ is a derivative of the challenge ciphertext, the adversary cannot query the decryption oracle with it. To check whether a ciphertext is a derivative of the challenge ciphertext, we can simply record the output of $\mathcal{O}_{\text{re}}$ when the input is $(pk_k^*, A, C^*)$. Hence, it is easy to see that if the adversary can distinguish Game $G_4$ from Game $G_5$, we can build an algorithm that breaks the IE-CCA security for Cramer–Shoup scheme (Cramer and Shoup, 1998) by using the adversary. As a result, we have that

$$|\Pr[E_5] - \Pr[E_4]| \leq q_{rd} \cdot \left(\frac{4 + q_d}{q} + \epsilon_{\text{ddh}} + \epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F\right) \quad (6)$$

**Game $G_6$.** In this game, we continue to modify the re-encryption key generation oracle and decryption oracle as follows.

- $\mathcal{O}_{\text{re}}$: On input $(pk_k, pk_j)$, if $pk_k$ is the guessed challenge public key, then $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$ is the ciphertext corresponding to $r^t$ ($\neq r$). The other steps in this oracle are the same as that in Game $G_5$. At last, $(pk_k, r, \overline{A}, \overline{B}, \overline{C}, \overline{D})$ is recorded into the table $T_{\text{re}}$.  

- $\mathcal{O}_{\text{dec}}$: On input $(pk_k, K)$, where $K$ is re-encrypted ciphertext. After getting $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$ as in Game $G_6$, the challenger searches $(pk_k, r, \overline{A}, \overline{B}, \overline{C}, \overline{D})$ in the table $T_{\text{re}}$. If it does not exist, the challenger does the same as that in Game $G_5$; otherwise, the challenger uses $r$ to compute Step 3 in $\text{PRE-Dec}$. The other steps in this oracle are the same as that in Game $G_6$.

It is easy to see that if the adversary can distinguish Game $G_5$ from Game $G_6$, we can build an algorithm that breaks the IE-CCA security for Cramer–Shoup scheme by using the adversary. As a result, we have that

$$|\Pr[E_5] - \Pr[E_4]| \leq q_{rd} \cdot \left(\frac{4 + q_d}{q} + \epsilon_{\text{ddh}} + \epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F\right) \quad (7)$$

**Game $G_7$.** In this game, we continue to modify the challenge phase, such that $F(A|x|C^t|D^t|T|x^t)$ is replaced by a random key $k^*$ from $\{0, 1\}^{\ell_1}$.

Since $T$ and $x^*$ are uniformly distributed, we have $T^t$ is uniformly distributed. Furthermore, $F$ is a PRBG. Hence, $F(T^t)$ is uniformly distributed over $\{0, 1\}^d$. Consequently, we have

$$|\Pr[E_7] - \Pr[E_6]| \leq \epsilon_{\text{PRBG}}^F \quad (8)$$

In Game $G_7$, only $D^t = \text{SKE.Enc}(k^*, m_B)$ uniquely fixes $m_B$ no matter in the original challenge ciphertext or the re-encrypted ciphertexts of the challenge ciphertext, hence we have that

$$\Pr[E_6] = \epsilon_{\text{CCA}}^C + \frac{1}{2}$$

Combining the equations (1)–(9), we have that

$$\text{Adv}_{\text{PRE-IE-CCA}}(\lambda) \leq q_{pk} \cdot \left(\epsilon_{-\text{EDBDH}} + \epsilon_{\text{SKE}} + 2\epsilon_{\text{CCA}}^C + 2\epsilon_{\text{PRBG}}^F + q_{rd} \cdot (\delta + \epsilon_{\text{CTR}}^{H_2}) + q_{ek} \left(\frac{4 + q_d}{q} + \epsilon_{\text{ddh}} + \epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F\right) + q_{rk} \left(\epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F\right)\right)$$

**Theorem 3.** Our proposal is IE-CCA-R secure in the standard model under assumptions that the DDH problem is hard, that $H_1$ is target collision resistant, and that $\overline{F}$ is a secure PRBG. In particular, we have that

$$\text{Adv}_{\text{PRE-IE-CCA-R}}^\text{IE-CCA-R} \leq \frac{4 + q_d}{q} + \epsilon_{\text{ddh}} + \epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F \cdot$$

The meanings of the notations are the same as that in Theorem 2.

This theorem can be easily obtained by the IE-CCA security of the Cramer–Shoup scheme, since the re-encrypted ciphertext is actually the ciphertext of Cramer–Shoup scheme, and the private key $(x_1, x_2, x_3, x_4, x_5)$ in our proposal is used as the same as that in Cramer–Shoup scheme.

**Theorem 4.** Our proposal is IK-CCA secure in the standard model under assumptions that the DDH problem is hard, that $H_1, H_2$ are target collision resistant, $H_1$ is collision resistant, and that $\overline{F}$ is a secure PRBG. In particular, we have that

$$\text{Adv}_{\text{PRE-IE-CCA}}^\text{IK-CCA} \leq \frac{4 + 2q_d}{q} + 2\epsilon_{\text{ddh}} + \epsilon_{\text{CTR}}^{H_1} + \epsilon_{\text{PRBG}}^F + \epsilon_{H_1} + \epsilon_{\text{CTR}}^{H_2},$$

where $\epsilon_{\text{CTR}}^{H_1}, \epsilon_{\text{CTR}}^{H_2}$ are the probabilities that the adversary breaks the target collision resistance of $H_1, H_2, \epsilon_{H_1}^2, \epsilon_{H_2}^2$ is the probability that the adversary breaks the collision resistance of $H_1$, and the meanings of the other notations are the same as that in Theorem 2.

**Proof.** We incrementally define a sequence of games starting at the real attack (Game $G_0$), and ending up at Game $G_4$, which clearly shows that the adversary cannot break the system. We define $E_0$ to be the event that $b = b'$ in Game $G_0$, where $b$ is the bit involved in the challenge phase, and $b'$ is the output of $A$ in the guess phase.

**Game $G_0$.** This game corresponds to the real attack (the IK-CCA game for single-use unidirectional PRE). By definition,

$$\Pr[E_0] = \frac{1}{2} = \text{Adv}_{\text{PRE-IE-CCA}}(\lambda)$$

**Game $G_1$.** In this game, we modify the oracles related to uncorrupted public key $pk$'s by using the IK-CCA security oracles of Cramer–Shoup scheme (Bellare et al., 2001). In particular, we use the setup oracle of Cramer–Shoup scheme to get the values of $(P_1, P_2, Z_1, Z_2, Z_3)$ in $pk$, and use the decryption oracle of Cramer–Shoup scheme to decrypt $(A, B, C, D)$ and $(\overline{A}, \overline{B}, \overline{C}, \overline{D})$ under $pk$. 
Since the setup and decryption oracles of Cramer–Shoup scheme are perfect, we have that
\[
Pr[E_1] = Pr[E_0]
\]  
(11)

\textbf{Game G2.} In this game, we modify the challenge phase as follows.

\textbf{Challenge:} On input \((pk_1, pk_2) = ((X_1, X_2, Z_1, Z_2, Z_3), (X_1, X_2, Z_1, Z_2, Z_3))\) by the adversary, the challenger chooses a random bit \(b\). If \(b = 1\), the challenger performs Steps 1–3 in PRE.ReKeyGen to get \(rk^{(3)}_{pk_1, pk_2}\), and uses an uncorrupted public re-encryption key \(pk\neq pk_1\) instead of \(pk_1\) to encrypt \(r\) to get \(rk^{(3)}_{pk_1, pk_2} = (\hat{A}, \hat{B}, \hat{C}, \hat{D})\). The other steps are the same as that in Game G1.

Since the IK-CCA game restricts the adversary from querying decryption oracle with the ciphertexts re-encrypted by the challenge re-encryption key, and the corresponding plaintext of \((\hat{A}, \hat{B}, \hat{C}, \hat{D})\) has different format with that of the ciphertexts from \text{PRE.Enc} and \text{PRE.ReEnc}. The challenger only needs to decrypt \((\hat{A}, \hat{B}, \hat{C}, \hat{D}) = (\hat{A}, \hat{B}, \hat{C}, \hat{D})\) and \((\hat{A}, \hat{B}, \hat{C}, \hat{D}) \neq (\hat{A}, \hat{B}, \hat{C}, \hat{D})\).

Therefore, it is easy to see that if the adversary can distinguish Game G3 from Game G2, then we can build an algorithm that breaks IK-CCA security of Cramer–Shoup scheme by using the adversary. As a result, we have that
\[
|Pr[E_2] - Pr[E_1]| \leq \frac{4 + 2q_d}{q} + \epsilon_{ddh} + \epsilon_{H_1} + \epsilon_{PRBC}
\]  
(12)

\textbf{Game G3.} In this game, we continue to modify the challenge phase as follows.

\textbf{Challenge:} On input \((pk_1, pk_2) = ((X_1, X_2, Z_1, Z_2, Z_3), (X_1, X_2, Z_1, Z_2, Z_3))\) by the adversary, the challenger chooses a random bit \(b\). If \(b = 1\), the challenger performs Steps 1–2 in PRE.ReKeyGen to get \(rk^{(1)}_{pk_1, pk_2}\). Then, when the adversary chooses random numbers \(Q_1, Q_2, T_1, T_2, T_3\) from \(\mathbb{G}_T\), and sets \(rk^{(2)}_{pk_1, pk_2} = (Q_1, Q_2, (T_1, H_{r1}(r), (T_2, H_{r2}(r), T^*)). The other steps are the same as that in Game G2.

Due to the following reasons, the challenger never needs to actually decrypt the ciphertexts under \(rk^{(2)}_{pk_1, pk_2}\). Here, actually means the output is not \(⊥\).

- According to the restrictions in the IK-CCA game for single-use unidirectional PRE, the challenger never decrypts the re-encrypted ciphertexts computed by the challenge re-encryption key.
- Due to the DDH assumption, the adversary cannot get \(r\) from \((\hat{A}, \hat{B}, \hat{C}, \hat{D})\).
- Without \(r\), the adversary has no idea about \(H_{r1}(r)\) and \(H_{r2}(r)\) due to the target collision resistance of \(H_{r1}\) and \(H_{r2}\).
- Without knowing \(H_{r1}(r)\) and \(H_{r2}(r)\), the adversary cannot generate well-formed re-encrypted ciphertext without using the challenge re-encryption.

Hence, it is easy to see that if the adversary can distinguish Game G3 from Game G2, we can build an algorithm that breaks the DDH problem in \(\mathbb{G}_T\) (given \((P_1, P_2, t_1, t_2, t_3, Q_1, Q_2, t_1^*, t_2^*, t_3^*)\), decide whether \(\log_{p_1} Q_1 = \log_{p_1} Q_2 = \log_{t_1} t_1 = \log_{t_2} t_2 = \log_{t_3} t_3^*\)) by using the adversary. As a result, we have that
\[
|Pr[E_3] - Pr[E_2]| \leq \epsilon_{ddh} + \epsilon_{H_1} + \epsilon_{H_2}
\]  
(13)

Furthermore, in Game G3, it is easy to see that when \(b = 1\), the challenge re-encryption key is a random value from the re-encryption key space. Hence, we have that
\[
Pr[E_3] = \frac{1}{2},
\]  
(14)

Combining the equations (10)–(14), we have that
\[
\text{Adv}_{\text{IK-CCA}}^\text{PRE}(\lambda) \leq \frac{4 + 2q_d}{q} + 2\epsilon_{ddh} + \epsilon_{H_1} + \epsilon_{PRBC} + \epsilon_{H_1} + \epsilon_{H_2}
\]

\[\square\]

3.4. The anonymous sharing system equipped with our PRE scheme

In this subsection, we give the details of the anonymous sharing system showed in Fig. 1 equipped with our PRE scheme.

Before the anonymous sharing system starts to work, every group member has obtained the group account and generated its own public–private key pair by running PRE.Setup and PRE.KeyGen. The owner firstly encrypts her files \(m\) by running \(C_1 \leftarrow \text{PRE.Enc}(pk_o, m)\), where \(pk_o\) and \(m\) are the public key of the owner and files to be encrypted, respectively. Then the owner uses the group account to login the server and upload the encrypted files.

When the owner wants to share her files with other group member, she firstly generates the re-encryption key \(rk_{o,s}\) by running PRE.ReKeyGen\((sk_o, pk_s)\), where \(sk_o\) and \(pk_s\) are the private key of the owner and the public key of the intended group member, respectively. Secondly, the owner uses the group account to login the server and upload \(rk_{o,s}\) with a label \(L_{o,s}\). Thirdly, the sender sends the label \(L_{o,s}\) to the intended group member.

When the intended group member wants to obtain the shared files, he uses the group account to login the server and then sends the label \(L_{o,s}\) to the server. On receiving the label \(L_{o,s}\), the server re-encrypts the encrypted files by running \(C_2 \leftarrow \text{PRE.ReEnc}(rk_{o,s}, C_1)\), sends \(C_2\) to the intended group member, who can obtain the files \(m\) by running PRE.Dec\((sk_o, C_2)\), where \(sk_o\) is the private key of the intended group member.

4. Conclusion

In this paper, we first revised the security model of key privacy for single-use unidirectional PRE. The revised security model allows multiple re-encryption keys to correspond to one delegation, and allows the adversary to get the re-encryption key of any delegation he wants. After that, based on the 5-EDBDH assumption and DDH assumption, we proposed the first PRE scheme which is CCA-secure and key-privacy in the standard model.

Acknowledgements

The authors thank the anonymous reviewers for their insightful comments and helpful suggestions. Jun Shao was supported by NSFC, no. 61003308; ZJGSUSF, no. 1130X(M2010045); and ECZJF, no. Z201017312. Peng Liu was supported by AFSOR, FA9550-07-1-0527 (MURI); ARO, W9111NF-09-1-0525 (MURI); NSF, CNS-0905131 and CNS-0916469. Yuan Zhou was supported by NSFC, No. 60873217.

References
