J Ambient Intell Human Comput (2015) 6:699–711
DOI 10.1007/s12652-015-0261-3

ORIGINAL RESEARCH

A new proxy re-encryption scheme for protecting critical information systems

Xu An Wang[1] · Jianfeng Ma[2] · Xiaoyuan Yang[3]

Received: 15 December 2014 / Accepted: 24 February 2015 / Published online: 19 March 2015

Abstract The risks of critical systems involved in key-recovery, key-escrow have barely taken to be seriously treated by the researchers. And the failures of even the best cryptographic techniques are often caused by the inherent security weaknesses in our computer systems rather than breaking the cryptographic mechanism directly. Thus key-recovery and key-escrow attacks are among the most important issues in protecting critical information systems. Proxy re-encryption, introduced by Blaze et al. in 1998, allows a proxy to transform a ciphertext computed under Alice's public key into one that can be opened under Bob's decryption key, without the proxy knowing any secret key of Alice and Bob, thus it can be used in modern critical information system well to avoid the key-recovery and key-escrow attack. In CANS'08, Deng et al. proposed the first IND-CCA2 secure proxy re-encryption without bilinear parings in the random oracle model. They left an open problem of constructing IND-CCA2 secure proxy re-encryption scheme in the standard model yet without pairings. In this paper, based on Cramer–Shoup encryption scheme, we try to solve this open problem by presenting a new proxy re-encryption scheme, which is IND-CCA2 secure in the standard model in a relatively weak model and does not use bilinear parings. Our main idea is roughly using the Cramer–Shoup encryption twice, but also taking care of the security in the security model of proxy re-encryption. We compare our work with Canetti–Hohenberger scheme II, the results show our scheme is more efficient. We also show its application in protecting the security of critical information systems.

Keywords Restricted chosen ciphertext security · Cramer–Shoup encryption · Standard model · Critical systems

1 Introduction

Today we are more and more relying on using information technology to smoothly running our work in our daily life. And these information technology always heavily needs critical infrastructures. Generally speaking, the following are the major critical national infrastructures: telecommunications; generation, transmission and distribution of electric power; storage and distribution of gas and oil; water supplies; transportation; banking and finance; emergency services; and continuity of government services etc. (Abelson et al. 1997). They all need modern information technology to work smoothly. All of these critical infrastructures are closely interdependent and that they all depend on underlying computer-communication information infrastructures, such as the communication networks, software architectures and intelligent systems, computing resources, databases, private networks, the Internet, and cloud computation etc. (Solhaug and Seehusen 2014; Yao et al. 2014; Spaho et al. 2014; Xhafa et al. 2014; Li and Kim 2010; Li et al. 2011, 2014). However if the security of these critical systems are weak, many problems and even

✉ Xu An Wang
wangxazjd@163.com

[1] School of Telecommunications Engineering, Xidian University, 710071 Xi'an, Shaanxi, People's Republic of China

[2] School of Computer Science and Technology, Xidian University, 710071 Xi'an, Shaanxi, People's Republic of China

[3] Key Laboratory of Information and Network Security, Engineering University of Chinese Armed Police Force, 710086 Xi'an, Shaanxi, People's Republic of China

some disaster results can be risen. A fact we must know is that very serious vulnerabilities and threats exist in all of these critical infrastructures. Such systems must be able to recover from the failures and intrusion in order to maintain their function. Therefore, achieving resilience and security in the complex, interconnected, and interdependent systems are very important for us to assure our whole system to be safe.

Using strong cryptography is essential to protect the security of the future of computer-communication systems, therefore to the protection of the critical national infrastructures. Cryptography is critical in achieving confidentiality, authentication, and integrity, although it is only one small link in the whole protection mechanism, it is a vital link and among one of the most trustful ones. Although cryptographic compromises have not been a major source of security risks in the past, preventing them will be especially critical to the successful conduct of electronic commerce, which is growing very rapidly and places stringent demands on computer and communication systems. Secure implementation of cryptographic systems requires strong operating systems, networking, and application software, and strong authentication of users and systems.

The risks involved in key-recovery, key-escrow in cryptography are the most serious threatens in the modern information security systems (Clark et al. 1996). Key management is one of the most complex part in a cryptographic system and can be easily running wrong. A thorough analysis of the potential risks of key management must be conducted before instituting any key-management technologies that could be inherently flawed, extremely riskful, and possibly counterproductive to the overall goal of protecting the infrastructures. In reality, the use of flawed key-management techniques would greatly reduce security rather than increase it, and these implementations are very common. Indeed there is a common belief that cryptographic systems are typically broken not by exhaustively searching for the keys, but rather by finding much simpler ways to bypass or compromise the cryptography and key management.

One of the serious risks of cryptographic key recovery is that today's information infrastructures are so weak, and that inherent risks are likely to arise in the key management itself. Thus good cryptographic key management related technique is very desired. The concept of proxy re-encryption is one of the promising good techniques to deal with the key management problems, which is the main topic of this paper. We can see Fig. 1 for its using in protecting security of modern critical systems.

Blaze et al. (1998) proposed a new kind of cryptographic primitive called *proxy re-encryption* which is a strictly subfield of proxy encryption. In proxy re-encryption, a proxy can transform a ciphertext computed under Alice's public key into one that can be opened under Bob's decryption key. The key feature of proxy re-encryption is that the proxy can implement the transforming functionality without knowing any of Alice or Bob's secret key. Thus it reduce the additional work on key-management, which is critical for protecting billions of modern embedding information control equipments.

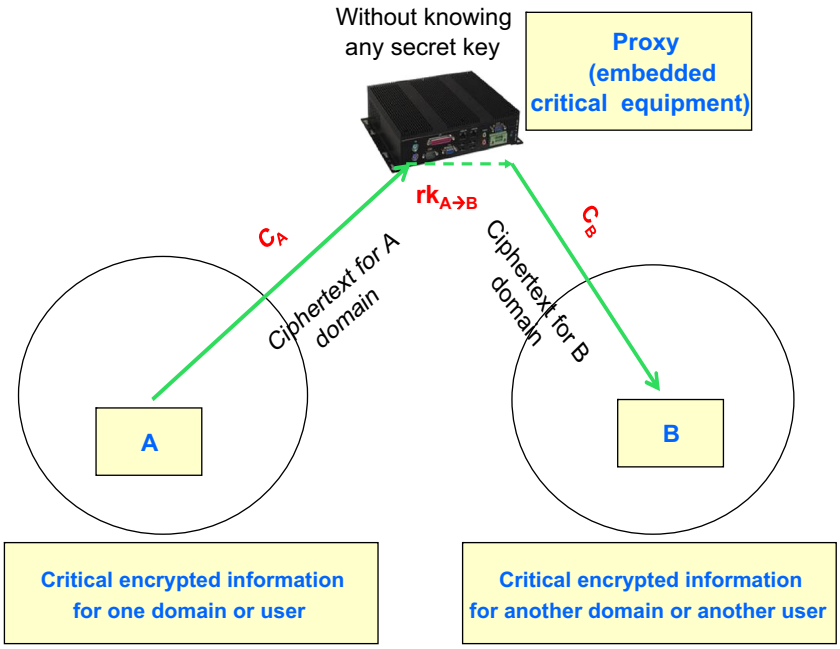

Fig. 1 Using proxy re-encryption in protecting critical information systems

In NDSS'05, Ateniese et al. (2005, 2006) proposed a few new proxy re-encryption schemes and discussed its several potential applications especially in the secure distributed storage. In CCS'07, Canetti and Hohenberger (2007) proposed the first IND-CCA2 secure bidirectional proxy re-encryption scheme. Later in PKC'08, Libert and Vergnaud (2008) proposed the first IND-CCA2 secure unidirectional proxy re-encryption scheme. But these schemes are both constructed by using bilinear pairing. Until very recently, Deng et al. (2008) proposed the first IND-CCA2 proxy re-encryption scheme without pairings. They left an open problem to construct an IND-CCA2 secure proxy re-encryption without pairing in the standard model. In this paper, we aim at solving this open problem. Based on Cramer–Shoup encryption scheme and using the a technique called *proof of equality of two discrete logarithms*, we first construct an IND-CCA2 secure proxy re-encryption scheme in the random oracle model. This result can be seen as another IND-CCA2 secure proxy re-encryption scheme without pairings. Then we show how to construct the IND-CCA2 secure proxy re-encryption scheme in the standard model under a relatively weak model and without pairings.

1.1 Background

In a PRE scheme, a proxy with re-encryption key can transform a ciphertext computed under Alice's public key into one that can be opened under Bob's decryption key. But the proxy can't get any information about the plaintext or the secret keys of the delegator or delegatee. Blaze et al. (1998) proposed the first proxy re-encryption scheme based on ElGamal encryption. But this scheme is *bidirectional* and *colluding unsafe*. PRE is rooted from *proxy encryption*. Mambo and Okamoto (1997) proposed the concept of *proxy encryption*. In their scheme, the delegatee can decrypt the ciphertext by cooperating with the proxy. But neither the delegatee nor the proxy can decrypt the ciphertext by himself.

In 2005, Ateniese et al. (2005, 2006) proposed the first *unidirectional* proxy re-encryption schemes. They proposed three attempts to construct their unidirectional proxy re-encryptions: the first is based on a variant of Paillier encryption, the second is based on a variant of BBS scheme with pairings, and the third is a two level encryption scheme with pairings. But these schemes can only achieve IND-CPA secure. They leave constructing IND-CCA2 secure proxy re-encryption schemes as an open problem.

In CCS'07, Canetti and Hohenberger (2007) proposed the first IND-CCA2 secure proxy re-encryption scheme by applying the CHK transformation Canetti et al. (2003) to the second scheme in Ateniese et al. (2005, 2006). The idea of their scheme is as following: Assume the ciphertext needs to be re-encrypted is $c = (X, Y)$. If the encrypter signs (X, Y) in the CHK transformation, then the proxy can't re-encrypt (X, Y) without invalidating the signature. But if the encrypter only signs, say Y, then the adversary can arbitrarily mutate X, thus changes the decryption value. To solve this problem, they smartly add an element Z to the ciphertext, such that (Y, Z) will be signed and Z allows anyone to check that the unsigned value X was not mutated in any meaningful way.

In PKC'08, Libert and Vergnaud (2008) further extended their research. They proposed the second IND-CCA2 secure proxy re-encryption scheme based on the third scheme in Ateniese et al. (2005, 2006). They follow the paradigm of Canetti and Hohenberger (2007). But if directly apply CHK transformation to the second scheme in Ateniese et al. (2005, 2006), the validity of translated ciphertext cannot be *publicly checked*. Thus they randomize the re-encryption algorithm of the second scheme in Ateniese et al. (2005, 2006) to make the re-encrypted ciphertexts *publicly verifiable*.

Both of the IND-CCA2 secure proxy re-encryption schemes are following the paradigm of CHK transformation and based on bilinear pairings. They leave the open problem of constructing IND-CCA2 secure proxy re-encryption schemes without bilinear pairing.

In CANS'08, Deng et al. (2008) proposed the first IND-CCA2 secure proxy re-encryption scheme without pairings. They smartly integrated an CCA secure hashed Elgamal encryption and a modified Schnnor's signature into a proxy re-encryption scheme, while achieving IND-CCA2 secure, which no longer follows the CHK transformation. Since this scheme is constructed without pairing, it is almost the most efficient scheme in the literature.

1.2 Our contribution

We propose an IND-CCA2 secure proxy re-encryption scheme based on Cramer–Shoup encryption in the standard model in a relatively weak model. The idea of our scheme is different from all the others. In the scheme, we require the re-encryption key must be secret. The delegator sends the proxy's "blinding checking" key and encrypts the delegatee's "blinding checking" key to the proxy. Then the proxy first checks the original ciphertext's validity, re-encrypts the ciphertext and sends the ciphertext corresponding to the delegatee's "blinding checking" key to the delegatee. We assure the re-encrypted ciphertext can not be mutated by the *universal one way hash function* by outside attackers.

1.3 Roadmap

We organize our paper as following. In Sect. 2, we recall the concept of IND-CCA2 secure proxy re-encryption

scheme and its security model. In Sect. 3 we review some basic ingredients of our scheme including Cramer–Shoup encryption. In Sect. 4, we propose our scheme in the standard model which we denote as $\prod_{ST}$. For clearly understanding our work, we compare the efficiency of our scheme with CH II scheme in Sect. 4.3. In Sect. 4.4, we prove the security of $\prod_{ST}$ in the standard model in a relatively weak model. Finally we give our conclusion in Sect. 5.

2 Definition and Security models for bidirectional proxy re-encryption

2.1 Definition for bidirectional proxy re-encryption

First we recall the definition of bidirectional proxy re-encryption in Canetti and Hohenberger (2007).

Definition 1 (*Bidirectional PRE*) A bidirectional proxy re-encryption scheme (PRE) is a tuple of algorithms (KeyGen, ReKeyGen, Enc, ReEnc, Dec):

1. KeyGen $(1^k) \rightarrow (pk, sk)$. On input the security parameter 1^k, the key generation algorithm KeyGen outputs a public key pk and a secret key sk.
2. ReKeyGen $(sk_1, sk_2) \rightarrow rk_{1\leftrightarrow 2}$. On input two secret keys sk_1 and sk_2, the re-encryption key generation algorithm ReKeyGen outputs a bidirectional re-encryption key $rk_{1\leftrightarrow 2}$.
3. Enc $(pk, m) \rightarrow C$. On input a public key pk and a message $m \in \{0,1\}^*$, the encryption algorithm Enc outputs a ciphertext C.
4. ReEnc $(rk_{1\leftrightarrow 2}, C_1) \rightarrow C_2$. On input a re-encryption key rk and a ciphertext C_1, the re-encryption algorithm ReEnc outputs a second ciphertext C_2 or the error symbol $\perp$.

Roughly speaking, the correctness requires that, for all $m \in \mathcal{M}$ and all $(pk, sk) \leftarrow$ KeyGen (1^k), it holds that Decrypt(sk, Encrypt $(pk, m)) = m$. Besides, for all $(pk_i, sk_i) \leftarrow$ KeyGen (1^k) and $(pk_j, sk_j) \leftarrow$ KeyGen (1^k), it holds that Decrypt $(sk_j$, ReEncrypt(RekeyGen (sk_i, sk_j), Encrypt $(pk_i, m)) = m$.

In proxy re-encryption, if a ciphertext can be consecutively re-encrypted, this proxy re-encryption scheme is said to be *multi-hop*. In contrast, if a re-encrypted ciphertext can not be further re-encrypted, this proxy re-encryption scheme is said to be *single-hop*. For our scheme is a *single-hop* proxy re-encryption scheme, we only concentrate on this kind of proxy re-encryption schemes.

2.2 Security models for bidirectional proxy re-encryption

Now we review the game used for formulating the security requirement which is the same as defined in Canetti and Hohenberger (2007). The game defines an interaction between an adversary and a number of oracles, representing the capabilities of the adversary in an interaction with a PRE scheme. It proceeds as follows:

Definition 2 (*Bidirectional PRE-CCA game*) Let k be the security parameter. Let $\mathcal{A}$ be an oracle TM, representing the adversary. The game consists of an execution of $\mathcal{A}$ with the following oracles, which can be invoked multiple times in any order, subject to the constraints below:

- Uncorrupted key generation oracle: obtain a new key pair as $(pk, sk) \leftarrow KeyGen(1^k)$, $\mathcal{A}$ is given pk.
- Corrupted key generation oracle: obtain a new key pair as $(pk, sk) \leftarrow KeyGen(1^k)$. $\mathcal{A}$ is given both pk and sk.
- Re-encryption key generation oracle: on input (pk, pk') by the adversary, where pk, pk' were generated before by KeyGen, return the re-encryption key $rk_{pk\leftrightarrow pk'} = ReKeyGen(sk, sk')$ where sk, sk' are the secret keys that correspond to pk, pk'. We require that either both pk and pk' are corrupted, or alternatively both are uncorrupted. We do not allow for re-encryption key generation queries between a corrupted and an un-corrupted key. (This represents the restriction that the identities of parties whose security is compromised should be fixed in advance).
- Challenge Oracle: this oracle can be queried once. On input (pk^*, m_0, m_1), where pk^* is called the challenge key, the oracle chooses a bit $b \leftarrow \{0,1\}$ and returns the challenge ciphertext $C^* = Enc(pk^*, m_b)$. (As we note later, the challenge key must be uncorrupted for $\mathcal{A}$ to win).
- Re-encryption oracle: on input (pk, pk', C), where (pk, pk') were generated before by KeyGen, if pk' is corrupted and (pk, C) is a derivative of (pk^*, C^*), then return a special symbol $\perp$ which is not in the domains of messages and ciphertexts. Else, return the re-encrypted ciphertext $C' = ReEnc(ReKeyGen\,(sk, sk'), C)$. Derivatives of (pk^*, sk^*) are defined inductively, as follows:
  1. (pk^*, C^*) is a derivative of itself.
  2. If (pk, C) is a derivative of (pk^*, C^*) and (pk', C') is a derivative of (pk, C), then (pk', C') is a derivative of (pk^*, C^*).
  3. If $\mathcal{A}$ has queried the re-encryption oracle O_{enc} on input (pk, pk', C) and obtained response (pk', C'), then (pk', C') is a derivative of (pk, C).

4. If $\mathcal{A}$ has queried the re-encryption key generation oracle O_{rkey} on input (pk, pk') or (pk', pk), and $Dec(pk', C') \in (m_0, m_1)$, then (pk', C') is a derivative of (pk, C).

- Decryption Oracle: on input (pk, C), if the pair (pk, C) is a derivative of the challenge ciphertext C^*, or pk was not generated before by KeyGen, then return a special symbol $\perp$ which is not in the domain D of messages. Else, return $Dec(sk, C)$.
- Decision Oracle: this oracle can also be queried only once, on input b', if $b' = b$ and the challenge key pk^* is not corrupted, then output 1, else output 0.

We say that $\mathcal{A}$ wins the PRE-CCA game with advantage ε if the probability, over the random choices of $\mathcal{A}$ and the oracles, that the decision oracle is invoked and outputs 1, is at least $1/2 + \varepsilon$.

For the details about definitions of derivatives, please refer to Canetti and Hohenberger (2007).

Definition 3 (*Bidirectional PRE-CCA security*) A PRE scheme is bidirectional PRE-CCA secure for domain D of messages if it is correct for D as in Definition 2, and any probabilistic polynomial time adversary wins the bidirectional PRE-CCA game only with negligible advantage.

Definition 4 (*Bidirectional PRE-CCA game in the Weak Model*) This game is almost the same as above with the following exception:

- The proxy is not an attacker and it protect its re-encryption keys well. This is a reasonable assumption, for in almost the existing bidirectional PRE scheme Canetti and Hohenberger (2007), the proxy is forbidden to collude with the delegatee or the delegator and the re-encryption keys need to be protected well.
- The delegatee do not attack the CCA security of the delegator, we think this is a reasonable assumption, for if the delegator allows the proxy re-encrypt the ciphertext to the delegatee, the delegatee can always know the corresponding plaintext, especially for the bidirectional PRE case.

3 Preliminary

In this section, we review some basic backgrounds needed to construct our bidirectional proxy re-encryption scheme.

3.1 Basic Cramer–Shoup encryption

The scheme assumes a group G of prime order q, where q is large, assume that cleartext messages are (or can be encoded as) elements of G. It uses a universal one-way family of hash functions that map long bit strings to elements of Z_q.

1. **KeyGeneration**. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in \mathbb{G}$ are chosen, and random elements $x_1, x_2, y_1, y_2, z \in Z_q$ are also chosen. Next, the group elements $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$ are computed. Next, a hash function $H : G^3 \to \mathbb{Z}_q^*$ is chosen from the family of universal one-way hash functions. The public key is (g_1, g_2, c, d, h, H), the private key is (x_1, x_2, y_1, y_2, z).
2. **Encryption**. Given a message $m \in G$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes $u_1 = g_1^r, u_2 = g_2^r, e = h^r m, \alpha = H(u_1, u_2, e), v = c^r d^{r\alpha}$. The ciphertext is (u_1, u_2, e, v).
3. **Decryption**. Given a ciphertext (u_1, u_2, e, v), the decryption algorithm runs as follows. It first computes $\alpha = H(u_1, u_2, e)$, and tests if $u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha} = v$. If this condition does not hold, the decryption algorithm outputs "reject"; otherwise, it outputs $m = e/u_1^z$.

Cramer–Shoup encryption is the first provable IND-CCA2 secure encryption scheme in the standard model Cramer and Shoup (1998). Its security is based on DDH assumption. It has great influence on constructing IND-CCA2 secure encryption schemes. Since then, many variants have been proposed Cramer and Shoup (2003); Kurosawa and Desmedt (2004), thus our scheme can be viewed as the first work on constructing IND-CCA2 secure proxy re-encryption in the framework of *hash proof system*.

3.2 Collision resistant hash function

A family of hash functions is said to be collision resistant, if upon drawing a function H at random from the family, it is infeasible for an adversary to two different inputs x and y such that $H(x) = H(y)$ (Bellare and Rogaway 1997).

4 Our proposed scheme

4.1 Main idea

From Cramer–Shoup encryption in Sect. 3.1, we find that Cramer–Shoup encryption ciphertext can not be *publicly verifiable*. So our main difficulty is to make Cramer–Shoup encryption ciphertext *publicly verifiable*. Note that the verification equation is of the form:

$$u_1^{x_1 + y_1\alpha} u_2^{x_2 + y_2\alpha} = v \tag{1}$$

For a valid Cramer–Shoup encryption ciphertext (u_1, u_2, e, v), if we add a randomly $k \in [1, q)$ to all the exponentiation, the above equation is still satisfied. That is, we have

$$u_1^{kx_1+ky_1\alpha}u_2^{kx_2+ky_2\alpha} = v^k \quad (2)$$

Furthermore, we can make (kx_1, kx_2, ky_1, ky_2) as the check key to verify the Cramer–Shoup encryption ciphertext.

4.2 IND-CCA2 secure proxy re-encryption based on Cramer–Shoup encryption in the standard model in the relative weak model

The scheme assumes a group G of prime order q, where q is large, and cleartext messages are (or can be encoded as) elements of G. It uses a universal one-way family of hash functions that map long bit strings to elements of Z_q.

1. **KeyGen**. The key generation algorithm runs as follows. Random elements $g_1, g_2 \in \mathbb{G}$ are chosen and a hash function H is chosen from the family of universal one-way hash functions, and all of them are available to all users. The delegator randomly chooses elements $(x_1, x_2, y_1, y_2, k_1, k_2, z)$ all from Z_q and computes

$$c = g_1^{x_1}g_2^{x_2},\ d = g_1^{y_1}g_2^{y_2},$$
$$c_1 = g_1^{k_1x_1}g_2^{k_1x_2},\ d_1 = g_1^{k_1y_1}g_2^{k_1y_2},$$
$$c_2 = g_1^{k_2x_1}g_2^{k_2x_2},\ d_2 = g_1^{k_2y_1}g_2^{k_2y_2},\ h = g_1^z$$

The delegator's public key is

$$pk = (g_1, g_2, c, d, c_1, d_1, c_2, d_2, h, H)$$

and the corresponding private key is

$$sk = (x_1, x_2, y_1, y_2, k_1, k_2, z)$$

The delegatee randomly chooses elements $(x_1', x_2', y_1', y_2', k_1', k_2', z')$ all from Z_q, and computes

$$c' = g_1^{x_1'}g_2^{x_2'},\ d' = g_1^{y_1'}g_2^{y_2'},$$
$$c_1' = g_1^{k_1'x_1'}g_2^{k_1'x_2'},\ d_1' = g_1^{k_1'y_1'}g_2^{k_1'y_2'},$$
$$c_2' = g_1^{k_2'x_1'}g_2^{k_2'x_2'},\ d_2' = g_1^{k_2'y_1'}g_2^{k_2'y_2'},\ h' = g_1^{z'}.$$

The delegatee's public key is

$$pk' = (g_1, g_2, c_1', d_1', c_2', d_2', h', H)$$

and the corresponding private key is

$$sk' = (x_1', x_2', y_1', y_2', k_1', k_2', z')$$

2. **ReKeyGen**. The delegator runs basic Cramer–Shoup encryption. It encrypts $(k_2x_1,\ k_2x_2, k_2y_2,\ k_2y_2)$ under the delegatee's public key (g_1, g_2, c', d', H). Here we denote this ciphertext as $c_0 = (u_1^0, u_2^0, e^0,\ v^0)$. Similarly, the delegatee runs basic Cramer–Shoup encryption. It encrypts $(k_2'x_1', k_2'x_2', k_2'y_1', k_2'y_2')$ under the delegator's public key (g_1, g_2, c, d, H), we denote the ciphertext as $c_0' = (u_1'^0, u_2'^0, e'^0, v'^0)$. On input

$$sk = (x_1, x_2, y_1, y_2, k_1, k_2, z),$$
$$sk' = (x_1', x_2', y_1', y_2', k_1', k_2', z')$$

the delegator and delegatee run interactively and output the re-encryption key

$$rk = (k_1x_1, k_1x_2, k_1y_1, k_1y_2, k_1'x_1', k_1'x_2', k_1'y_1', k_1'y_2', c_0, c_0', z/z')$$

and send it to the proxy via secure channel, *rk* must be kept private.

3. **Enc**. Given a message $m \in G$, this algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes

$$u_1 = g_1^r,\ u_2 = g_2^r,\ e = h^r m,\ \alpha = H(u_1, u_2, e),$$
$$v = c^r d^{r\alpha},\ v_2 = {c_2}^r {d_2}^{r\alpha},\ \alpha' = H(u_1, u_2, e, h^r, m),$$
$$\alpha_1 = H(u_1, u_2, e, v, v_2, \alpha'),\ v_1 = {c_1}^r {d_1}^{r\alpha_1}.$$

The ciphertext is $(u_1, u_2, e, v, v_2, v_1, \alpha')$.

4. **ReEnc**. If

$${u_1}^{k_1x_1+k_1y_1\alpha_1}{u_2}^{k_1x_2+k_1y_2\alpha_1} \neq v_1$$

where $\alpha_1 = H(u_1, u_2, e, v, v_2, \alpha')$, this algorithm returns "reject". Else it computes ${u_1}' = {u_1}^{z/z'}$, and returns $(u_1, {u_1}', u_2, e, v_2, \alpha', c_0)$ to the delegatee.

5. **Dec$_1$**. Given a normal ciphertext $(u_1, u_2, e, v, v_2, v_1, \alpha')$, this algorithm runs as follows. If

$${u_1}^{x_1+y_1\alpha}{u_2}^{x_2+y_2\alpha} \neq v$$

where $\alpha = H(u_1, u_2, e)$ or

$${u_1}^{k_1x_1+k_1y_1\alpha_1}{u_2}^{k_1x_2+k_1y_2\alpha_1} \neq v_1$$

where $\alpha_1 = H(u_1, u_2, e, v, v_2, \alpha')$, it returns "reject"; otherwise, it computes $m = e/u_1^z$ and returns m.

6. **Dec$_2$**. Given a re-encryption ciphertext $(u_1, {u_1}', u_2, e, v_2, \alpha', c_0)$, this algorithm runs as follows.
   (a) It first runs basic Cramer–Shoup decryption algorithm. It decrypts c_0 and gets the plaintext $(k_2x_1, k_2x_2, k_2y_1, k_2y_2)$.
   (b) If
   $${u_1}^{k_2x_1+k_2y_1\alpha}{u_2}^{k_2x_2+k_2y_2\alpha} \neq v_2$$
   where $\alpha = H(u_1, u_2, e)$, it outputs "reject".
   (c) It computes $m = e/{u_1'}^{z'}$. If
   $$\alpha' \neq H(u_1, u_2, e, {u_1'}^{z'}, m)$$
   it outputs "reject".
   (d) Otherwise, it outputs m.

Note that the first level ciphertext is of the form $(u_1, u_2, e, \alpha',\ v, v_2, v_1)$. Compared with the basic Cramer–Shoup encryption, we add (α', v_2, v_1) to the ciphertext. $\alpha' = H(u_1, u_2, e, m)$ can be seen as a *tag* for m, which will help

Fig. 2 Efficiency Comparison between CH II Scheme and Our Scheme

Schemes			CH Scheme II	Our Scheme
Comput. Cost	Encrypt		$1t_p + 3t_e + 1t_{me} + 1t_s$	$3t_e + 3t_{me}$
	Re-Encrypt		$4t_p + 2t_e + 1t_v$	$1t_e + 1t_{me}$
	Decrypt	2nd-level CiphTxt	$5t_p + 2t_e + 1t_v$	$2t_e + 2t_{me}$
		1st-level CiphTxt	$5t_p + 2t_e + 1t_v$	$2t_e + 1t_{me}$
CiphTxt Length	2nd-level CiphTxt		$1\|pk_s\|+3\|\mathbb{G}_e\|+1\|\mathbb{G}_T\|+1\|\sigma_s\|$	$9\|\mathbb{G}\|+1\|\mathbb{Z}_q\|$
	1st-level CiphTxt		$1\|pk_s\|+3\|\mathbb{G}_e\|+1\|\mathbb{G}_T\|+1\|\sigma_s\|$	$6\|\mathbb{G}\|+1\|\mathbb{Z}_q\|$
Without Random Oracles?			✓	✓
Underlying Assumptions			DBDH	DDH

Fig. 3 Performance Comparison

Scheme	Encrypt	Re-encrypt	2-level Decrypt	1-level Decrypt
CH Scheme II	18.90ms	34.49ms	42.72ms	42.72ms
Our Scheme	20.16ms	6.16ms	12.32ms	12.32ms

the delegatee to verify the re-encrypted ciphertext not being mutated by the proxy or external adversaries. v_1 can be seen as a *tag* to the proxy for verifying the validity of the ciphertext by using $(k_1x_1, k_1x_2, k_1y_1, k_1y_2)$, which is directly sent by the delegator as the re-encryption key. v_2 can be seen as a *tag* to the delegatee for verifying the validity of the ciphertext by $(k_2x_1, k_2x_2, k_2y_1, k_2y_2)$, which is implicitly sent by the delegator as a part of re-encrypted ciphertext c_0 via proxy.

Although we make some changes to the basic Cramer–Shoup encryption, the scheme still does not rely on random oracle. We denote this scheme as $\prod_{ST}$.

4.3 Comparison

Since Deng et al.'s proxy re-encryption scheme is only secure in the random oracle model, to conduct a fair comparison, in this section, we only compare our $\prod_{ST}$ scheme with Canetti and Hohenberger's PRE II scheme (we denote as CH II Scheme) Canetti and Hohenberger (2007), which is bidirectional and IND-CCA2 secure in the standard model. Figure 2[1] gives a comparison between our scheme and CH II Scheme. The comparison results indicate that the computation cost of our $\prod_{ST}$ Scheme is much more efficient than CH II Scheme due to the heavy pairing computation in bilinear group. The encryption in CH II Scheme needs 3 exponentiations, 1 pairing, 1 multi-exponentiation and 1 one-time signature signing, while the encryption in our scheme only involves 3 exponentiations and 3 multi-exponentiations. The security of our scheme relies on the DDH assumption, which is weaker than the decisional bilinear Diffie–Hellman (DBDH) assumption used in CH II Scheme.

In Fig. 2, we neglect some operations such as hash function evaluation, modular multiplication and XOR, since the computational cost of these operations is far less than that of exponentiations or pairings. Note that, using the technique in Canetti and Goldwasser (1999), Kiltz and Galindo (2006) and Kiltz (2006), the re-encryption and decryption in Canetti–Hohenberger PRE II Scheme can further save two pairings, at the cost of several exponentiation operations.

In Fig. 3, we give the performance comparison between CH Scheme II and our scheme, according to the benchmark of JPBC library based on TestBed 1 (Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40 GHz, 3 GB Ram, Ubuntu 10.04). We remark that we just roughly give the comparison results which maybe are not very accurate, but are enough to give the hints on the comparison.

4.4 Security analysis

In this section, we will prove the security of $\prod_{ST}$ in the standard model.

Theorem 1 Our proxy re-encryption scheme $\prod_{ST}$ is secure against adaptive chosen ciphertext attack in the standard model under the relatively weak model assuming that (1) the hash function H is collision resistance, and (2) the Diffie–Hellman decision problem is hard in the group G.

Proof The intuition is that we make sure that the ciphertext is a valid Cramer–Shoup ciphertext whether before the re-encryption or after the re-encryption. The proxy can check that the ciphertext which needs to be re-encrypted is a valid Cramer–Shoup ciphertext by "blinding

[1] **Note:** t_p, t_e and t_{me} represent the computational cost of a bilinear pairing, an exponentiation and a multi-exponentiation respectively, while t_s and t_v represent the computational cost of a one-time signature signing and verification respectively. $|\mathbb{G}|$, $|\mathbb{Z}_q|$, $|\mathbb{G}_e|$ and $|\mathbb{G}_T|$ denote the bit-length of an element ing groups $\mathbb{G}$, $\mathbb{Z}_q$, $\mathbb{G}_e$ and $\mathbb{G}_T$ respectively. Here $\mathbb{G}$ and $\mathbb{Z}_q$ denote the groups used in our scheme, while $\mathbb{G}_e$ and $\mathbb{G}_T$ are the bilinear groups used in CH scheme II, i.e., the bilinear pairing is $e : \mathbb{G}_e \times \mathbb{G}_e \to \mathbb{G}_T$. Finally, $|pk_s|$ and $|\sigma_s|$ denote the bit length of the one-time signature's public key and a one-time signature respectively.

check" key $(k_1x_1, k_1x_2, k_1y_1, k_1y_2)$, the delegatee can check that the ciphertext after the re-encryption is the original valid Cramer–Shoup ciphertext by "blinding check" key $(k_2x_1, k_2x_2, k_2y_2, k_2y_2)$, and he can also check ${u_1}'$ is indeed ${u_1}^{z/z'}$ by the tag $\alpha' = H(u_1, u_2, e, m)$, thus he can make sure that $m = e/({u_1}')^{z'}$ is indeed the original m which encrypted to the delegator. For the "special structure" of Cramer–Shoup ciphertext, the adversary can not get any help from the Decryption Oracle(Dec_1, Dec_2).

Assume adversary $\mathcal{B}$ can break the IND-CCA2 property of the scheme, then there exists an algorithm $\mathcal{A}$ which can distinguish whether a four tuple $(g_1, g_2, u_1, u_2) \in \mathbb{G}$ is a DDH tuple or not. $\mathcal{A}$ can answer the queries issued by $\mathcal{B}$ as following:

- **Key generation oracle:**
  - If user i is corrupted, $\mathcal{A}$ randomly chooses

    $sk_i = (x_{1i}, x_{2i}, y_{1i}, y_{2i}, k_{1i}, k_{2i}, z_i) \in Z_q$

    and a hash function H at random, then computes pk_i is

    $(g_1, g_2, c_i = g_1^{x_{1i}} g_2^{x_{2i}},$
    $d_i = g_1^{y_{1i}} g_2^{y_{2i}}, c_{i1} = g_1^{k_{1i}x_{1i}} g_2^{k_{1i}x_{2i}},$
    $d_{i1} = g_1^{k_{1i}y_{1i}} g_2^{k_{1i}y_{2i}}, c_{i2} = g_1^{k_{2i}x_{1i}} g_2^{k_{2i}x_{2i}},$
    $d_{i2} = g_1^{k_{2i}y_{1i}} g_2^{k_{2i}y_{2i}}, h_i = g_1^{z_i}, H),$

    returns pk_i and sk_i to the adversary.
  - If the user j is uncorrupted, $\mathcal{A}$ randomly chooses $sk_j = (x_{1j}, x_{2j}, y_{1j}, y_{2j}, k_{1j}, k_{2j}, z_{1j}, z_{2j}) \in Z_q$, it computes pk_j is

    $(g_1, g_2, c_j = g_1^{x_{1j}} g_2^{x_{2j}}, d_j = g_1^{y_{1j}} g_2^{y_{2j}},$
    $c_{j1} = g_1^{k_{1j}x_{1j}} g_2^{k_{1j}x_{2j}}, d_{j1} = g_1^{k_{1j}y_{1j}} g_2^{k_{1j}y_{2j}},$
    $c_{j2} = g_1^{k_{2j}x_{1j}} g_2^{k_{2j}x_{2j}}, d_{j2} = g_1^{k_{2j}y_{1j}} g_2^{k_{2j}y_{2j}},$
    $h_j = g_1^{z_{1j}} g_2^{z_{2j}}, H)$

    returns pk_j to the adversary. $\mathcal{A}$ preserves a User-Key-list of the form $(corrupted, i, sk_i, \mathsf{pk}_i, H)$ or $(uncorrupted, j, sk_j, pk_j, H)$.
- **Re-key generation oracle:**
  - On input (i, j), if one of i and j is uncorrupted and the other is corrupted, then this call is illegal.
  - Else if both i and j are uncorrupted, $\mathcal{A}$ runs basic Cramer–Shoup encryption algorithm to encrypt $(k_{2i}x_{1i}, k_{2i}x_{2i}, k_{2i}y_{1i}, k_{2i}y_{2i})$ under j's pubic key (g_1, g_2, c_j, d_j), denote the ciphertext as c_0^i. $\mathcal{A}$ also randomly chooses $t \in Z_q^*$ and outputs the re-encryption key

    $rk_{i\leftrightarrow j} = (k_{1i}x_{1i}, k_{1i}x_{2i}, k_{1i}y_{1i},$
    $k_{1i}y_{2i}, k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, t)$
  - Else if i and j are both corrupted, $\mathcal{A}$ outputs the re-encryption key

    $rk_{i\leftrightarrow j} = (k_{1i}x_{1i}, k_{1i}x_{2i}, k_{1i}y_{1i}, k_{1i}y_{2i}$
    $k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, z_i/z_j)$

    $\mathcal{A}$ preserves a Re-Key-list of the form

    $(uncorrupted, i, uncorrupted, j, k_{1i}x_{1i}, k_{1i}x_{2i},$
    $k_{1i}y_{1i}, k_{1i}y_{2i}, k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, t)$

    or

    $(corrupted, i, corrupted, j, k_{1i}x_{1i}, k_{1i}x_{2i},$
    $k_{1i}y_{1i}, k_{1i}y_{2i}, k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, z_i/z_j)$
- **Encryption oracle:**
  - If user i is uncorrupted, then given a message $m \in \mathbb{G}$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes

    $u_{1i} = g_1^r, u_{2i} = g_2^r, e_i = u_{1i}^{z_{1i}} u_{2i}^{z_{2i}} m,$
    $\alpha_i = H(u_{1i}, u_{2i}, e_i), v_i = c_i^r d_i^{r\alpha_i}, v_{2i} = c_{2i}^r d_{2i}^{r\alpha_i},$
    $\alpha_i' = H(u_{1i}, u_{2i}, e_i, u_{1i}^{z_{1i}} u_{2i}^{z_{2i}}, m),$
    $\alpha_{1i} = H(u_{1i}, u_{2i}, e_i, v_i, v_{2i}, \alpha_i'), v_{1i} = c_{1i}^r d_{1i}^{r\alpha_{1i}}$

    The ciphertext is

    $(u_{1i}, u_{2i}, e_i, v_i, v_{2i}, \alpha_i', v_{1i})$
  - If user j is corrupted, then given a message $m \in \mathbb{G}$, the encryption algorithm runs as follows. First, it chooses $r \in Z_q$ at random. Then it computes

    $u_{1j} = g_1^r, u_{2j} = g_2^r, e_j = u_{1j}^{z_j} m,$
    $\alpha_j = H(u_{1j}, u_{2j}, e_j), v_j = c_j^r d_j^{r\alpha_j},$
    $v_{2j} = c_{2j}^r d_{2j}^{r\alpha_j}, \alpha_j' = H(u_{1j}, u_{2j}, e_j, u_{1j}^{z_j}, m),$
    $\alpha_{1j} = H(u_{1j}, u_{2j}, e_j, v_j, v_{2j}, \alpha_j'), v_{1j} = c_{1j}^r d_{1j}^{r\alpha_{1j}}$

    The ciphertext is

    $(u_{1j}, u_{2j}, e_j, v_j, v_{2j}, \alpha_j', v_{1j})$
- **Re-encryption oracle:** on input

  $(u_{1i}, u_{2i}, e_i, v_i, v_{2i}, \alpha_i', v_{1i})$

  from user i to user j, $\mathcal{A}$ runs as following.
  - $\mathcal{A}$ searches the Re-Key-list list and if finding an item including i, j where i, j are both uncorrupted, then

1. $\mathcal{A}$ first verifies ciphertext

   $(u_{1i}, u_{2i}, e_i, v_i, v_{2i}, \alpha_i', v_{1i})$

   is valid or not. If $u_{1i}^{k_{1i}x_{1i}+k_{1i}y_{1i}\alpha_{1i}} u_{2i}^{k_{1i}x_{2i}+k_{1i}y_{2i}\alpha_{1i}} \neq v_{1i}$ where $\alpha_{1i} = H(u_{1i}, u_{2i}, e_i, v_i, v_{2i}, \alpha_i')$, then return "Reject".
2. Otherwise compute ${u_{1i}}' = {u_{1i}}^t$ and outputs $(u_{1i}, {u_{1i}}', u_{2i}, e_i, v_{2i}, \alpha_i', c_0^i)$ to j.

- if i, j are both corrupted, $\mathcal{A}$ do the same as the case of i, j are both uncorrupted except ${u_{1i}}' = {u_{1i}}^{z_i/z_j}$.
- If there is no such pair in the Re-Key-list, then return $\perp$.

- Dec_1**Oracle:** level 1 ciphertext is the normal ciphertext. On input $c = (u_{1j}, u_{2j}, e_j, v_j, v_{2j}, \alpha_j', v_{1j})$ to user j, $\mathcal{A}$ runs as following.
  - $\mathcal{A}$ searches the User-Key-list, if there is an item including $(j, uncorrupted)$, it tests if

    $u_{1j}^{x_{1j}+y_{1j}\alpha_j} u_{2j}^{x_{2j}+y_{2j}\alpha_j} = v_{1j}$

    where $\alpha_j = H(u_{1j}, u_{2j}, e_j)$ and

    $u_{1j}^{k_{1j}x_{1j}+k_{1j}y_{1j}\alpha_{1j}} u_{2j}^{k_{1j}x_{2j}+k_{1j}y_{2j}\alpha_{1j}} = v_{1j}$

    where $\alpha_{1j} = H(u_{1j}, u_{2j}, e_j, v_j, v_{2j}, \alpha_j')$ holds.

    1. If this condition does not hold, the decryption algorithm outputs "reject".
    2. Otherwise, it outputs $m = e_j / {u_{1j}}^{z_{1j}} {u_{2j}}^{z_{2j}}$,
  - If finding an item including $(j, corrupted)$, runs $Dec_1(sk_j, c)$.
  - If there is no such item in the User-Key-list, then returns $\perp$.

- Dec_2 **Oracle:** level 2 cipertext is the re-encryption ciphertext. On input $(u_{1i}, {u_{1i}}', u_{2i}, e_i, v_{2i}, \alpha_i', c_0^i)$ from user i to j by the proxy, $\mathcal{A}$ runs as following.
  - $\mathcal{A}$ first searches Re-Key-list and if there is an item including $(uncorrupted, i, uncorrupted, j)$, then
    1. $\mathcal{A}$ runs $Dec_1(sk_j, c_0^i)$, assume the plaintext is $(k_{2i}x_{1i}, k_{2i}x_{2i}, k_{2i}y_{2i}, k_{2i}y_{2i})$.
    2. If ${u_{1i}}^{k_{2i}x_{1i}+k_{2i}y_{1i}\alpha_i} {u_{2i}}^{k_{2i}x_{2i}+k_{2i}y_{2i}\alpha_i} \neq v_{2i}$ where $\alpha_i = H(u_{1i}, u_{2i}, e_i)$, then returns "Reject".
    3. Else if ${u_{1i}}' \neq {u_{1i}}^t$, returns "Reject".
    4. Otherwise computes $m = e_i / (u_{1i})^{z_{1i}} (u_{2i})^{z_{2i}}$ and returns m.
  - Else if there is an item including $(corrupted, i, corrupted, j)$, $\mathcal{A}$ does the same as above except computes $m = e_i / (u_{1i})^{z_i}$.
  - If there is no such item in the Re-Key-list, then returns $\perp$.

Analysis: next we show our oracle simulation is perfect.

- **Key Generation Oracle Simulation**.
  - For corrupted users, the simulated output (sk_i, pk_i) is an identical distribution to the real output.
  - For uncorrupted users, assuming $g_2 = g_1^w$, the simulated output (sk_j, pk_j) also is an identical distribution to the real output for $h_j = g_1^{z_1} g_2^{z_2} = g_1^{z_1 + wz_2} = g_1^{z'}$.
- **Re-key generation oracle simulation**.
  - When one of users (i,j) is corrupted, call oracle with input (i,j) is illegal and the output "$\perp$" is identical to the real output.
  - If users i and j are both uncorrupted, the simulated output $rk_{i \leftrightarrow j} = (k_{1i}x_{1i}, k_{1i}x_{2i}, k_{1i}y_{1i}, k_{1i}y_{2i}, k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, t)$ is indistinguishable with the real output $rk_{i \leftrightarrow j} = (k_{1i}x_{1i}, k_{1i}x_{2i}, k_{1i}y_{1i}, k_{1i}y_{2i}, k_{1j}x_{1j}, k_{1j}x_{2j}, k_{1j}y_{1j}, k_{1j}y_{2j}, c_0^i, z_i/z_j)$.
  - If users i and j are both corrupted, the simulated output is the same as the real output.
- **Encryption oracle simulation**. The difference between real encryption and simulated encryption is as following.
  - In real encryption $e = h^r m$ while in simulated encryption $e = u_1^{z_1} u_2^{z_2} m$.
  - If users i and j are both corrupted, the simulated output is same as the real output.
- **Re-encryption oracle simulation**. The difference between the real re-encryption and the simulated re-encryption is as following.
  - If (i,j) are both uncorrupted, ${u_{1i}}' = {u_{1i}}^t$ in simulated re-encryption while ${u_{1i}}' = {u_{1i}}^{z_i/z_j}$ in real re-encryption. The distributions can not be distinguished by the adversary.
  - If users i and j are both corrupted, the simulated output is same as the real output.
- Dec_1 **Oracle Simulation**. The difference between the real Dec_1 and the simulated Dec_1 is as following.
  - if i is uncorrupted, in simulated Dec_1 $m = \frac{e_i}{{u_{1i}}^{z_{1i}} {u_{2i}}^{z_{2i}}}$ while $m = e_i / (u_{1i})^{z_i}$ in real Dec_1. The two distributions can not be distinguished by the adversary from the upcoming first claim.
  - If users i and j are both corrupted, the simulated output is the same as the real output.
- Dec_2 **Oracle Simulation**. The difference between the real Dec_2 and the simulated Dec_2 is as following.

- If (i,j) are both uncorrupted, and if ${u_{1i}}' \neq {u_{1i}}^t$, in the real Dec_2, j first computes $m' = e_i/({u_{1i}})^{{}'z_j}$ and finds that $\alpha_i' = H(u_{1i}, u_{2i}, e_i, ({u_{1i}})^{{}'z_j}, m)$, then returns "reject"; while in the simulated Dec_2, j directly returns "reject". If ${u_{1i}}' = {u_{1i}}^t$, $m = \frac{e_i}{{u_{1i}}^{z_{1i}} {u_{2i}}^{z_{2i}}}$ in simulated Dec_2 while $m = e_i/({u_{1i}})^{{}'z_j} = e_i/({u_{1i}})^{z_i}$. The two distributions can not be distinguished by the adversary from the upcoming first claim.
- If users i and j are both corrupted, the simulated output is the same as the real output.

Denote distribution $\mathcal{R}$ as random quadruples $(g_1, g_2, u_1, u_2) \in \mathbb{G}^4$, distribution $\mathcal{D}$ as quadruples $(g_1, g_2, u_1, u_2) \in \mathbb{G}^4$, where g_1, g_2 are random, and $u_1 = g_1^r, u_2 = g_2^r$ for random $r \in Z_q$.

If the inputs comes from $\mathcal{D}$, the simulation will be nearly perfect, and the adversary $\mathcal{B}$ will have a non-negligible advantage in guessing the hidden b. But if the input comes from $\mathcal{R}$, the adversary $\mathcal{B}$'s view is essentially independent of b, and therefore the adversary $\mathcal{A}$'s advantage is negligible. This immediately implies a statistical test distinguish $\mathcal{R}$ from $\mathcal{D}$: run $\mathcal{A}$ and adversary $\mathcal{B}$, and if $\mathcal{A}$ outputs b and the adversary $\mathcal{B}$ outputs b', the distinguisher outputs 1 if $b = b'$, and 0 otherwise. We can get the following two lemmas.

Lemma 1 When $\mathcal{A}$'s input comes from $\mathcal{D}$, the joint distribution of the adversary $\mathcal{B}$'s view and the hidden bit b is statistically indistinguishable from that in the actual attack.

Proof Consider the joint distribution of the adversary's view and the bit b when the input comes from the distribution $\mathcal{D}$. Say $u_1 = g_1^r$ and $u_2 = g_2^r$.

It is clear in this case that the output of the encryption oracle has the right distribution, since $u_1^{x_1} u_2^{x_2} = c^r$, $u_1^{y_1} u_2^{y_2} = d^r$, $u_1^{k_1 x_1} u_2^{k_1 x_2} = c_1^r$, $u_1^{k_1 y_1} u_2^{k_1 y_2} = d_1^r$, $u_1^{k_2 x_1} u_2^{k_2 x_2} = c_2^r$, $u_1^{k_2 y_1} u_2^{k_2 y_2} = d_2^r$ and $u_1^{z_1} u_2^{z_2} = h^r$. Actually, these equations imply that $e = m_b h^r$ and $v = c^r d^{r\alpha}$, $v_2 = c^r d^{r\alpha}$, $v_1 = c^r d^{r\alpha_1}$ where α, α_1 and α' themselves are already of the right form.

To complete the proof, we need to argue that the output of the decryption oracle has the right distribution. Let us call $(u_1', u_2', e', v', v_2', v_1', \alpha')$ a valid ciphertext if $\log_{g_1}^{u_1'} \neq \log_{g_2}^{u_2'}$. □

Note that if a ciphertext is valid, with $u_1' = g_1^{r'}$ and $u_2' = g_2^{r'}$, then $h^{r'} = (u_1')^{z_1}(u_2')^{z_2}$. Therefore, the Dec_1 and Dec_2 oracles output $e/h^{r'}$, just as they should. Consequently, the lemma follows immediately from the following:

Claim The decryption oracle in both an actual attack against the cryptosystem and in an attack against the simulator rejects all invalid ciphertexts, except with negligible probability.

We now prove this claim by considering the distribution of the point $\mathbf{P} = (x_1, x_2, y_1, y_2)$, conditioned on the adversary's view. Let $\log(\cdot)$ denote $\log_{g_1}(\cdot)$, and $w = \log g_2$.

From the adversary's view, $\mathbf{P}$ is a random point on the plane $\mathbf{P}$ formed by intersecting the hyperplane

$$\log c = x_1 + w x_2 \tag{3}$$

$$\log d = y_1 + w y_2 \tag{4}$$

$$\log c_1 = k_1 x_1 + w k_1 x_2 \tag{5}$$

$$\log d_1 = k_1 y_1 + w k_1 y_2 \tag{6}$$

$$\log c_2 = k_2 x_1 + w k_2 x_2 \tag{7}$$

$$\log d_2 = k_2 y_1 + w k_2 y_2 \tag{8}$$

These equations come from the public key. The output from the encryption oracle does not constrain $\mathbf{P}$ any further, as the hyperplane defined by

$$\log v = r x_1 + w r x_2 + \alpha r y_1 + \alpha r w y_2 \tag{9}$$

$$\log v_2 = r k_2 x_1 + w r k_2 x_2 + \alpha r k_2 y_1 + \alpha w r k_2 y_2 \tag{10}$$

$$\log v_1 = r k_1 x_1 + w r k_1 x_2 + \alpha_1 r k_1 y_1 + \alpha_1 w r k_1 y_2 \tag{11}$$

contains $\mathcal{P}$.

Now suppose the adversary submits an invalid ciphertext $(u_1', u_2', e', v', v_2', \alpha', v_1')$ to the Dec_1 oracle, where $\log u_1' = r_1'$ and $\log u_2' = w r_2'$, with $r_1' \neq r_2'$. The Dec_1 oracle will output reject, unless $\mathbf{P}$ happens to lie on the hyperplane $\mathcal{H}$ defined by

$$\log v' = r_1' x_1 + w r_2' x_2 + \alpha_0' r_1' y_1 + \alpha_0' r_2' w y_2 \tag{12}$$

$$\log v_1' = r_1' k_1 x_1 + w r_2' k_1 x_2 + \alpha_1' r_1' k_1 y_1 + \alpha_1' r_2' w k_1 y_2 \tag{13}$$

where $\alpha_0' = H(u_1', u_2', e)$ and $\alpha_1' = H(u_1', u_2', e', v', v_2', \alpha')$. But it is clear that the Eqs. (3) (4) and (12), (5) (6) and (13) are linearly independent and so $\mathcal{H}$ intersects the plane $\mathcal{P}$ at a line.

Or suppose the adversary submits an invalid ciphertext $(u_1', u_2', e', v_2', \alpha', c_0' = ({(u_1^0)}', {(u_2^0)}', {(e^0)}', {(v^0)}'))$ to the Dec_2 oracle, where $\log u_1' = r_1'$ and $\log u_2' = w r_2'$, with $r_1' \neq r_2'$. The Dec_1 oracle will reject, unless the below (14)–(18) equations are satisfied:

$${(v^0)}' = ({(u_1^0)}')^{x_1' + y_1' ({\alpha^0})'} ({(u_2^0)}')^{x_2' + y_2' ({\alpha^0})'} \tag{14}$$

$$(k_2 x_1, k_2 x_2, k_2 y_1, k_2 y_2) = {(e^0)}'/({(u_1^0)}')^{z'} \tag{15}$$

$$\log v_2' = r_1' k_2 x_1 + w r_2' k_2 x_2 + {\alpha_0}' r_1' k_2 y_1 + {\alpha_0}' r_2' w k_2 y_2 \tag{16}$$

$$m = e'/(u_1')^{z'} \tag{17}$$

$$\alpha' = H(u_1', u_2', e', (u_1')^{z'}, m) \tag{18}$$

where $\alpha_0' = H(u_1', u_2', e')$ and $(\alpha^0)' = H((u_1^0)', (u_2^0)', (e^0)')$. The adversary can construct $(u_1', u_2', e', v_2', \alpha', c_0')$ such that

$$(v^0)' = ((u_1^0)')^{x_1'+y_1'(\alpha^0)'}((u_2^0)')^{x_2'+y_2'(\alpha^0)'} \tag{19}$$

$$(k_2x_1, \left(\frac{r_1'}{r_2'}\right)k_2x_2, k_2y_1, \left(\frac{r_1'}{r_2'}\right)k_2y_2) = (e^0)'/(u_1^0)'^{z'} \tag{20}$$

$$\log v_2' = r_1'k_2x_1 + wr_1'k_2x_2 + \alpha_0{}'r_1'k_2y_1 + \alpha_0{}'r_1'wk_2y_2 \tag{21}$$

$$m = e'/(u_1')^{z'} \tag{22}$$

$$\alpha' = H(u_1', u_2', e', m) \tag{23}$$

is negligible because m is unknown. And it is clear that the Eqs. (7) (8) and (16) are linearly independent, so the probability of accepting is negligible.

It follows that the first time the adversary submits an invalid ciphertext, the Dec_1 oracle rejects with probability $1 - 1/q^2$ and the Dec_2 oracle rejects with probability $1 - 1/q$. This rejection actually constrains the point $\mathbf{P}$, puncturing the plane $\mathcal{H}$ at a line. Therefore, for $i = 1, 2, \ldots$, the ith invalid ciphertext submitted by the adversary will be rejected with probability at least $1 - 1/(q^3 - i + 1)$. From this it follows that the decryption oracle rejects all invalid ciphertexts, except with negligible probability.

Lemma 2 When $\mathcal{A}$'s input comes from $\mathcal{R}$, the distribution of the hidden bit b is (essentially) independent from the adversary $\mathcal{B}$'s view.

Proof Let $u_1 = g_1^{r_1}$ and $u_2 = g_1^{wr_2}$. We can assume that $r_1 \neq r_2$, since this occurs except with negligible probability. The lemma follows immediately from the following two claims. □

Claim If the decryption oracle rejects all invalid ciphertexts during the attack, then the distribution of the hidden bit b is independent of the adversary's view.

To see this, consider the point $\mathbf{Q} = (z_1, z_2) \in Z_q^2$. At the beginning of the attack, this is a random point on the line

$$\log h = z_1 + wz_2 \tag{24}$$

determined by the public key. Moreover, if the decryption oracle only decrypts valid ciphertexts $(u_1', u_2', e', v', v_2', \alpha', v_1')$ or $(u_1', u_2', e', v_2', \alpha', c_0')$, then the adversary obtains only linearly dependent relations $r' \log h = r'z_1 + r'wz_2$ (since $(u_1')^{z_1}(u_2')^{z_2} = g_1^{r'z_1}g_2^{r'z_2} = h^{r'}$). Thus, no further information about $\mathbf{Q}$ is leaked.

Consider now the output $(u_1, u_2, e, v, v_2, \alpha, v_1)$ of the simulator's encryption oracle. We have $e = \epsilon \cdot m_b$, where $\epsilon = u_1^{z_1}u_2^{z_2}$. Now consider the equation

$$\log \epsilon = r_1z_1 + wr_2z_2 \tag{25}$$

Clearly, (24) and (25) are linearly independent, and so the conditional distribution of ϵ (conditioning on b and everything in the adversary's view other than e) is uniform. In other words, ϵ is a perfect one-time pad. It follows that b is independent of the adversary's view.

Claim The decryption oracle will reject all invalid ciphertexts, except with negligible probability.

As in the proof of Lemma 1, we study the distribution of $\mathbf{P} = (x_1, x_2, y_1, y_2)$, conditioned on the adversary's view. From the adversary's view, this is a random point on the line $\mathcal{L}$ formed by intersecting the hyperplanes (3), (4) and

$$\log v = r_1x_1 + wr_2x_2 + \alpha r_1y_1 + \alpha r_2wy_2 \tag{26}$$

$$\log v_2 = r_1k_2x_1 + wr_2k_2x_2 + \alpha r_1k_2y_1 + \alpha wr_2k_2y_2 \tag{27}$$

$$\log v_1 = r_1k_1x_1 + wr_2k_1x_2 + \alpha_1r_1k_1y_1 + \alpha_1wr_2k_1y_2 \tag{28}$$

Equations (26)–(28) come from the output of the encryption oracle.

Now assume that the adversary submits an invalid ciphertext $(u_1', u_2', e', v', v_2', \alpha'', v_1') \neq (u_1, u_2, e, v, v_2, \alpha', v_1)$, where $\log u_1' = r_1'$ and $\log u_2' = wr_2'$, with $r_1' \neq r_2'$. Let $\alpha_0' = H(u_1', u_2', e')$ and $\alpha_1' = H(u_1', u_2', e', v', v_2', \alpha')$.

There are three cases we consider.

- *Case 1.* $(u_1', u_2', e') = (u_1, u_2, e)$ or $(u_1', u_2', e', v', v_2', \alpha'') = (u_1, u_2, e, v, v_2, \alpha')$. In this case, the hash values are the same, but $v' \neq v$, $v_2' \neq v_2$ or $v_1' \neq v_1$ implies that the Dec_1 oracle or Dec_2 oracle will certainly output reject.
- *Case 2.* $(u_1', u_2', e') \neq (u_1, u_2, e)$ and $\alpha_0' \neq \alpha$ or $(u_1', u_2', e', v', v_2', \alpha'') = (u_1, u_2, e, v, v_2, \alpha')$ and $\alpha_1' \neq \alpha_1$. The Dec_1 oracle or Dec_2 oracle will output reject unless the point $\mathbf{P}$ lies on the hyperplane $\mathcal{H}$ defined by (12) (13) (16). However, the Eqs. (3) (4) (12) (26), (5) (6) (16) (27) and (7) (8) (13) (28) are linearly independent. This can be verified by observing the determinant of the following three matrixes are not equal 0.

$$\begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1' & wr_2' & \alpha_0'r_1' & \alpha_0'r_2'w \\ r_1 & wr_2 & \alpha r_1 & \alpha r_2w \end{pmatrix}$$

$$\begin{pmatrix} k_2 & k_2w & 0 & 0 \\ 0 & 0 & k_2 & k_2w \\ r_1'k_2 & wr_2'k_2 & \alpha_0'r_1'k_2 & \alpha_0'r_2'wk_2 \\ r_1k_2 & wr_2k_2 & \alpha r_1k_2 & \alpha r_2wk_2 \end{pmatrix}$$

$$\begin{pmatrix} k_1 & k_1w & 0 & 0 \\ 0 & 0 & k_1 & k_1w \\ r_1'k_1 & wr_2'k_1 & \alpha_1'r_1'k_1 & \alpha_1'r_2'wk_1 \\ r_1k_1 & wr_2k_1 & \alpha_1r_1k_1 & \alpha_1r_2wk_1 \end{pmatrix}$$

Thus, $\mathcal{H}$ intersects the line $\mathcal{L}$ at a point, from which it follows (as in the proof of Lemma 1) that the decryption oracle outputs reject, except with negligible probability.

- *Case 3.* $(u_1', u_2', e') \neq (u_1, u_2, e)$ and $\alpha_0' = \alpha$ or $(u_1', u_2', e', v', v_2', \alpha'') \neq (u_1, u_2, e, v, v_2, \alpha')$ and $\alpha_1' = \alpha_1$. This will result collisions, which is contradict with our assumption about hash function's collision resistance.

Thus we prove the main Theorem.

Remark 1 Why our scheme can only achieve IND-CCA security in the relatively weak model instead of in the normal model? That is because in our scheme, the proxy knows rk is $(k_1x_1, k_1x_2, k_1y_1, k_1y_2, k_1'x_1', k_1'x_2', k_1'y_1', k_1'y_2', c_0, c_0', z/z')$ and the delegatee knows $(k_2x_1, k_2x_2, k_2y_1, k_2y_2)$, thus they will know the partial information of x_1, x_2, y_1, y_2 (for example, the value of $x_1/x_2, y_2/y_1$ etc.) Therefore, our scheme can not achieve the delegator's IND-CCA security for the proxy and the delegatee, but it can achieve IND-CCA security for any outside adversaries.

5 Conclusion

In this paper, we first discuss our scheme's application in protecting the security of modern critical systems, especially on its implementation without the secret keys feature. As we all know, the key management is very complex and critical in modern information systems among critical infrastructures, such as billions of embedded control equipments of the cars, the household appliances, ATMs, smart meter for measuring electricity etc. Thus we believe our proposal is useful for smoothly running these applications.

Then we propose a concrete PRE scheme which partially solved an open problem proposed in Deng et al. (2008). That is, how to construct an IND-CCA2 secure proxy re-encryption without pairing in the standard model. We achieved this goal by constructing an IND-CCA2 secure proxy re-encryption based on Cramer–Shoup encryption. We propose an IND-CCA2 secure proxy re-encryption scheme $\prod_{ST}$ based on Cramer–Shoup encryption. Compared with the other IND-CCA2 secure proxy re-encryption scheme (CH II Scheme) in the standard model Canetti and Hohenberger (2007), the computation cost of our scheme is much more efficient due to not relying on pairings. However, we note that although $\prod_{ST}$ can be only proved secure in standard model under a relatively weak model. How to improve the scheme to be secure in the strong model is our future work.

Acknowledgments This work was supported by Natural Science Foundation of Shaanxi Province (Grant No. 2014JM8300), the Changjiang Scholars and Innovation Research Team in University (Grant NO. IRT 1078), the Key Problem of NFSC-Guangdong Union Foundation (Grant NO. U1135002), the Major Nature Science Foundation of China (Grant NO. 61370078), China 863 project, the Fundamental Research Funds for the Center Universities (Grant NO. JY10000903001), Nature Science Foundation of China (Grant NO. 61103230).

References

Abelson H, Anderson R, Bellovin SM, Benaloh J, Blaze M, Diffie W, Gilmore J, Neumann PG, Rivest RL, Schiller JI, Schneier B (1997) The risks of key recovery, key escrow, and trusted third-party encryption. World Wide Web J (Web Security: A Matter of Trust) 2, 3. O'Reilly Associates, Summer, pp 241–257

Ateniese G, Fu K, Green M, Hohenberger S (2005) Improved proxy re-encryption schemes with applications to secure distributed storage. NDSS pp 29–43

Ateniese G, Fu K, Green M, Hohenberger S (2006) Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans Inf Syst Secur 9(1):1–30

Bellare M, Rogaway P (1997) Collision-resistant hashing: towards making UOWHFs practical. In: Advances in Cryptology-Crypto'97. Springer, Berlin

Blaze M, Bleumer G, Strauss M (1998) Divertible protocols and atomic proxy cryptography. In: Advances in Cryptology-Eurocrypt'98. LNCS, vol 1403. Springer, Berlin, pp 127–144

Clark D, Earl Boebert W, Gerhart S, Guttag J, Kemmerer R, Kent S, Mann Lambert M, Lampson W, Lane J, McIlroy MD, Neumann PG, Rabin MO, Schmitt W, Tipton HF, Walker ST, Ware WH (1996) Computers at risk: safe computing in the information age. In: National Research Council, National Academy Press, Washington, pp 20418

Cramer R, Shoup V (1998) A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Advances in Cryptology-Crypto'98. LNCS, vol 1462. Springer, Berlin, pp 13–25

Cramer R, Shoup V (2003) Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J Comput 33:167–226

Canetti R, Goldwasser S (1999) An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Advances in Cryptology-Eurocrypt'99. LNCS, vol 1592. Springer, Berlin, pp 90–106

Canetti R, Halevi S, Katz J (2003) A forward-secure public-key encryption scheme. In: Advances in cryptology-EUROCRYPT'03. LNCS, vol 2656. Springer, Berlin, pp 255–271

Canetti R, Hohenberger S (2007) Chosen ciphertext secure proxy re-encryption. In: Proceedings of the 14th ACM conference on computer and communications security (CCS 2007), pp 185–194

Deng R, Weng J, Liu S, Chen K (2008) Chosen ciphertext secure proxy re-encryption without pairing. In: CANS'08. LNCS, vol 5339. Springer, Berlin, pp 1–17

Kurosawa K, Desmedt Y (2004) A new paradigm of hybrid encryption scheme. In: Crypto'04. LNCS, vol 3152. Springer, Berlin, pp 426–442

Kiltz E, Galindo D (2006) Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In: Cryptology ePrint Archive, Report 2006/034. http://eprint.iacr.org/

Kiltz E (2006) Chosen-ciphertext secure identity-based encryption in the standard model with short ciphertexts. In: Cryptology ePrint Archive, Report 2006/122. http://eprint.iacr.org/

Li J, Chen X, Li M, Li J, Lee P, Lou W (2014) Secure deduplication with efficient and reliable convergent key management. IEEE Trans Parallel Distrib Syst 25(6):1615–1625

Li J, Kim K (2010) Hidden attribute-based signatures without anonymity revocation. Inf Sci 180(9):1681–1689 (Elsevier)

Li J, Wang Q, Wang C, Ren K (2011) Enhancing attribute-based encryption with attribute hierarchy. Mobile Networks and Applications (MONET) 16(5):553–561 (Springer-Verlag)

Libert B, Vergnaud D (2008) Unidirectional chosen-ciphertext secure proxy re-encryption. In: 11th International workshop on practice and theory in public key cryptography (PKC) 2008. LNCS, vol 4939. Springer, Berlin, pp 360–379

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

Spaho E, Sakamoto S, Barolli L, Xhafa F, Ikeda M (2014) Trustworthiness in P2P: performance behaviour of two fuzzy-based systems for JXTA-overlay platform. Soft Comput 18(9):1783–1793

Solhaug B, Seehusen F (2014) Model-driven risk analysis of evolving critical infrastructures. J Ambient Intell Humaniz Comput 5(2):187–204

Xhafa F, Wang J, Chen X, Liu JK, Li J, Krause P (2014) An efficient PHR service system supporting fuzzy keyword search and fine-grained access control. Soft Comput 18(9):1795–1802

Yao C, Xu L, Huang X, Liu JK (2014) A secure remote data integrity checking cloud storage system from threshold encryption. J Ambient Intell Humaniz Comput 5(6):857–865