Secure Bidirectional Proxy Re-Encryption for Cryptographic Cloud Storage[☆]

Jun Shao[a], Rongxing Lu[b], Xiaodong Lin[c], Kaitai Liang[d]

[a] *School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, Zhejiang, P.R. China*
[b] *School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798*
[c] *Faculty of Business and Information Technology, University of Ontario Institute of Technology, Ontario, Canada*
[d] *Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong*

Abstract

Bidirectional proxy re-encryption allows ciphertext transformation between Alice and Bob via a semi-trusted proxy, who however cannot obtain the corresponding plaintext. Due to this special property, bidirectional proxy re-encryption has become a flexible tool in many dynamic environments, such as cryptographic cloud storage. Nonetheless, how to design a secure and efficient bidirectional proxy re-encryption is still challenging. In this paper, we propose a novel bidirectional proxy re-encryption scheme that holds the following properties: 1) constant ciphertext size no matter how many times the transformation is performed; 2) master secret security in the random oracle model, i.e., Alice (resp. Bob) colluding with the proxy cannot obtain Bob's (resp. Alice's) private key; 3) replayable chosen ciphertext (RCCA) security in the random oracle model. The above three properties are usually required in the cryptographic cloud storage. Furthermore, the proposed new master secret security may be of independent interest, as it is closer to the original desire: delegate the decryption rights while keeping the signing rights.

Keywords: bidirectional proxy re-encryption, replayable chosen-ciphertext attack, master secret security, multi-use, constant size, cryptographic cloud storage

1. Introduction

Proxy re-encryption (PRE) [2] allows a secure ciphertext transformation in a way that a semi-trusted proxy can use a re-encryption key delegated from Alice (and Bob) to re-encrypt a ciphertext under Alice's public key into a new ciphertext that Bob can decrypt by using his own private key. However, the proxy cannot do any decryption on the ciphertexts of either Alice or Bob. If the re-encryption key can be used to do the re-encryption in both directions, the PRE scheme is called bidirectional; otherwise, it is called unidirectional. Both types of PRE schemes have their own interesting applications. In this work, we shall focus on bidirectional proxy re-encryption (BPRE), as it still encounters many research challenges when applied to practical scenarios.

Among the applications of BPRE [10, 14, 8, 20, 17, 18, 11, 9, 16], the cryptographic cloud storage (sharing) [20, 16] has become more and more popular. This kind of application usually works in a dynamic environment which requires the PRE scheme to hold multi-usability and constant ciphertext size. In other words, it demands that the transformed ciphertext can be further transformed while the ciphertext size keeps the same.

To the best of our knowledge, there are only few BPRE schemes [2, 6, 13, 19] satisfying the above requirements. However, those previously reported schemes in [2, 6, 13] suffer from the so-called collusion

[☆] The extended abstract of this paper appeared at Provsec 2014.

Email addresses: chn.junshao@gmail.com (Jun Shao), rxlu@ntu.edu.sg (Rongxing Lu), xiaodong.lin@uoit.ca (Xiaodong Lin), kliang4-c@my.cityu.edu.hk (Kaitai Liang)

attack, i.e., Alice (resp. Bob) colluding with the proxy can obtain Bob's (resp. Alice's) private key. In practice, collusion resistance is crucial, especially when Alice (resp. Bob) uses the same private key to perform decrypting and signing, and she (resp. he) wants to delegate the decryption rights while keeping the signing rights. In the applications of cryptographic cloud storage (sharing), the cloud server (acting as the proxy in the BPRE scheme) is assumed to be not colluding with any user in the system. However, as we know, this assumption in the reality is not always true.

In general, the security notion dealing with the collusion attack is called master secret security proposed by Ateniese et al. [1]. Recently, Weng and Zhao [19] proposed two BPRE schemes based on pairings. One is multi-use but with only CPA secure, the other is CCA secure but not multi-use. Meanwhile, it has been showed that replayable chosen ciphertext (RCCA) security is also crucial in the applications of distributed storage [6]. Therefore, in this paper, to address the above challenges, we would like to propose the first scheme with multi-useability, constant ciphertext size, and RCCA security. The proposed BPRE scheme in this paper can (partially) solve the problem in the cryptographic cloud storage (sharing) mentioned above. Furthermore, the proposed BPRE scheme satisfies our new master secret security where Alice (resp. Bob) colluding with the proxy cannot sign messages on behalf of Bob (resp. Alice). This new definition is closer to the original desire for the master secret security, compared to the existing master secret security where Alice (resp. Bob) colluding with the proxy cannot obtain Bob's (resp. Alice's) private key.

1.1. Main Differences between the Conference Version and the Current Version

The main difference between the conference version [12] and the current version is the security model for the master secret security. In the conference version, the security model only captures the attacks aiming at computing private key, while the security model in the current version captures the attacks aiming at the forgery on signatures. The security obtained in the latter model is stronger than that in the former model, and the latter one is closer to the original desire for the master secret security.

To coordinate the new security model, we made the following changes in this current version.

- We added the definition of Auxiliary Digital Signature to the definition part.
- We added the signing oracle into the security games of the RCCA security and master secret security, and changed the winning requirements in the security game of the master secret security.
- We added a concrete auxiliary digital signature scheme.
- We gave the new security proofs in the new security models.

1.2. Related Work

At EUROCRYPT 1998, Blaze, Bleumer and Strauss [2] proposed the first BPRE scheme (named `BBS98`) base on ElGamal encryption [7]. Later on, Canetti and Hohenberger [6] proposed the first (R)CCA-secure BPRE scheme (named `CH07`) by using pairings. At PKC 2011, Matsuda, Nishimaki and Tanaka [13] proposed a new pairing-free CPA-secure bidirectional scheme (named `MNT10`). All of the above schemes hold multi-usability and constant ciphertext size, but they all suffer from the collusion attack. The main reason that the collusion attack works is that the re-encryption key is computed by sk_A/sk_B, where sk_A and sk_B are the private keys of Alice and Bob, respectively. It is easy to see that once sk_A (resp. sk_B) and sk_A/sk_B are put together, sk_B (resp. sk_A) would be revealed.

Recently, Weng and Zhao [19] proposed two new BPRE schemes by using pairings. The first one (named `WZ11a`) is multi-use, CPA-secure and master secret secure (in the sense of the old definition), and the second one (named `WZ11b`) is multi-use, CCA-secure, and master secret secure (in the sense of the old definition). To obtain master secret security, the re-encryption key is computed by sk'_A/sk'_B, where sk'_A and sk'_B are not the private keys but the decryption keys of Alice and Bob, respectively. The analogous relations between sk_A and sk'_A can be found in the identity-based encryption [15, 3], where the private key generator's master secret key and the user's private key can be considered as the analogies sk_A and sk'_A, respectively. Clearly, knowing sk'_A does not hurt the secrecy of sk_A.

Table 1: Summary of bidirectional proxy re-encryption schemes.

	`BBS98`[2]	`CH07`[6]	`MNT10`[13]	`WZ11a`[19]	`WZ11b`[19]	Our proposal
RCCA **CA**[a]	✗ DDH	✓ DBDH	✗ re-applicable LTDFs	✗ DBDH	✓ 1-AwDBDHI	✓ mDBDH
MSS[b] **CA**	✗ —	✗ —	✗ —	—[c] —	—[d] —	✓ CDH
MU[e]	✓	✓	✓	✓	✗	✓

In Table 1, we summarize the existing BPRE schemes in terms of secrecy of message, master secret security, multi useability and their complexity assumptions. From this table, we can see that our proposal would be the only one that can hold the desired properties at the same time.

The rest of this paper is organized as follows. In Section 2, we give the definitions, security models of BPRE. Our new definition of the master secret security is given in this section. Then, we present our proposal in Section 3, including the description, security analysis and computation comparison. Finally, we draw our conclusions in Section 4.

2. Definitions

2.1. Definition of Bidirectional Proxy Re-Encryption

Definition 1 (Bidirectional Proxy Re-Encryption). A Bidirectional proxy re-encryption scheme is a tuple of probabilistic polynomial time (PPT) algorithms (`KeyGen`, `ReKeyGen`, `Enc`, `ReEnc`, `Dec`):

- `KeyGen`$(1^\kappa) \to (pk, sk)$. On input of the security parameter 1^κ, the key generation algorithm `KeyGen` outputs a public key and private key pair (pk, sk).

- `ReKeyGen`$(sk_1, sk_2) \to rk_{1,2}$. On input of two private keys sk_1 and sk_2, the re-encryption key generation algorithm `ReKeyGen` outputs a bidirectional re-encryption key $rk_{1,2}$. Since it is bidirectional, `ReKeyGen`$(sk_2,\ sk_1)$ can be easily computed from `ReKeyGen`(sk_1, sk_2) via a public function $\mathcal{F}$. For instance, $\mathcal{F}$ is the inversion of `ReKeyGen`(sk_1, sk_2) in [2, 6, 19].

  In general speaking, this algorithm is interactive, involving the delegator, delegatee and proxy.

- `Enc`$(pk, m) \to C$. On input of a message m from the message space and a public key pk, the encryption algorithm `Enc` outputs a ciphertext C.

- `ReEnc`$(rk_{1,2}, C_1) \to C_2$. On input of a re-encryption key $rk_{1,2}$ and a ciphertext C_1, the re-encryption algorithm `ReEnc` outputs a re-encrypted ciphertext C_2 or a special symbol `reject`.

- `Dec`$(sk, C) \to m$. On input of a private key sk, and a ciphertext C, the decryption algorithm `Dec` outputs a message m or a special symbol `reject`.

[a] CA denotes "Complexity Assumption".

[b] MSS denotes "Master Secret Security".

[c] It is unclear whether it can be proved secure in our new security model.

[d] It is unclear whether it can be proved secure in our new security model.

[e] MU denotes "Multi-Useability".

Correctness of BPRE. For any message m in the message space and any key pairs $(pk, sk), (pk', sk') \leftarrow \texttt{KeyGen}(1^\kappa)$, the following two conditions must hold:

$$\texttt{Dec}(sk, \texttt{Enc}(pk, m)) = m \text{ and } \texttt{Dec}(sk', \texttt{ReEnc}(rk, C)) = m,$$

where rk is generated by $\texttt{ReKeyGen}(sk, sk')$ or $\mathcal{F}(\texttt{ReKeyGen}(sk', sk))$, and C is the ciphertext for message m under pk from algorithm `Enc` or `ReEnc` if the bidirectional proxy re-encryption scheme is multi-use.

Recall the original desire for the master secret security: delegate decryption rights while keeping the signing rights. To give the definition of the master secret security closer to the original desire, we need to define the auxiliary digital signature for BPRE.

Definition 2 (Auxiliary Digital Signature). An auxiliary digital signature (ADS) for BPRE contains the following PPT algorithms (`KeyGen`, `Sign`, `Verify`).

- $\texttt{KeyGen}(1^\kappa) \rightarrow (pk, sk)$. Identical to that in BPRE.
- $\texttt{Sign}(sk, m) \rightarrow \sigma$. On input of a private key sk and a message m from the message space, the signing algorithm `Sign` outputs a signature σ.
- $\texttt{Verify}(pk, m, \sigma) \rightarrow 0/1$. On input of a public key pk, a message m from the message space and a signature σ, the verifying algorithm `Verify` outputs 1 if σ is valid or 0 otherwise.

Correctness of ADS. The correctness of ADS is the same as that the traditional digital signature. That is, for any message m in the message space and any key pairs $(pk, sk) \leftarrow \texttt{KeyGen}(1^\kappa)$, the following two conditions must hold:

$$\texttt{Verify}(pk, m, \texttt{Sign}(sk, m)) = 1 \text{ and } \texttt{Verify}(pk, m', \texttt{Sign}(sk, m)) = 0$$

where $m' \neq m$.

2.2. Replayable Chosen Ciphertext Security for Multi-Use Bidirectional Proxy Re-Encryption

The replayable chosen ciphertext security for multi-use BPRE is defined by the following RCCA game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. As usual, the challenger $\mathcal{C}$ does not answer any queries which the adversary $\mathcal{A}$ can answer by itself using the secret it has been supplied, and returns only one answer for the same query. Moreover, the adversary $\mathcal{A}$ should decide which party will be corrupted before the game starts. In other words, our RCCA security is defined in the static model.

Setup: The challenger $\mathcal{C}$ sets up the system parameters, and initializes one empty table $\mathbb{T}_k$ which will be used to record all key pairs.

Phase 1: The adversary $\mathcal{A}$ can issue the following queries adaptively.

- *Public key generation oracle* $\mathcal{O}_{pk}$: $\mathcal{C}$ takes a security parameter 1^κ, runs $\texttt{KeyGen}(1^\kappa)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in the table $\mathbb{T}_k$. In the following, sk_i is the corresponding private key of pk_i.
- *Private key generation oracle* $\mathcal{O}_{sk}$: On input of pk_i by $\mathcal{A}$, $\mathcal{C}$ searches for pk_i in the table $\mathbb{T}_k$, and returns sk_i if pk_i is corrupted.
- *Re-encryption key generation oracle* $\mathcal{O}_{rk}$: On input of two different public keys (pk_i, pk_j) by $\mathcal{A}$, $\mathcal{C}$ returns the re-encryption key $rk_{i,j} = \texttt{ReKeyGen}(sk_i, sk_j)$. It is required that both pk_i and pk_j are corrupted or uncorrupted.
- *Re-encryption oracle* $\mathcal{O}_{re}$: On input of (pk_i, pk_j, C) by $\mathcal{A}$, $\mathcal{C}$ returns the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk_i, sk_j), C)$.
- *Decryption oracle* $\mathcal{O}_{dec}$: On input (pk_i, C_i), $\mathcal{C}$ returns $\texttt{Dec}(sk_i, C_i)$.

- *Signing oracle $\mathcal{O}_{sign}$*: On input of (pk_i, m_i), $\mathcal{C}$ returns $\texttt{Sign}(sk_i, m_i)$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs two equal length plaintexts m_0^*, m_1^* from the message space, and an uncorrupted public key pk^* on which it wishes to challenge. $\mathcal{C}$ picks a random bit $\mathbf{b} \in \{0,1\}$ and sets $C^* = \texttt{Enc}(pk^*, m_{\mathbf{b}}^*)$. It sends C^* as the challenge to $\mathcal{A}$.

Phase 2: This phase is almost the same as Phase 1 but with the following restrictions.

- $\mathcal{O}_{re}$: On input of (pk_1, pk_2, C_1) by $\mathcal{A}$, if (pk_1, C_1) is a derivative of (pk^*, C^*), and pk_2 is corrupted, $\mathcal{C}$ outputs $\texttt{reject}$. We say (pk_1, C_1) is a derivative of (pk^*, C^*) if one of the following conditions holds.
  - $(pk_1, C_1) = (pk^*, C^*)$.
  - (pk, C) is a derivative of (pk^*, C^*), and (pk_1, C_1) is a derivative of (pk, C).
  - $(pk_1, C_1) \leftarrow \mathcal{O}_{re}(pk, pk_1, C)$, where (pk, C) is a derivative of (pk^*, C^*).
  - The adversary A can use the re-encryption keys from $\mathcal{O}_{rk}$ to transform ciphertexts under pk^* to that under pk by running $\texttt{ReEnc}$, and $\mathcal{O}_{dec}(pk_1, C_1) \in \{m_0^*, m_1^*\}$.
- $\mathcal{O}_{dec}$: On input of (pk_i, C_i), if the output is m_0^* or m_1^*, $\mathcal{C}$ returns $\texttt{reject}$.

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

Remark 1. *Compared to the existing security game of confidentiality for PRE, our security game additionally contains the signing oracle. It is because that in our definition of BPRE, any BPRE scheme always has an ADS scheme.*

We refer to such an adversary $\mathcal{A}$ as an RCCA adversary. We define adversary $\mathcal{A}$'s advantage in attacking multi-use BPRE as the following function of the security parameter κ: $\mathbf{Adv}_{\texttt{MBPRE}}^{\texttt{RCCA}}(1^\kappa) = |\Pr[\mathbf{b} = \mathbf{b}'] - 1/2|$. Using the RCCA game, we can define RCCA security of multi-use BPRE.

Definition 3 (RCCA Security). If for any PPT RCCA adversary $\mathcal{A}$ the function $\mathbf{Adv}_{\texttt{MBPRE}}^{\texttt{RCCA}}(1^\kappa)$ is negligible, the multi-use bidirectional proxy re-encryption scheme is RCCA-secure.

2.3. Master Secret Security for Multi-Use Bidirectional Proxy Re-Encryption.

The master secret security for multi-use BPRE is defined by the following MSS game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. As usual, the challenger $\mathcal{C}$ does not answer any queries which the adversary $\mathcal{A}$ can answer by itself using the secret it has been supplied, and returns only one answer for the same query. Moreover, all the public keys involved in the following oracles (except $\mathcal{O}_{pk}$) are from $\mathcal{O}_{pk}$. It is worth mentioning that in the MSS game, the adversary does not need to decide the corrupted user before the game starts. In other words, our master secret security is defined in the adaptive model.

Find: The adversary $\mathcal{A}$ can issue the following queries adaptively.

- *Public key generation oracle $\mathcal{O}_{pk}$*: $\mathcal{C}$ takes a security parameter 1^κ, runs $\texttt{KeyGen}(1^\kappa)$ to generate a key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$ and records (pk_i, sk_i) in the table $\mathbb{T}_k$. In the following, sk_i is the corresponding private key of pk_i.
- *Private key generation oracle $\mathcal{O}_{sk}$*: On input of pk_i by $\mathcal{A}$, $\mathcal{C}$ searches for pk_i in the table $\mathbb{T}_k$, and returns sk_i.
- *Re-encryption key generation oracle $\mathcal{O}_{rk}$*: On input of (pk_i, pk_j) by $\mathcal{A}$, $\mathcal{C}$ returns the re-encryption key $rk_{i,j} = \texttt{ReKeyGen}(sk_i, sk_j)$.
- *Re-encryption oracle $\mathcal{O}_{re}$*: On input of (pk_i, pk_j, C) by $\mathcal{A}$, $\mathcal{C}$ returns the re-encrypted ciphertext $C' = \texttt{ReEnc}(\texttt{ReKeyGen}(sk_i, sk_j), C)$.
- *Decryption oracle $\mathcal{O}_{dec}$*: On input of (pk_i, C_i), $\mathcal{C}$ returns $\texttt{Dec}(sk_i, C_i)$.

- *Signing oracle* $\mathcal{O}_{sign}$: On input of (pk_i, m_i), $\mathcal{C}$ returns $\mathtt{Sign}(sk_i, m_i)$.

Output: $\mathcal{A}$ outputs a tuple of (pk^*, m^*, σ^*). If all the following conditions are satisfied, then $\mathcal{A}$ wins the game.

- $\mathtt{Verify}(pk^*, m^*, \sigma^*) = 1$.
- pk^* has never been queried to $\mathcal{O}_{sk}$.
- (pk^*, m^*) has never been queried to $\mathcal{O}_{sign}$.

Remark 2. *Compared to the existing security game for the master secret security, our security game additionally contains the signing oracle, and the winning requirement changes from outputting a private key of an uncorrupted user to a successful forgery of the signature.*

We also define $\mathbf{Adv}^{\mathtt{MSS}}_{\mathtt{MBPRE}}(1^\kappa) = \Pr[\mathcal{A}\ \mathtt{Wins}]$ for the security parameter κ as that in RCCA security.

Definition 4 (Master Secret Security). If for any PPT MSS adversary $\mathcal{A}$ the function $\mathbf{Adv}^{\mathtt{MSS}}_{\mathtt{MBPRE}}(1^\kappa)$ is negligible, the multi-use bidirectional proxy re-encryption scheme is MS-secure.

2.4. Bilinear Groups

In this subsection, we briefly review the definitions about bilinear maps and bilinear map groups, which follow those in [3, 5].

1. $\mathbb{G}$ and $\mathbb{G}_T$ are two (multiplicative) cyclic groups of prime order q;
2. g is a generator of $\mathbb{G}$;
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two groups as above. An *admissible bilinear map* is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

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

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

2.5. Complexity Assumptions

The security of our proposal is based on the modified decisional bilinear Diffie-Hellman assumption and computational Diffie-Hellman assumption. Since the latter assumption is well-known, we only give the definition of the former assumption in this paper.

Definition 5 (Modified Decisional Bilinear Diffie-Hellman Assumption). Let $(q, g, \mathbb{G}, \mathbb{G}_T, e) \leftarrow \mathtt{BSetup}(1^k)$. The modified decisional bilinear Diffie-Hellman problem (mDBDH) in $(\mathbb{G}, \mathbb{G}_T)$ is defined as follows: given tuple $(g, g^a, g^{1/a}, g^b, g^c, T) \in \mathbb{G}^5 \times \mathbb{G}_T$ as input, decide whether $S = \hat{e}(g, g)^{abc}$. An algorithm $\mathcal{A}$ has advantage ε in solving the mDBDH problem in $(\mathbb{G}, \mathbb{G}_T)$ if

$$|\Pr[\mathcal{A}(g, g^a, g^{1/a}, g^b, g^c, \hat{e}(g, g)^{abc}) = 0] - \Pr[\mathcal{A}(g, g^a, g^{1/a}, g^b, g^c, T) = 0]| \geq \epsilon,$$

where the probability is taken over the random choices of $a, b, c \in \mathbb{Z}_q$, $S \in \mathbb{G}$ and the random bits of $\mathcal{A}$.

We say that the (t, ϵ)-modified decisional Bilinear Diffie-Hellman assumption holds in $(\mathbb{G}, \mathbb{G}_T)$ if no t-time algorithm has advantage ϵ at least in solving the mDBDH problem in $(\mathbb{G}, \mathbb{G}_T)$.

3. Our Proposal

3.1. Description of Our BPRE Scheme

The system parameters are $(q, g, h, e, \mathbb{G}, \mathbb{G}_T)$, where $(q, g, e, \mathbb{G}, \mathbb{G}_T)$ are from `BSetup`, and h is a random element from $\mathbb{G}$. Furthermore, it requires two secure cryptographic hash functions $H_1 : \{0,1\}^* \to \{0,1\}^\kappa$ and $H_2 : \{0,1\}^* \to \{0,1\}^\kappa$, where κ is a security parameter.

- `KeyGen`: Select random $x_1, x_2, z_1 \in \mathbb{Z}_q$, Next, compute $X_{1,g} = \hat{e}(g,g)^{x_1}$, $X_{1,h} = \hat{e}(g,h)^{x_1}$, $X_2 = g^{x_2}$, $Z_1 = g^{z_1}$, $z_2 = (x_1 - z_1)/x_2 \bmod q$, and $Z_2 = g^{z_2}$. The public key is
$$pk = (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2),$$
and the private key is
$$sk = (x_1, x_2, z_1, z_2).$$

- `ReKeyGen`: On input two private keys $sk = (x_1, x_2, z_1, z_2)$ and $sk' = (x_1', x_2', z_1', z_2')$, it outputs the re-encryption key
$$rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q).$$
The re-encryption key can be computed efficiently by the method in [2, 6].

- `Enc`: On input $pk = (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2)$ and $m \in \mathbb{G}_T$, do the following steps.
  - Choose random r from $\mathbb{Z}_q$.
  - Compute
  $$u_1 = g^r, \quad u_2 = X_2^r, \quad v = X_{1,h}^r \cdot m, \quad u_3 = X_{1,g}^r,$$
  $$u_4 = H_1(v||u_3)^r, \quad u_5 = H_2(v||u_3)^r$$
  Note that the item u_5 is only useful in the security proof.
  - Output $C = (u_1, u_2, v, u_3, u_4, u_5)$ as the ciphertext.

- `ReEnc`: On input a re-encryption key $rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q)$ and a ciphertext $C = (u_1, u_2, v, u_3, u_4, u_5)$ under key $pk = (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2)$, the proxy performs as follows.
  - Check whether
  $$\begin{aligned} \hat{e}(u_1, H_1(v||u_3)) &= \hat{e}(g, u_4) && (1) \\ \hat{e}(u_1, H_2(v||u_3)) &= \hat{e}(g, u_5) && (2) \\ \hat{e}(u_1, Z_1) \cdot \hat{e}(u_2, Z_2) &= u_3 && (3) \end{aligned}$$
  all hold. If not, abort; otherwise, do the next steps.
  - Compute $u_1' = u_1^{rk_1}$, $u_2' = u_2^{rk_2}$, $u_4' = u_4^{rk_1}$, and $u_5' = u_5^{rk_1}$.
  - Output $(u_1', u_2', v, u_3, u_4', u_5')$ as the re-encrypted ciphertext.

  Note that
  $$u_1' = u_1^{rk_1} = g^{r \cdot z_1/z_1'}, \quad u_2' = u_2^{rk_2} = g^{r \cdot x_2 \cdot z_2/z_2'}$$
  $$u_4' = u_4^{rk_1} = H_1(v||u_3)^{r \cdot z_1/z_1'}, \quad u_5' = u_5^{rk_2} = H_2(v||u_3)^{r \cdot z_1/z_1'}$$

- `Dec`: On input a private key (x_1, x_2, z_1, z_2) and any ciphertext $C = (u_1, u_2, v, u_3, u_4, u_5)$, the decryptor performs as follows.
  - Check whether Equalities (1), (2), (3) all hold. If not, abort; otherwise, do the next steps.

– Compute $m = v/(\hat{e}(u_1^{z_1}, h)\cdot \hat{e}(u_2^{z_2}, h))$. Note that if the ciphertext C is from `Enc`, we have that

$$v/(\hat{e}(u_1^{z_1}, h)\cdot \hat{e}(u_2^{z_2}, h)) = \hat{e}(h, g^{x_1\cdot r})\cdot m/(\hat{e}(h, g^{z_1\cdot r})\cdot \hat{e}(h, g^{z_2\cdot x_2\cdot r})) = m;$$

if the ciphertext C is from `ReEnc`, we have that

$$v/(\hat{e}(u_1^{z_1}, h)\cdot \hat{e}(u_2^{z_2}, h)) = \hat{e}(h, g^{x'_1\cdot r})\cdot m/(\hat{e}(h, g^{z'_1\cdot r})\cdot \hat{e}(h, g^{z'_2\cdot x'_2\cdot r})) = m.$$

– Output the message m.

The correctness of the above BPRE scheme can be easily obtained by the description.

3.2. Description of Our ADS Scheme

The system parameters are almost the same as that in our BPRE scheme, except we need an additional secure cryptographic hash function $H_3 : \{0,1\}^* \rightarrow \mathbb{Z}_q$.

- `KeyGen`: Identical to that in our BPRE scheme.
- `Sign`: On input a private key $sk = (x_1, x_2, z_1, z_2)$ and a message m, it outputs the signature $\sigma = H_3(m||pk)^{x_2}$, where pk is the public key corresponding to sk.
- `Verify`: On input $pk = (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2)$, a signature σ and a message m, if $\hat{e}(X_2, H_3(m||pk)) = \hat{e}(g, \sigma)$ holds, it outputs 1 or 0 otherwise.

It is easy to see that the above ADS scheme is the short signature scheme in [4], hence the correctness is obtained.

3.3. Security Analysis

Theorem 1. *If the mDBDH assumption holds in $\mathbb{G}$, our proposal is RCCA-secure in the random oracle model. In particular, we have*

$$\mathbf{Adv}^{\texttt{RCCA}}_{\texttt{MBPRE}}(1^\kappa) \leq \mathbf{Adv}^{\mathcal{A}}_{\texttt{mDBDH}}(1^\kappa),$$

where $\mathbf{Adv}^{\mathcal{A}}_{\texttt{mDBDH}}(1^\kappa)$ is the advantage of that $\mathcal{A}$ breaks the mDBDH assumption under the security parameter κ.

Proof. Assume there exists a RCCA adversary $\mathcal{A}$ that can break the RCCA security of our proposal. Then we can build another algorithm $\mathcal{B}$ that can break the mDBDH assumption (i.e., given $g, g^a, g^{1/a}, g^b, g^c, T$, it is hard to decide $T = \hat{e}(g,g)^{abc}$) by playing the RCCA game with $\mathcal{A}$. The details are as follows. Before the game starts, $\mathcal{B}$ sets $h = g^b$.

H_1 **Oracle:** On input of (v, u_3), check whether the tuple (v, u_3, α_1) exists in the list L_{H_1}. If yes, return $(g^a)^{\alpha_1}$; otherwise, it chooses a random α_1 from $\mathbb{Z}_q$, and then records (v, u_3, α_1) in the list L_{H_1}, and returns $(g^a)^{\alpha_1}$. Note that if the input (v, u_3) is a part of the challenge ciphertext, then $\mathcal{B}$ just returns g^{α_1}.

H_2 **Oracle:** On input of (v, u_3), check whether the tuple (v, u_3, α_2) exists in the list L_{H_2}. If yes, return $(g^{1/a})^{\alpha_2}$; otherwise, it chooses a random α_2 from $\mathbb{Z}_q$, and then records (v, u_3, α_2) in the list L_{H_2}, and return $(g^{1/a})^{\alpha_2}$. Note that if the input (v, u_3) is a part of the challenge ciphertext, then $\mathcal{B}$ just returns g^{α_2}.

H_3 **Oracle:** On input of (m, pk), check whether the tuple (m, pk, α_3) exists in the list L_{H_3}. If yes, return g^{α_3}; otherwise, choose a random value α_3 from $\mathbb{Z}_q$, and then record (m, pk, α_3) in the list L_{H_3}, and return g^{α_3}.

Phase 1: $\mathcal{B}$ builds the oracles as follows.

- $\mathcal{O}_{pk}$: $\mathcal{B}$ chooses random elements z_1, z_2, x_2 from $\mathbb{Z}_q$. If the public key is uncorrupted, then $\mathcal{B}$ computes $X_1 = g^{z_1}((g^a)^{x_2})^{z_2}$ and returns

$$\begin{aligned} pk &= (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2) \\ &= (\hat{e}(g, X_1), \hat{e}(h, X_1), g^{x_2}, g^{z_1}, (g^a)^{z_2}); \end{aligned}$$

  if the public key is corrupted, then $\mathcal{B}$ computes $X_1 = g^{z_1}(g^{x_2})^{z_2}$ and returns

$$\begin{aligned} pk &= (X_{1,g}, X_{1,h}, X_2, Z_1, Z_2) \\ &= (\hat{e}(g, X_1), \hat{e}(h, X_1), g^{x_2}, g^{z_1}, g^{z_2}). \end{aligned}$$

  At last, $\mathcal{B}$ records (pk, X_1, z_1, z_2, x_2) in $\mathbb{T}_k$.
- $\mathcal{O}_{sk}$: On input of a corrupted public key pk by $\mathcal{A}$, $\mathcal{B}$ gets (pk, z_1, z_2, x_2) from $\mathbb{T}_k$. $\mathcal{B}$ returns $sk = (z_1 + x_2 \cdot z_2 \bmod q, x_2, z_1, z_2)$.
- $\mathcal{O}_{rk}$: On input of (pk, pk') by $\mathcal{A}$, $\mathcal{B}$ gets (pk, z_1, z_2, x_2) and (pk', z_1', z_2', x_2') from $\mathbb{T}_k$, and then returns

$$rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q).$$

  Note that we have the following for the correctness.
  - If the two public keys are both uncorrupted,

$$sk = (z_1 + x_2 \cdot a \cdot z_2 \bmod q, x_2, z_1, z_2 \cdot a \bmod q)$$

    and

$$sk' = (z_1' + x_2' \cdot a \cdot z_2' \bmod q, a \cdot x_2', z_1', z_2' \cdot a \bmod q),$$

    hence, $rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q)$.
  - If the two public keys are both corrupted,

$$sk = (z_1 + x_2 \cdot z_2 \bmod q, x_2, z_1, z_2)$$

    and

$$sk' = (z_1' + x_2' \cdot z_2' \bmod q, x_2', z_1', z_2'),$$

    hence, $rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q)$.
- $\mathcal{O}_{re}$: On input of $C = (u_1, u_2, v, u_3, u_4, u_5)$ and two public keys pk and pk', $\mathcal{B}$ first checks the well-formness as the real execution. If it does not pass, abort; otherwise, do the followings.
  - If the two public keys are both uncorrupted or corrupted, then $\mathcal{B}$ gets the re-encryption key from $\mathcal{O}_{rk}$, and returns the result of `ReEnc`.
  - Otherwise, do the followings.
    - Find the items (pk, X_1, z_1, z_2, x_2) and $(pk', X_1', z_1', z_2', x_2')$ in $\mathbb{T}_k$.
    - Compute $u_1' = u_1^{z_1/z_2}$, $u_4' = u_4^{z_1/z_2}$, and $u_5' = u_5^{z_1/z_2}$.
    - Find the original decryptor pk_o of the ciphertext C by checking $\hat{e}(u_1, X_1) = u_3$ for all items in $\mathbb{T}_k$. Assume that the found item is $(pk_o, X_{o1}, z_{o1}, z_{o2}, x_{o2})$.
    - Compute $g^r = u_1^{z_1/z_{o1}}$, $(g^a)^r = (u_4^{z_1/z_{o1}})^{1/\alpha_1}$ and $(g^{1/a})^r = (u_5^{z_1/z_{o1}})^{1/\alpha_2}$, where α_1, α_2 are the values in L_{H_1} and L_{H_2} corresponding to (v, u_3), respectively.
    - If pk_o and pk' are both corrupted or uncorrupted, then compute $u_2' = (g^r)^{x_{o2} \cdot z_{o2}/z_2}$. If pk_o is corrupted and pk' is uncorrupted, then compute $u_2' = ((g^{1/a})^r)^{x_{o2} \cdot z_{o2}/z_2}$. If pk_o is uncorrupted and pk' is corrupted, then compute $u_2' = ((g^a)^r)^{x_{o2} \cdot z_{o2}/z_2}$.
    - Output $(u_1', u_2', v, u_3, u_4', u_5')$ as the resultant ciphertext.

- $\mathcal{O}_{dec}$: On input of $C = (u_1, u_2, v, u_3, u_4)$ under pk, $\mathcal{B}$ first checks the well-formness as the real execution. If it does not pass, output $\perp$; otherwise, re-encrypt the ciphertext to the one under a corrupted public key by querying $\mathcal{O}_{re}$, then decrypt the resultant ciphertext by using the corresponding private key.
- $\mathcal{O}_{sign}$: On input of (m, pk), $\mathcal{B}$ firstly gets the value of α_3 by querying H_3 hash oracle, and then returns $X_2^{\alpha_3}$.

Challenge: $\mathcal{A}$ sends $\mathcal{C}$ two messages m_0, m_1 from $\mathbb{G}$ and an uncorrupted public key

$$\begin{aligned} pk^* &= (X_{1,g}^*, X_{1,h}^*, X_2^*, Z_1^*, Z_2^*) \\ &= (\hat{e}(g, g^{z_1^*}(g^a)^{z_2^*}), \hat{e}(h, g^{z_1^*}(g^a)^{z_2^*}), g^{x_2^*}, g^{z_1^*}, (g^a)^{z_2^*}), \end{aligned}$$

$\mathcal{B}$ chooses a random number $\mathbf{b} \in \{0, 1\}$, and returns $(u_1^*, u_2^*, v^*, u_3^*, u_4^*, u_5^*)$ as the challenge ciphertext.

$$u_1^* = g^c, \quad u_2^* = (g^b)^{x_2^*}, \quad v^* = \hat{e}(g^b, g^c)^{z_1^*} \cdot T^{x_2^* \cdot z_2^*} \cdot m_{\mathbf{b}},$$

$$u_3^* = \hat{e}(X_1^*, g^c), \quad u_4 = (g^b)^{\alpha_1^*}, \quad u_4 = (g^b)^{\alpha_2^*},$$

where $(X_1^*, z_1^*, z_2^*, x_2^*)$, α_1^* and α_2^* are from $\mathbb{T}_k$, L_{H_1} and L_{H_2}, respectively.

Note that if $T = g^{abc}$, we have that $v^* = \hat{e}(g^b, g^c)^{z_1^*} \cdot T^{x_2^* \cdot z_2^*} \cdot m_{\mathbf{b}} = \hat{e}(g^b, X_1^*)^c \cdot m_{\mathbf{b}}$.

Phase 2: Almost the same as that in Phase 1, except the restrictions in the RCCA game.

Guess: $\mathcal{A}$ outputs the guess $\mathbf{b}'$. If $\mathbf{b}' = \mathbf{b}$, $\mathcal{B}$ decides $T = \hat{e}(g, g)^{abc}$; otherwise, $T \neq \hat{e}(g, g)^{abc}$.

It is easy to see that in the random oracle, the above simulation is perfect. Hence, we obtain this theorem.□

Theorem 2. *If the CDH assumption holds in $\mathbb{G}$, our proposal is MS-secure in the random oracle model. In particular, we have*

$$\mathbf{Adv}_{\mathtt{MBPRE}}^{\mathtt{MSS}}(1^\kappa) \leq \frac{\mathbf{Adv}_{\mathtt{CDH}}^{\mathcal{A}}(1^\kappa)}{e(1 + q_{sk})q_{H_3}},$$

where $\mathbf{Adv}_{\mathtt{CDH}}^{\mathcal{A}}(1^\kappa)$ is the advantage of that $\mathcal{A}$ breaks the CDH assumption under the security parameter κ, and q_{sk} and q_{H_3} is the number of private key generation queries and H_3 hash oracle queries, respectively.

Proof. Assume there exists an MSS adversary $\mathcal{A}$ that can break the MS security of our proposal. Then we can build another algorithm $\mathcal{B}$ that can break the CDH assumption (i.e., given g, g^a, g^b, it is hard to compute g^{ab}) by playing the MSS game with $\mathcal{A}$. The details are as follows.

Find: $\mathcal{B}$ builds the oracles as follows.

- H_1, H_2 hash oracles: Identical to the proof of Theorem 1.
- H_3 hash oracle: On input of (m, pk), check whether the tuple (m, pk, α_3) exists in the list L_{H_3}. If yes, return g^{α_3}. Otherwise, it decides whether (m, pk) is the target (message, public key) pair. If yes, return g^b; otherwise, choose a random value α_3 from $\mathbb{Z}_q$, and then record (m, pk, α_3) in the list L_{H_3}, and return g^{α_3}.
- $\mathcal{O}_{pk}$: $\mathcal{B}$ chooses random elements z_1, z_2, x_2 from $\mathbb{Z}_q$ and decides $\theta \in \{0, 1\}$ under $\Pr[\theta = 1] = \delta$. If $\theta = 0$, then $\mathcal{B}$ returns

  $$\begin{aligned} pk &= (X_{1,g}, X_{1,h}, X_1, Z_1, Z_2) \\ &= (\hat{e}(g, g^{z_1}((g^a)^{x_2})^{z_2}), \hat{e}(h, g^{z_1}((g^a)^{x_2})^{z_2}), (g^a)^{x_2}, g^{z_1}, g^{z_2}); \end{aligned}$$

  if $\theta = 1$, then $\mathcal{B}$ returns

  $$\begin{aligned} pk &= (X_{1,g}, X_{1,h}, X_1, Z_1, Z_2) \\ &= (\hat{e}(g, g^{z_1}(g^{x_2})^{z_2}), \hat{e}(h, g^{z_1}(g^{x_2})^{z_2}), g^{x_2}, g^{z_1}, g^{z_2}) \end{aligned}$$

  At last, $\mathcal{B}$ records $(pk, z_1, z_2, x_2, \theta)$ in $\mathbb{T}_k$.

- $\mathcal{O}_{sk}$: On input of a public key pk by $\mathcal{A}$, $\mathcal{B}$ gets $(pk, z_1, z_2, x_2, \theta)$ from $\mathbb{T}_k$. If $\theta = 1$, then $\mathcal{B}$ returns $sk = (z_1 + x_2 \cdot z_2 \bmod q, x_2, z_1, z_2)$; otherwise, $\mathcal{B}$ outputs `failure`.
- $\mathcal{O}_{rk}$: On input of (pk, pk') by $\mathcal{A}$, $\mathcal{B}$ gets $(pk, z_1, z_2, x_2, \theta)$ and $(pk', z_1', z_2', x_2', \theta')$ from $\mathbb{T}_k$, and then returns $rk = (rk_1, rk_2) = (z_1/z_1' \bmod q, z_2/z_2' \bmod q)$.
- $\mathcal{O}_{re}$: $\mathcal{B}$ can use the re-encryption keys from $\mathcal{O}_{rk}$ to reply the queries.
- $\mathcal{O}_{dec}$: Identical to the proof of Theorem 1.
- $\mathcal{O}_{sign}$: $\mathcal{B}$ firstly gets the value of α_3 by querying H_3 hash oracle, and then returns $X_2^{\alpha_3}$.

Output: $\mathcal{A}$ outputs a signature $\sigma^* = H_3(m^*||pk^*)^{a \cdot x_2^*}$, then $\mathcal{B}$ outputs the CDH solution ${\sigma^*}^{1/x_2^*}$, where x_2^* is the value in $\mathbb{T}_k$ corresponding to pk^*.

If $\mathcal{B}$ has guessed the right target (message, public key) pair and has not output `failure`, then the above simulation is perfect. On one hand, $\mathcal{B}$ has guessed the right target pair with the probability of $1/q_{H_3}$ at least. On the other hand, the probability of that $\mathcal{B}$ has not output `failure` is $\delta^{q_{sk}}(1-\delta)$. The maximize value of $\delta^{q_{sk}}(1-\delta)$ is $1/(e(1+q_{sk}))$ when $\delta = 1 - 1/(q_{sk}+1)$. Hence, we obtain this theorem. □

4. Conclusion

In this paper, we have proposed a novel multi-use, bidirectional proxy re-encryption with constant ciphertext size, master secret security and RCCA security. This proposed BPRE scheme can provide a (partial) solution to the collusion attacks in the current cryptographic cloud storage. Another contribution in this paper is that we propose a new definition for the master secret security. The new security definition is closer to the original desire for the master secret security compared to the existing definitions. There are still various future works left, e.g., how to design a pairing-free, multi-use BPRE scheme with constant ciphertext size, master secret security, and RCCA security is still unknown.

Acknowledgements

The authors thank the anonymous reviewers for their insightful comments and helpful suggestions. In addition, this work was also supported by the National Basic Research Program of China (973 Program) No. 2012CB315804, NSFC Nos. 61472364, 61379121, and NSFZJ No. LR13F020003. Rongxing Lu would also like to thank the support of Nanyang Technological University under Grant NTU-SUG (M4081196) and MOE Tier 1 (M4011177).

[1] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*. The Internet Society, 2005.

[2] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer, 1998.

[3] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 231–229, 2001.

[4] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532, 2001.

[5] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.

[6] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 185–194. ACM, 2007.

[7] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[8] Chun-Ying Huang, Yun-Peng Chiu, Kuan-Ta Chen, and Chin-Laung Lei. Secure multicast in dynamic environments. *Computer Networks (Amsterdam, Netherlands: 1999)*, 51(10):2805–2817, July 2007.

[9] Junbeom Hur. Improving Security and Efficiency in Attribute-Based Data Sharing. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[10] Apu Kapadia, Patrick P. Tsang, and Sean W. Smith. Attribute-based publishing with hidden credentials and hidden policies. In *NDSS*. The Internet Society, 2007.

[11] Hsiao-Ying Lin and Wen-Guey Tzeng. A secure erasure code-based cloud storage system with secure data forwarding. *IEEE Trans. Parallel Distrib. Syst*, 23(6):995–1003, 2012.

[12] Rongxing Lu, Xiaodong Lin, Jun Shao, and Kaitai Liang. Rcca-secure multi-use bidirectional proxy re-encryption with master secret security. In *Provable Security - 8th International Conference, ProvSec 2014, Hong Kong, China, October 9-10, 2014. Proceedings*, pages 194–205, 2014.

[13] Toshihide Matsuda, Ryo Nishimaki, and Keisuke Tanaka. Cca proxy re-encryption without bilinear maps in the standard model. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2010.

[14] Ritesh Mukherjee and J. William Atwood. Scalable solutions for secure group communications. *Computer Networks (Amsterdam, Netherlands: 1999)*, 51(12):3525–3548, August 2007.

[15] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53, 1984.

[16] Jun Shao, Rongxing Lu, and Xiaodong Lin. FINE: A fine-grained privacy-preserving location-based service framework for mobile devices. In *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*, pages 244–252, 2014.

[17] Yipin Sun, Rongxing Lu, Xiaodong Lin, Xuemin Shen, and Jinshu Su. An Efficient Pseudonymous Authentication Scheme With Strong Privacy Preservation for Vehicular Communications. *IEEE Transactions on Vehicular Technology*, 59(7):3589–3603, 2010.

[18] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30(5), 2011.

[19] Jian Weng and Yunlei Zhao. Direct constructions of bidirectional proxy re-encryption with alleviated trust in proxy. *IACR Cryptology ePrint Archive*, 2011:208, 2011.

[20] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, pages 534–542. IEEE, 2010.