On the Security of a Cloud-Based Revocable IBPRE Scheme for Data Sharing

Jindan Zhang, Baocang Wang

Abstract In ESORICS2014, Liang et al. proposed an efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing, aimed at both supporting user revocation and delegation of decryption rights. The main strategy is to let the cloud periodic re-encrypt ciphertexts under the current time period to the next time period. If the user is revoked in the forth coming time period, he cannot decrypt the ciphertexts by using the expired private key anymore. Compared with traditional revocation technique by using PKG, this method has the advantage of computation and communication efficiency. However, in this paper we show an attack which allow the revoked user can decrypt the ciphertexts under the future time period, if the revoked users colludes with the proxy. Although cloud-based revocable identity based proxy re-encryption is a great idea for public cloud storage sharing, it needs further research before this scheme can be practically adapted.

1 Introduction

Nowadays it is a very popular method to store personal or business files on the cloud for its very cheap cost and instant accessing everywhere/everytime. Cloud service providers can manage the hardware and software for cloud storage very flexible and efficient in a cost-effective way. However there are some issues need to be solved before cloud storage can be adapted more widely. Roughly speaking, there are two main kinds of obstacles for cloud storage.

Jindan Zhang
State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China, e-mail: 69957106@qq.com

Baocang Wang
State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China, e-mail: bcwang79@aliyun.com

F. Xhafa et al. (eds.), *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*, Lecture Notes on Data Engineering and Communications Technologies 1, DOI 10.1007/978-3-319-49109-7_38

The first one is the bandwidth, although there are rapid development on the communication bandwidth these years such as we witness the growing from 2G, 3G to 4G, several years ago the personal average bandwidth is about 30kb while now it is about 20Mb, the growth rate is about 1:200. However this growth rate for cloud storage is still a great problem. Please note the data generation and growing speed is more sharply. Several years ago average a person has 100MB datum but now this amount is 1TB, the growth rate is about 1:10000. Although we shall use 5G technique in the near future, the bandwidth shall be still a problem compared with the burst growing of datum.

The second one is the security and usability, most users prefer to upload their datum to the cloud only if their files can be sure to be protected well. Most users require the controllability of their uploaded files, or more better, the visibility of their files. However now most cloud storage services can not do this well. In the past when users store their file in their own computers or storage equipments, they can be sure their files being secure for only themselves control on the equipments, there are some physical boundaries on the file's security. However if we upload the files to the cloud, the physical boundaries are not exist anymore, every one's files are virtually organized in the cloud servers, and the attacker can break in the servers to get other person's files or launch the side channel attack on other person's files. If the uploaded files can not be used anywhere/anytime/anyplace, then cloud storage will be no more attractive, thus usability is also an important concern for cloud storage service. Furthermore, how to share files with others is another important usage for cloud storage, if the data owners can not easily and securely share their files with others, lots of effective business corporation among many parties can not be implemented, which will hurdle cloud computation's widely adapted.

To solve these challenges, especially the security problem in cloud storage, we need develop some novel techniques such as PDP, POR, verifiable computation etc. Here we focus on one basic requirement for cloud storage, how to share files securely and flexibly. Proxy re-encryption is a primitive used to share files securely by using cryptographic techniques [1] [2]. In proxy re-encryption [3] [4], a file encrypted under delegator's public key can be re-encrypted by the delegatee's public key by the proxy, without the proxy knowing the underlying plaintexts or secret keys. For the cloud has very powerful computation ability and is semi-trusted, thus it can be the proxy for re-encryption. Thus until now there are many work on the construction of proxy re-encryption [18] [5] [7, 8] [12] [13] [14–17] and how to use proxy re-encryption to share the files with others [9–11], these works are very impressive. In 2001, Boneh and Franklin proposed the first practical identity based encryption scheme, which can reduce the burden caused by public certificates via public key infrastructure. By utilizing the benefits of identity based encryption and proxy re-encryption, a new primitive called identity based proxy re-encryption [6] has been proposed and used for sharing cloud data securely. Recently, Liang et al. proposed a cloud-based revocable identity based proxy re-encryption scheme, which can not only support ciphertext re-encryption, but also support revocation of cloud users, which is a basic requirement for cloud data sharing. However in this paper, we show their scheme can not resist the collusion attack, that is, if the revoked users collude

with the proxy, they can derive some secret information (master private key) which can decrypt any ciphertext in any period. Thus although we think cloud-based revocable identity based proxy re-encryption is a great idea for public cloud storage sharing, it needs further research before this scheme can be practically adapted.

2 Definition and Security Model

2.1 Definition

Definition 1. A Cloud-Based Revocable Identity-Based Proxy Re-Encryption (CR-IB-PRE) scheme consists of the following algorithms. Below we let $\mathscr{I}, \mathscr{T}, \mathscr{M}$ be identity space, time space and message space, respectively.

1. Setup: the setup algorithm intakes a security parameter k and a maximal number of users N, and outputs the public parameters mpk, the master secret key msk, the initial state st and an empty revocation list RL. For simplicity, we assume the following algorithms include mpk implicitly.
2. KeyGen: the private key generation algorithm intakes msk, and a users identity $id \in I$, and outputs a private key sk_{id} for the user id and an updated state st.
3. TokenUp: the token update algorithm intakes msk, an identity id, a token update time period $T_i \in \mathscr{T}$, the current revocation list RL and st, and outputs a token τ_i, where $i \in [1, poly(1^k)]$.
4. DeKeyGen: the decryption key generation algorithm intakes sk_{id}, τ_i, and outputs a decryption key $sk_{id|i}$ for the user id under the time period T_i or $\perp$, if id has been revoked, where $i \in [1, poly(1^k)]$.
5. ReKeyGen: the re-encryption key generation algorithm intakes $sk_{id|i}$, msk, T_i and T_i', and generates the re-encryption key as follows, where $1 \leq i \leq i'$.
   a. ReKeyToken: The re-encryption key token generation algorithm intakes msk, T_i, $T_{i'}$, outputs a re-encryption key token $\psi_{i \to i'}$.
   b. ReKey: the re-encryption key algorithm intakes $sk_{id|i}$ and $\psi_{i \to i'}$, outputs a re-encryption key $rk_{id \to id'}$ which can be used to transform a ciphertext under (id, T_i) to another ciphertext under $(id, T_{i'})$.
6. Enc: the encryption algorithm intakes id, T_i, and a message $m \in \mathscr{M}$ and outputs an original ciphertext C under (id, T_i) which can be further re-encrypted.
7. ReEnc: the re-encryption algorithm intakes $rk_{id|i \to i'}$, and a ciphertext C under (id, T_i), and outputs either a re-encrypted ciphertext C under $(id, T_{i'})$ or a symbol $\perp$ indicating C is invalid, where $1 \leq i \leq i'$.
8. Dec:the decryption algorithm intakes $sk_{id|i}$ and outputs either a message m or a symbol $\perp$ indicating C is invalid.
9. Revoke: the revocation algorithm intakes an identity to be revoked id, a revocation time period T_i, the current revocation list RL, and a state st, and outputs an updated RL.

Correctness: For any (mpk, msk) output by Setup, any time period $T_i \in \mathscr{T}$, any message $m \in \mathscr{M}$, and all possible states st and revocation list RL (where $i \in [1, poly(1^n)]$), if sk_{id} is output by $KeyGen(msk, id), \tau \leftarrow TokenUp(msk, id, T_i, RL, st), sk_{id|i} \leftarrow DeKeyGen(sk_{id}, \tau_i), rk_{id|i\to j} \leftarrow ReKeyGen(sk_{id|i}, msk, T_i, T_j)$ (note for simplicity we set $j = i+1$ here), we have

- if id is not revoked by T_1: $Dec(sk_{id|1}, Enc(id, T_1, m)) = m$.
- if id is not revoked by T_i:

$$Dec(sk_{id|i}, ReEnc(rk_{id|i-1\to i}, \cdots, ReEnc(rk_{id|1\to 2}, Enc(id, T_1, m)))\cdots) = m$$

3 Review of Construction

In our construction we let state st be an unspecified data structure DS, and it depends on which structure we use, e.g., st can be a binary tree. By a tuple (RL, st) we mean a revocation list and its corresponding data structure

1. Setup$(1^k, N)$. The setup algorithm runs $(q, g, p, \mathbb{G}, \mathbb{G}_T) \leftarrow \mathscr{G}(1^k)$, where q is the order of group $\mathbb{G}$, it chooses $\alpha, \beta \in_R \mathbb{Z}_q^*$, group elements $g_2, g_3, v_1, v_2 \in_R \mathbb{G}$, a random n-length set $U = \{u_j | 0 \le j \le n\}$ and a target collision resistant (TCR) hash function $TCR_1 : \mathbb{G} \to \mathbb{Z}_q^*$, where $u_j \in_R \mathbb{G}$. The public parameter is $mpk = (g, g_1, g_2, g_3, v_1, v_2, U, TCR_1)$, the master secret key is $msk = (g_2^{alpha}, g_3^{\beta})$, $RL = \emptyset$ and $st = DB$, where $g_1 = g^{\alpha}$.
2. KeyGen(msk, id). PKG chooses $r_{id} \in_R \mathbb{Z}_q^*$, sets the partial private key sk_{id} as $sk_{id_1} = g_3^{\beta}(u_o \prod_{j\in\mathscr{V}_{id}} u_j)^{r_{id}}$, $sk_{id_2} = g^{r_{id}}$, where V_{id} is the set of all j for which the j-th bit (of id) is equal to 1.
3. TokenUp(msk, id, T_i, RL, st). PKG will check RL first so as to see whether id is revoked or not. If it is revoked, output $\perp$; else proceed. Choose $r_{T_i} \in_R \mathbb{Z}_q^*$, and set the token τ_i as $\tau_{i,1} = (g_2^{\alpha}/g_3^{\beta}) \cdot (v_1 \cdot v_2^{T_i})^{r_{T_i}}$, $\tau_{i,2} = g^{r_{T_i}}$, where i is the index for the time period.
4. DeKeyGen(sk_{id}, τ_i). A user id runs the algorithm as follows.
   a. Choose $\tilde{r} \in_R \mathbb{Z}_q^*$, and randomize the token as $\tau_{i,1} = \tau_{i,1} \cdot (v_1 \cdot v_2^{T_i})^{\tilde{r}}$, $\tau_{i,2} = \tau_{i,2} \cdot g^{\tilde{r}}$.
   b. Choose $r_1, r_2 \in_R \mathbb{Z}_q^*$, and set the updated secret key $sk_{id|i}$ for identity id and time period T_i as

$$\begin{aligned} sk_{id|i,1} &= sk_{id_1} \cdot \tau_{i,1} \cdot (u_0 \prod_{j\in\mathscr{V}_{id}} u_j)^{r_1} \cdot (v_1 \cdot v_2^{T_i})^{r_2} \\ &= g_2^{\alpha}(u_0 \prod_{j\in\mathscr{V}_{id}} u_j)^{\hat{r_1}} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r_2}} \\ sk_{id|i,2} &= sk_{id_1} \cdot g^{r_1} = g^{\hat{r_1}}, sk_{id|i,3} = \tau_{i,2} \cdot g^{r_2} = g^{\hat{r_2}}, \end{aligned}$$

where $\hat{r_1} = r_{id} + r_1$, $\hat{r_2} = r_{T_i} + \tilde{r} + r_2$. Note the user will share $r_1, r_2, \tilde{r}$ with the *PKG* (suppose it is fully trusted) such that the *PKG* can store $(id|i, \hat{r_1}, \hat{r_2})$ in a list $List^{sk_{id|i}}$ for further use.

5. ReKeyGen($sk_{id|i}, msk, T_i, T_i'$). The re-encryption key $rk_{id|i\to i'}$ is generated as follows.
   a. ReKeyToken(msk, T_i, T_i'). If a user id holding $sk_{id|i}$ is allowed to update his key to another time period $T_{i'}$, *PKG* generates the re-encryption key token $\varphi_{i\to i'}$ as $\varphi_{i\to i'}^{(1)} = \frac{(v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)}}{(v_1 \cdot v_2^{T_i})^{\hat{r_2}}}$, $\varphi_{i\to i'}^{(2)} = (\hat{C_0}, \hat{C_1}, \hat{C_2}, \hat{C_3}) \leftarrow Enc(id, T_{i'}, \xi)$, where $\xi \in_R \mathbb{G}_T$, $\hat{r_2}$ is recovered from $(id|i', \hat{r_1}, \hat{r_2})$ which is stored the $List^{sk_{id|i}}$.
   b. ReKey($sk_{id|i}, \varphi_{i\to i'}$). After receiving $\varphi_{i\to i'}$ from PKG, the user id generates the re-encryption key as follows.
      i. Choose $\rho \in_R \mathbb{Z}_q^*$, and set $rk_1 = sk_{id|i,1} \cdot \varphi_{i\to i'}^{(1)} \cdot (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^{\rho}$, $rk_2 = sk_{id|i,1} \cdot g^{\rho}$, and $rk_3 = \varphi_{i\to i'}^{(2)}$.
      ii. Output the re-encryption key $rk_{id|i\to i'} = (rk_1, rk_2, rk_3)$.
6. Enc(id, T_i, m). Given an identity id, a time period T_i, and a message $m \in \mathbb{G}_T$, the encryption algorithm chooses $t \in_R \mathbb{Z}_q^*$, and sets the original ciphertext C as $C_0 = m \cdot e(g_1, g_2)^t, C_1 = g^t, C_2 = (u_0 \prod_{j\in\mathcal{V}_{id}} u_j)^t, C_3 = (v_1 \cdot v_2^{T_i})^t$. We assume that the identity id and the time period T_i are implicitly included in the ciphertext.
7. ReEnc($rk_{id|i\to i'}, C$). Parse the ciphertext C under (id, T_i) as (C_0, C_1, C_2, C_3), and the re-encryption key $rk_{id|i\to i'}$ as (rk_1, rk_2, rk_3). The re-encryption algorithm computes $C_4 = \frac{e(C_1, rk_1)}{e(C_2, rk_2)} = e(g^t, g^{\alpha} \cdot (v_1 \cdot v_2^{T_{i'}})^{TCR_1(\xi)})$, and next sets the re-encrypted ciphertext C under $(id, T_{i'})$ as (C_0, C_1, C_4, rk_3). Note if C under $(id, T_{i'})$ needs to be further re-encrypted to the time period $T_{i''}$, then the proxy parses rk_3 as $(\hat{C_0}, \hat{C_1}, \hat{C_2}, \hat{C_3})$. Given a re-encryption key $rk_{id|i'\to i''} = (rk_1', rk_2', rk_3')$, the proxy computes $C_4' = \frac{e(\hat{C_1}, rk_1')}{e(\hat{C_2}, rk_2')}$, and sets the ciphertext C under $(id, T_{i''})$ as $(C_0, C_1, C_4, \hat{C_0}, \hat{C_1}, \hat{C_4'}, rk_3')$.
8. Dec($sk_{id|i}, C$). Given a ciphertext C under (id, T_i), the decryption algorithm works as follows.
   a. For the original ciphertext $C = (C_0, C_1, C_2, C_3)$, the decryptor computes

$$\frac{e(C_1, sk_{id|i,1})}{e(C_2, sk_{id|i,2}) e(C_3, sk_{id|i,3})} = e(g_1, g_2)^t$$

   and outputs the message

$$\frac{C_0}{e(g_1, g_2)^t} = \frac{m \cdot e(g_1, g_2)^t}{e(g_1, g_2)^t} = m$$

   b. For the the re-encrypted ciphertext C:

i. If the re-encrypted ciphertext is re-encrypted only once, i.e. $C = (C_0, C_1, C_4, rk_3 = (\hat{C}_0, \hat{C}_1, \hat{C}_2, \hat{C}_3))$, then the decryptor computes

$$\frac{\hat{C}_0 e(\hat{C}_2, sk_{id|i,2}) e(\hat{C}_3, sk_{id|i,3})}{e(\hat{C}_1, sk_{id|i,1})} = \xi$$

Accordingly, the decryptor can finally compute

$$C_0 \cdot \frac{e(C_1, (v_1 v_2^{T_i})^{TCR_1(\xi)})}{C_4} = m$$

ii. If the ciphertext under id is re-encrypted l times from time period T_1 to T_{l+1}, we denote the re-encrypted ciphertext as $C^{(l+1)} = (C_0^{(1)}, C_1^{(1)}, C_4^{(1)}, \cdots, C_0^{(l)}, C_1^{(l)}, C_4^{(l)}, rk_3^{(l+1)})$, where $C_0^{(1)}$ and $C_1^{(1)}$ are the components of original ciphertext under (id, T_1), and $rk_3^{(i+1)} = (C_0^{(i+l)}, C_1^{(i+l)}, C_2^{(i+l)}, C_3^{(i+l)})$ is the ciphertext under $(id, T_{i+1})(i \in [1, l])$. We recover the message m as follows. First set

$$\frac{C_0^{(l+1)} e(C_2{}^{(l+1)}, sk_{id|l+1,2}) e(C_3{}^{(l+1)}, sk_{id|l+1,3})}{e(C_1{}^{(l+1)}, sk_{id|l+1,1})} = \xi^{(l)}$$

from $i = 1$ to 2 set

$$\frac{C_0^{(i)} e(C_1^{(i)}, (v_1 v_2^{T_{i+1}})^{TCR_1(\xi^{(i)})})}{C_4^{(i)}} = \xi^{(i-1)}$$

finally compute

$$\frac{C_0^{(1)} e(C_1^{(1)}, (v_1 v_2^{T_2})^{TCR_1(\xi^{(1)})})}{C_4^{(1)}} = m$$

9. Revoke(id, T_i, RL, st). Update the revocation list by $RL \leftarrow RL \cup \{id, T_i\}$ and return the updated revocation list.

4 Our Attack

Our attack is based on the following observation: Assume user A is revoked at time period T_i', and he maliciously at time period $T_i (i \leq i')$ colludes with the cloud which holds the re-encryption key, they can derive a powerful secret key which can decrypt the ciphertext for any time period T_j for $j \geq i'$. Considering this scenario: data owner distributes his data contents with users $A, B, C, \cdots, Z$, and these users have paid fee for this content sharing service. Note here user A can be revoked by the data owner

for his payment is not continued any more. A try to corrupt with the cloud by pay a much less fee to the cloud, but the cloud want to hide this corrupt for the data owner, thus it honestly do the re-encryption process but leak the re-encryption keys to user A. By using this re-encryption key, A can derive the powerful secret key which is enough to decrypt the future data contents of data owner, while data owner can not blame the cloud for leaking the re-encryption keys, for these re-encryption keys maybe leaked by other ways, such as leaking in the outsourcing process to the cloud. Concretely the attack can be described as following:

1. According to the DeKeyGen algorithm, User A at time period T_i is with the secret key

$$\begin{aligned} sk_{id|i,1} &= sk_{id_1} \cdot \tau_{i,1} \cdot (u_0 \prod_{j \in \mathscr{V}_{id}} u_j)^{r_1} \cdot (v_1 \cdot v_2^{T_i})^{r_2} \\ &= g_2^{\alpha} (u_0 \prod_{j \in \mathscr{V}_{id}} u_j)^{\hat{r_1}} \cdot (v_1 \cdot v_2^{T_i})^{\hat{r_2}} \\ sk_{id|i,2} &= sk_{id_1} \cdot g^{r_1} = g^{\hat{r_1}}, sk_{id|i,3} = \tau_{i,2} \cdot g^{r_2} = g^{\hat{r_2}}, \end{aligned}$$

2. The cloud holds the re-encryption key $rk_{id|i-1 \to i}$ where $i-1 \leq i$, which is

$$rk_1 = sk_{id-1|i-1,1} \cdot \varphi_{i-1 \to i}^{(1)} \cdot (u_0 \prod_{j \in \mathscr{V}_{id}} u_j)^{\rho}, rk_2 = sk_{id|i-1,1} \cdot g^{\rho}$$

$$rk_3 = \varphi_{i \to i'}^{(2)}$$

where

$$\varphi_{i-1 \to i}^{(1)} = \frac{(v_1 \cdot v_2^{T_i})^{TCR_1(\xi)}}{(v_1 \cdot v_2^{T_{i-1}})^{\hat{r_2}}}$$

$$\varphi_{i-1 \to i}^{(2)} = (\hat{C}_0, \hat{C}_1, \hat{C}_2, \hat{C}_3) \leftarrow Enc(id, T_i, \xi)$$

The cloud sends the re-encryption key to user A.

3. User A then can decrypt the $\varphi_{i-1 \to i}^{(2)}$ and get ξ, with this ξ, he can get $(v_1 \cdot v_2^{T_{i-1}})^{\hat{r_2}}$, and thus can get

$$\begin{aligned} Fsk_{id|i-1,1} &= g_2^{\alpha} (u_0 \prod_{j \in \mathscr{V}_{id}} u_j)^{\hat{r_1}+\rho} \\ Fsk_{id|i-1,2} &= g^{\hat{r_1}+\rho} \end{aligned}$$

4. It is easily to see $Fsk_{id|i-1,1}, Fsk_{id|i-1,2}$ are powerful secret key for id, which can decrypt any period ciphertext for id. For example, for ciphertext (C_0, C_1, C_4, rk_3), user A first decrypt rk_3 (with only (C_0, C_1, C_2) to get ξ, and then finally get

$$C_0 \cdot \frac{e(C_1, (v_1 v_2^{T_i})^{TCR_1(\xi)})}{C_4} = m$$

5. Thus user A can decrypt period ciphertext for id, whether he has revoked or not.

5 Conclusion

In this paper, we show an attack to a recently proposed cloud-based revocable identity based proxy re-encryption scheme by colluding the proxy with the revoked user. Our research results show that, although using cloud for revoking users with identity based proxy re-encryption is a great idea to reduce the complicated burden for private key generation centers if using traditional method, it needs carefully design and more consideration for being adapted widely.

Acknowledgements This work was supported by the National Natural Science Foundation of China (no. 61572390), the 111 Project (No. B08038), the second author is the corresponding author.

References

1. G. Ateniese, K. Fu, M. Green and S. Hohenberger. Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage. In *NDSS(2005), pages 29–43.*
2. G. Ateniese, K. Fu, M. Green and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *ACM Transation Information System Security 9 (2006), no. 1, pages 1–30.*
3. M. Blaze, G. Bleumer and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Advances in Cryptology - Eurocrypt'98,* LNCS 1403, pp. 127–144. Springer–Verlag, 1998.
4. R. Canetti and S. Hohenberger, Chosen Ciphertext Secure Proxy Re-encryption. In *ACM CCS 2007*, pp. 185–194.2007.
5. R. Deng, J. Weng, S. Liu and K. Chen. Chosen Ciphertext Secure Proxy Re-encryption without Pairing. In *CANS'08,* LNCS 5339, pp.1-17, Springer-Verlag, 2008.
6. M. Green and G. Ateniese. Identity-based proxy re-encryption. In *ACNS 2007*, volume 4521 of *LNCS*, pages 288–306, 2007.
7. B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In *PKC 2008,* LNCS 4939, pages 360–379, Springer–Verlag, 2008.
8. B. Libert and D. Vergnaud. Unidirectional Chosen-Ciphertext Secure Proxy Re-Encryption. In *IEEE Transactions on Information Theory*, vol. 57, No. 3, pages 1786–1802, 2011.
9. K. Liang, W. Susilo, J. K. Liu. Privacy-Preserving Ciphertext Multi-Sharing Control for Big Data Storage. In *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 8, pages 1578–1589, 2015.
10. K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, T. Phuong, Q. Xie. A DFA-Based Functional Proxy Re-Encryption Scheme for Secure Public Cloud Data Sharing. In *IEEE Transactions on Information Forensics and Security*, Vol. 9, No. 10, pages 1667–1680, 2014.
11. K. Liang, W. Susilo. Searchable Attribute-Based Mechanism With Efficient Data Sharing for Secure Cloud Storage. In *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 9, pages 1981–1992, 2015.
12. J. Shao and Z. Cao. CCA-secure proxy re-encryption without pairing. In *PKC 2009,* LNCS 5443, pages. 357–376, Springer–Verlag, 2009.
13. J. Shao, Z. Cao, P. Lin. Generic construction for CCA-secure unidirectional proxy re-encryption. In Security and Communication Networks, no. 2, pages 1-16, 2009.
14. J. Weng, R. H. Deng, C. Chu, X. Ding, and J. Lai. Conditional proxy re-encryption secure against chosen-ciphertext attack. In *ACM ASIACCS 2009*, pages 322–332, 2009.

15. J. Weng, Y. Yang, Q. Tang, R. Deng, and F. Bao. Efficient conditional proxy re-encryption with chosen-ciphertext security. In *ISC 2009*, volume 5735 of *LNCS*, pages 151–166, 2009.
16. J. Weng, Y. Zhao, G. Hanaoka. On the Security of a Bidirectional Proxy Re-encryption Scheme from PKC 2010. In *PKC 2011*, pages 284-295, 2011.
17. J. Weng, M. Chen, Y. Yang, R. Deng, K. Chen and F. Bao CCA-secure unidirectional proxy re-encryption in the adaptive corruption model without random oracles. In *Science China Information Sciences*, 53, : 593-606, 2010.
18. S. Chow, J. Weng, Y. Yang, R. Deng Efficient unidirectional proxy re-encryption. In *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 316–332, 2010.