Strongly Secure and Cost-Effective Certificateless Proxy Re-encryption Scheme for Data Sharing in Cloud Computing

Zhiguang Qin(✉), Shikun Wu, and Hu Xiong

School of Information and Software Engineering,
University of Electronic Science and Technology of China,
Chengdu 610054, Sichuan, China
{qinzg,shkwu,xionghu}@uestc.edu.cn

Abstract. Proxy re-encryption (PRE) has been considered as a promising candidate to secure data sharing in public cloud by enabling the cloud to transform the ciphertext to legitimate recipients on behalf of the data owner, and preserving data privacy from semi-trusted cloud. Certificateless proxy re-encryption (CL-PRE) not only eliminates the heavy public key certificate management in traditional public key infrastructure, but also solves the key escrow problem in the ID-based public key cryptography. By considering that the existing CL-PRE schemes either rely on expensive bilinear pairings or are proven secure under weak security models, we propose a strongly secure CL-PRE scheme without resorting to the bilinear pairing. The security of our scheme is proven to be secure against adaptive chosen ciphertext attack (IND-CCA) under a stronger security model in which the Type I adversary is allowed to replace the public key associated with the challenge identity. Furthermore, the simulation results demonstrate that our scheme is practical for cloud based data sharing in terms of communication overhead and computation cost for data owner, the cloud and data recipient.

Keywords: Certificateless proxy re-encryption · Data sharing · Cloud computing · Without pairings

1 Introduction

With the rapid development of cloud computing, the security of outsourced data in cloud has attracted a lot of concern from industry and academe recently. In case the data is outsourced to a semi-trust cloud service provider (CSP), the data owner cannot take control of their own data directly. To avoid disclosing the outsourced data to CSP or other unauthorized users, it is essential for data owners to preserve the privacy of the outsourced data in the cloud [1]. Intuitively, traditional asymmetric encryption scheme can be adopted to enforce the access control of data outsourced in the cloud. It seems to be feasible for data owners to outsource encrypted data to the semi-trusted cloud if only the data owner

Y. Wang et al. (Eds.): BigCom 2015, LNCS 9196, pp. 205–216, 2015.
DOI: 10.1007/978-3-319-22047-5_17

himself/herself can access the encrypted data. However, it becomes cumbersome to share encrypted data between different users based on traditional encryption mechanisms. To share the encrypted data with other users, a data owner needs to download and decrypt the requested data, and further re-encrypt it using a target user's public key to accomplish data sharing. Another naive approach for data owner is to share his/her private key with the target user who is authorized to decrypt the outsourced data directly. Obviously, the former method renders heavy communication overhead and computation cost, and thus mismatches the purpose of cloud computing. The idea of disclosing private keys to authorized users in the latter method violates the least privilege principle. Thus, it is challenging to share encrypted data in the cloud computing environment.

Proxy re-encryption (PRE) [9], which enables a semi-trusted proxy to transform a ciphertext which has been encrypted under one public key into a ciphertext under another public key of the same message without leaking any information to the proxy, is considered to be a promising candidate to achieve secure data sharing in cloud computing. Consider the following scenario: the data owner, say Alice, wants to share the sensitive data outsourced in the cloud with a third party, Bob. It is natural for Alice to desire that the requested data can only be accessed by Bob. Inspired by the primitive of PRE, Alice can encrypt the sensitive data before outsourcing these data to the semi-trusted cloud. After receiving the request of decryption delegation from Bob, Alice generates a proxy re-encryption key using his/her own private key and Bob's public key, and sends this proxy re-encryption key to the semi-trusted cloud. Equipped with this proxy re-encryption key, CSP can transform the ciphertext encrypted under the public key of Alice into the ciphertext under the public key of Bob. Meanwhile, the outsourced data can only be accessed by Bob since the CSP cannot decrypt the encrypted data with the proxy re-encryption key. Finally, Bob can download and decrypt the outsourced data with his/her own private key.

Before the PRE can be widely deployed in the cloud environment, several issues should be addressed. Observing the heavy management of public key certificates in traditional public key encryption (PKE) [9], [10], [11] and the key escrow problem in the ID-based public key encryption (ID-PKE) [13], [5], it is natural to investigate PRE in the certificateless public key encryption (CL-PKE) setting [2]. To enjoy the merits of traditional PKE and ID-PKE without suffering the corresponding criticisms, Sur *et al.* [4] introduced the primitive of PRE into CL-PKE [5] and proposed the first concrete certificateless proxy re-encryption (CL-PRE) scheme. Concretely, CL-PRE leverages the identity of a user as an ingredient of its public key, while eliminates the key escrow problem in ID-PKE, and does not require the use of certificates to guarantee the authenticity of public keys in traditional PKE.

Since Sur *et al.* [4] introduced the CL-PRE, this cryptosystem has attracted more attention. In 2012, Xu *et al* [6] constructed a CL-PRE scheme, which is claimed to be chosen-plaintext attack (CPA) secure and is introduced to cloud based data sharing scenario. In 2013, Yang *et al.* [7] constructed the first CCA secure pairing-free CL-PRE scheme based on Baek *et al*'s [8] CL-PKE scheme.

In 2014, Wang *et al* [12] also proposed a CCA secure CL-PRE scheme without bilinear pairings under Yang *et al.*'s security models [7]. Compared to the previous CL-PRE schemes, an attractive feature in Yang *et al.*'s scheme is the efficiency gained from removing computationally-heavy pairing operations. However, the untransformed ciphertexts in their scheme may greatly consume computation and storage resources for data owners. In addition, we point out that the security models for CL-PRE in [7] are sightly weak in a sense that the Type I adversary is not allowed to replace the public key associated with the challenge identity.

Our Contributions. We define an architecture of cloud based data sharing using the primitive of CL-PRE. To this end, we further propose a strongly secure and pairing-free CL-PRE scheme based on Yang *et al.*'s [7] scheme. That is, our CL-PRE scheme is cost-effective and provable secure against the Type I and Type II adversaries in a strong sense that the Type I adversary is able to replace the public key associated with the challenge identity (before challenge phase). Moreover, we evaluate communication overhead and computation cost in terms of data owner, the CSP and data recipient. Our results demonstrate that our CL-PRE scheme outperforms other existing works at the requirements of large scale data sharing.

The rest of this paper is organized as follows. Section 2 gives definitions of CL-PRE for cloud based data sharing. Our CL-PRE scheme is given in Section 3 followed by the security and performance analysis in Section 4. Section 5 concludes this paper.

2 Preliminaries and Definitions

In this section, we first describe a system architecture and give the definition of CL-PRE for cloud based data sharing. Then, we define the security assumptions and security model for a secure data sharing with public cloud.

2.1 Definition of a System Architecture

We consider an architecture of cloud based data sharing by introducing the primitive of CL-PRE as shown in Fig. 1, which involves a data owner, the cloud service provider (CSP) and data recipients. A data owner creates the encrypted data and host her data to the semi-trusted CSP. The CSP stores the data owner's data and provides the data access to the data owner and the authorized data recipients. The data owner is able to share her encrypted data to data recipients, who should first request for decryption delegations. After receiving the requests, the data owner produces a proxy re-encryption key for each data recipient and sends them to the CSP through a secure channel. Utilizing these proxy re-encryption keys, the CSP can transform the data owner's encrypted data into each data recipient's without disclosing any information. Then, data recipients can download and decrypt the data owner's outsourced data by themselves. As mentioned above, we assume the CSP itself is semi-trusted, which means it

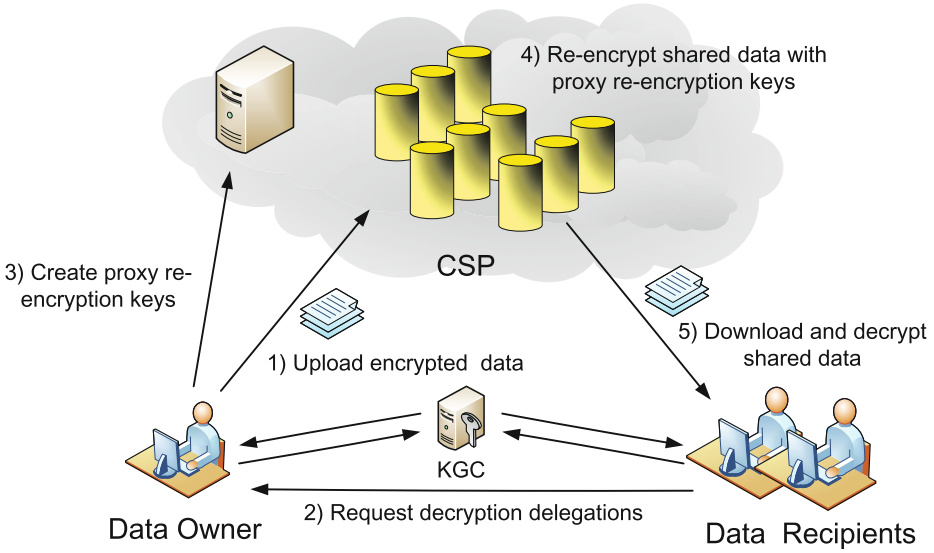

Fig. 1. The Architecture of CL-PRE for Cloud based Data Sharing

follows protocols and does not pollute data confidentiality actively as a malicious adversary, but it may be curious about the received data and may collude with data recipients to launch attacks on data owner.

Furthermore, a proxy re-encryption key is produced by inputting a data owner's private key and a data recipient's public key. Since the number of cloud users participating in data sharing may be large, traditional PKE based approach has the public key management issue, and ID-PKE based approach has the private key escrow problem. Uniquely, we adopt CL-PRE for data sharing in the cloud.

Definition 1 (CL-PRE for Data Sharing). *A cloud based data sharing mechanism designed by the primitive of certificateless proxy re-encryption (CL-PRE) consists of the following nine algorithms:*

- **Setup:** Taking a security parameter k as input, this algorithm is run by the key generation center (KGC) to produce a master key mk and a list of public parameters *params*.
- **PartialKeyExtract:** Taking a list of public parameters *params*, a master key mk and a cloud user's identifier ID_i as input, this algorithm is performed by KGC to return a partial public/private key pair (P_i, D_i) to the user with an identifier ID_i.
- **SetSecretValue:** Taking a list of public parameters *params* and an identifier ID_i as input, this algorithm is executed by the user ID_i to set a secret value z_i.

- **SetPrivateKey:** Taking a list of public parameters *params*, a partial private key D_i and a secret value z_i as input, this algorithm is carried out by user ID_i to set a private key sk_i.
- **SetPublicKey:** Taking a list of public parameters *params*, a partial public key P_i and a secret value z_i as input, this algorithm is carried out by user ID_i to set a public key pk_i.
- **Encrypt:** Taking a list of public parameters *params*, a plaintext data m and a public key pk_A as input, this algorithm is performed by data owner Alice to produce a ciphertext C_A. Then Alice uploads her ciphertext to CSP.
- **ReEncryptKey:** Taking a list of public parameters *params*, Alice's public/private key pair (pk_A, sk_A) associated with an identifier ID_A and data recipient Bob's public key pk_B associated with an identifier ID_B as input, this algorithm is implemented by Alice to generate a proxy re-encryption key $rk_{A\to B}$, which is further sent to CSP through a secure channel.
- **ReEncrypt:** Taking a list of public parameters *params*, a ciphertext C_A for Alice and a proxy re-encryption key $rk_{A\to B}$ as input, this algorithm is executed by the CSP to produce a transformed ciphertext C_B for Bob.
- **Decrypt:** Taking a list of public parameters *params*, a private key sk_i and a ciphertext C_i, this algorithm is run by the data owner/recipient associated with an identifier ID_i to obtain the underlying encrypted data m or return a distinguished symbol $\perp$.

The above CL-PRE scheme allows the CSP using a proxy re-encryption key $rk_{A\to B}$ to transform a ciphertext encrypted under a data owner Alice's public key pk_A into another ciphertext encrypted under a data recipient Bob's public key pk_B on the same message m without leaking m or sk_A/sk_B of data owner/data recipient to the CSP.

Correctness: For all data $m \in \mathcal{M}$, key pair (pk_A, sk_A) for Alice and (pk_B, sk_B) for Bob, Alice is able to correctly share her underlying encrypted data m with Bob in the cloud if and only if the following conditions are satisfied:

- $\mathbf{Decrypt}_1(params, sk_A, \mathbf{Encrypt}(params, ID_A, pk_A, m)) = m$.
- $\mathbf{Decrypt}_2(params, sk_B, \mathbf{ReEncrypt}(params, rk_{A\to B}, C_A)) = m$.

2.2 Assumption

Definition 2 (Computational Diffie-Hellman (CDH)). *Assume that g is a generator chosen randomly from group $\mathbb{G}$ with the primer order q. Let $\mathcal{A}$ be an adversary. $\mathcal{A}$ tries to solve the problem as follows: Given $(g, g^a, g^b) \in \mathbb{G}$, where $a, b \in \mathbb{Z}_q^*$, compute $g^{a\cdot b}$. We define $\mathbb{G}$ satisfies CDH problem assumption if there is no probability polynomial time algorithm to solve the CDH problem with advantage $Adv(\mathcal{A}) = Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}]$.*

2.3 Security Model

First we recall the security models of CL-PRE, given by Yang *et al.* [7] based on Baek *et al.*'s [8] security models of CL-PKE. Their definition considers two

types of adversaries, Type I and Type II. The difference between them is that a Type I adversary $\mathcal{A}_I$ does not have access to the master key but may replace public keys of arbitrary identities with values of its own choice, whereas a Type II adversary $\mathcal{A}_{II}$ does have access to the master key but may not replace public keys of entities. It must be pointed that Yang *et al.*'s security models for a Type I adversary is sightly weak in a sense that $\mathcal{A}_I$ is not allowed to replace the public key of the challenge identity ID^* in any phase in order to prove the second level ciphertext security of their CL-PRE scheme. However, Sun *et al.* [3] eliminated Baek *et al.*'s limitation and presented an improving CL-PKE scheme with strong Type I. As a result, the security models of CL-PRE we formalize by taking account of strongly CL-PKE security notions [2], [3] and PRE security notions [7], [9], [10] are the strong Type I and Type II, where $\mathcal{A}_I$ is able to replace the public key associated with the challenge identity. Furthermore, our CL-PRE scheme does not depend on bilinear pairings. Our strong security model is enough for many practical applications such as secure data sharing with public cloud [6]. The details of our strong security model can be found in the full paper.

3 Our Pairing-Free CL-PRE Scheme

In this section, we modify Yang *et al.*'s CL-PRE scheme [7] to construct a strongly secure one without pairings based on Sun *et al.*'s CL-PKE scheme [3].

The detailed description of our CL-PRE scheme is as follows:

- **Setup:** Taking a security parameter k as input, this algorithm produces a prime q and a group $\mathbb{G}$ of order q. Then it performs as follows:
  1. Choose a random generator $g \in \mathbb{G}$.
  2. Pick $s \in \mathbb{Z}_q^*$ at random and compute $h = g^s$.
  3. Select hash functions $H_1 : \{0,1\}^* \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_4 : \mathbb{G} \to \{0,1\}^{n+k_0}$, and $H_5 : \mathbb{G} \to \mathbb{Z}_q^*$.

  The system public parameters are $params = \langle q, n, k_0, g, h, H_1, H_2, H_3, H_4, H_5 \rangle$, where n, k_0 mean the bit-length of a plaintext and a random bit string, respectively. The system master key $mk = s$. Note that plaintext space is $\mathcal{M} = \{0,1\}^n$ and ciphertext space is $\mathcal{C} = \{0,1\}^{n+k_0}$.
- **PartialKeyExtract:** Taking *params*, mk and an identifier ID_A for Alice as input, this algorithm picks $\alpha_1, \alpha_2 \in \mathbb{Z}_q^*$ at random and computes $a_1 = g^{\alpha_1}$, $a_2 = g^{\alpha_2}$, $x_1 = \alpha_1 + sH_1(ID_A, a_1)$ and $x_2 = \alpha_2 + sH_1(ID_A, a_1, a_2)$. Then it returns a partial private key $D_A = x_1$ and a partial public key $P_A = (a_1, a_2, x_2)$ for Alice.
- **SetSecretValue:** Taking *params* and ID_A as input, this algorithm randomly picks $z_A \in \mathbb{Z}_q^*$ as a secret value for Alice.
- **SetPrivateKey:** Taking *params*, Alice's partial private key D_A and z_A as input, this algorithm returns a private key $sk_A = (x_1, z_A)$ for Alice.
- **SetPublicKey:** Taking *params*, Alice's partial public key P_A and secret value z_A as input, this algorithm computes $u_A = g^{z_A}$ and returns a public key $pk_A = (u_A, a_1, a_2, x_2)$ for that user.

- **ReEncryptKey:** Taking *params*, an identifier ID_A and a public/private key pair (pk_A, sk_A) for Alice, an identifier ID_B and a public key pk_B for Bob as input, this algorithm computes $t_B = b_1 h^{H_1(ID_B, b_1)}$ and $t_{AB} = H_3(t_B^{z_A} \parallel u_B^{x_1} \parallel ID_A \parallel pk_A \parallel ID_B \parallel pk_B)$. Then it returns a re-encryption key $rk_{A\to B} = (x_1 H_5(u_A) + z_A) t_{AB}$.
- **Encrypt:** Taking *params*, a plaintext message $m \in \mathcal{M}$ and Alice's public key pk_A as input, this algorithm performs as follows:
  1. Check $g^{x_2} \stackrel{?}{=} a_2 h^{H_2(ID_A, a_1, a_2)}$.
  2. Select $\sigma \in \{0,1\}^{k_0}$ at random and compute $t_A = a_1 h^{H_1(ID_A, a_1)}$ and $r = H_3(m \parallel \sigma \parallel ID_A \parallel u_A)$.
  3. Compute $c_1 = g^r$ and $c_2 = (m \parallel \sigma) \oplus H_4((t_A^{H_5(u_A)} \cdot u_A)^r)$.

  Then it returns a ciphertext $C_A = (c_1, c_2)$ for Alice.
- **ReEncrypt:** Taking *params*, Alice's ciphertext C_A and a re-encryption key $rk_{A\to B}$ as input, this algorithm computes $c_1' = c_1^{rk_{A\to B}}$ and $c_2' = c_2$. Then it outputs re-encrypted ciphertext $C_B = (c_1', c_2')$ for Bob or a distinguished symbol $\perp$.
- **Decrypt:** Taking *params*, a private key sk_{ID} and a ciphertext C_{ID} for user ID, this algorithm performs as follows:
  - **Decrypt$_1$:** To decrypt non re-encrypted ciphertext $C_A = (c_1, c_2)$ with $sk_A = (x_1, z_A)$, this algorithm computes $(m \parallel \sigma) = c_2 \oplus H_4(c_1^{(x_1 H_5(u_A) + z_A)})$. Then return plaintext m, if $r' = H_3(m \parallel \sigma \parallel ID_A \parallel u_A)$ and $g^{r'} = c_1$ holds. Otherwise, output $\perp$.
  - **Decrypt$_2$:** To decrypt re-encrypted ciphertext $C_B = (c_1', c_2')$ with $sk_B = (y_1, z_B)$, this algorithm computes as follows:
    1. Compute $t_A = a_1 h^{H_1(ID_A, a_1)}$ and $t_{BA} = H_3(u_A^{y_1} \parallel t_A^{z_B} \parallel ID_A \parallel pk_A \parallel ID_B \parallel pk_B)$.
    2. Compute $(m \parallel \sigma) = c_2' \oplus H_4((c_1')^{1/t_{BA}})$.
    3. If $r' = H_3(m \parallel \sigma \parallel ID_A \parallel u_A)$ and $(t_A^{H_5(u_A)\cdot u_A})^{r' t_{BA}} = c_1'$ holds, return m. Otherwise, output $\perp$.

It is easy to check the correctness of the pairing-free CL-PRE scheme above, we omit it here.

Remark 1. In our CL-PRE scheme, a existentially unforgeable under an adaptive chosen message attack (EUF-CMA) secure Schnorr signature is used to protect the partial public key from being replaced by attackers with the values of their choices. And $H_5(u_A)$ is necessary for resisting public key replacement attacks, where H_5 is collision free hash function.

Remark 2. The proposed scheme possesses properties such as *unidirectionality*, *single-hop*, *non-interactivity*, *non-transitivity*, *collusion-resistance*, both of which are suitable for security requirements in cloud based data sharing scenarios.

4 Analysis

4.1 Security Analysis

We have the following theorems about the security of the CL-PRE scheme. And due to the space limit, the proof of these theorems will be given in the full paper.

Theorem 1. *Our CL-PRE scheme is adaptive chosen ciphertext (IND-CCA) secure against the Type I adversary in the random oracle model, if for any polynomial time adversary $\mathcal{A}_I$ the CDH problem is intractable in $\mathbb{G}$.*
Theorem 2. *Our CL-PRE scheme is adaptive chosen ciphertext (IND-CCA) secure against the Type II adversary in the random oracle model, if for any polynomial time adversary $\mathcal{A}_{II}$ the CDH problem is intractable in $\mathbb{G}$.*

4.2 Performance Evaluation

Data sharing is a very resource demanding service with public cloud in terms of computation cost, communication overhead and storage space. In this subsection, we compare the performance of data owner, CSP and data recipient of our CL-PRE scheme with Xu *et al.*'s [6] scheme, Sur *et al.*'s [4] scheme and Yang *et al.*'s [7] scheme. For both Xu *et al.*'s [6] and Sur *et al.*'s pairing-based schemes, to satisfy 1024-bit RSA level security, we adopt the Tate pairing implemented over an elliptic curve defined on 512 bits prime field with a generator of order 160 bits. For Yang *et al.*'s [7] and our pairing-free schemes, to achieve the same security level, we implement them over 1024-bit prime finite field with a generator of order 160 bits. Additionally, we assume that the bit-length of $|m|$ and $|\sigma|$ is 1024 bits and 160 bits, respectively. Note that we obtain the running time for cryptographic operations using Miracal Library [14], a standard cryptographic library, on a PIV 3 GHZ processor with 512-MB memory and a Windows XP operation system. The running times of one pairing operation, one exponentiation operation and one map-to-point hash are 20.04ms, 5.83ms and 3.04ms, respectively. According to the description of multiple exponentiation algorithm in [7], we evaluate concrete running time and communication cost in Table 1 to make the comparison more clear between our scheme and three existing works [6], [4], [7].

From Table 1, our scheme is more efficient than Xu *et al.*'s scheme and Sur *et al.*'s scheme in terms of running time and ciphertext length. Compared with Yang *et al.*'s scheme, the overall performance of our scheme is more superior.

Computation Cost. In cloud based data sharing of CL-PRE, we assume that a data owner only has to do one encryption and decryption. From the observation of Table 1, the sum of running times on $\mathsf{Encrypt}$ and $\mathsf{Decrypt}_1$ in our scheme is the shortest. We believe this cost is not significant. However, the data owner should perform $\mathsf{ReEncryptKey}$ algorithm to generate a decryption delegation for each data recipient. As shown in Fig. 2, the computation cost of the data owner does increase linearly with the number of recipients. Our scheme and Yang *et al.*'s [7] scheme have the same overhead in $\mathsf{ReEncryptKey}$, so the curves coincide. For a data owner, both our scheme and Yang *et al.*'s scheme don't require high computational cost.

Next we analyze the computation cost of the CSP in different CL-PRE schemes. As a proxy, the CSP only has to carry out ciphertext transformations for numbers of data recipients. Thus, the $\mathsf{ReEncrypt}$ algorithm determines the

Table 1. The Performance Comparison of CL-PRE schemes

Schemes	Xu *et al.* [6]	Sur *et al.* [4]	Yang *et al.* [7]	Our CL-PRE
Encrypt	37.53 ms	32.66 ms	18.95 ms	23.32 ms
ReEncryptKey	40.57 ms	23.79 ms	12.65 ms	12.65 ms
ReEncrypt	20.04 ms	123.28 ms	12.65 ms	5.83 ms
$\text{Decrypt}_1(C_A)$	20.04 ms	58.04 ms	18.48 ms	11.66 ms
$\text{Decrypt}_2(C_B)$	43.12 ms	43.36 ms	24.78 ms	24.78 ms
$\|C_A\|$	3072 bits	4256 bits	3392 bits	2208 bits
$\|rk_{A\to B}\|$	3072 bits	3072 bits	160 bits	160 bits
$\|C_B\|$	4096 bits	4256 bits	2208 bits	2208 bits
Pairing-Free	×	×	√	√
Security Model	weak	weak	weak	strong
Assumption	DBDH	p-BDHI	CDH	CDH
Security	CPA	CCA	CCA	CCA

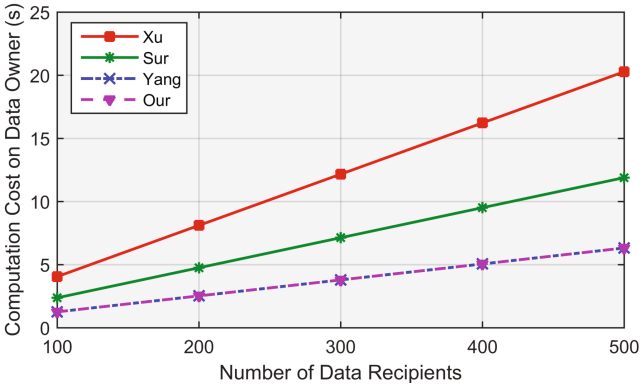

Fig. 2. The Computation Cost of the Data Owner

computation cost of the CSP. Table 1 shows that the running time on ReEncrypt in our scheme is superior to other schemes. The computation cost of the CSP increases accompanied by the number of data recipients. Clearly, Fig. 3 reflects the advantage.

For a data recipient, there is no difference that it obtains re-encrypted messages from the CSP and decrypts in the same way. The data recipient has to do one decryption only. The running time on Decrypt_2 in our scheme is equal to Yang *et al.*'s [7] scheme. As a result, the computation cost of our CL-PRE is inexpensive for cloud based data sharing.

Communication Overhead and Storage Overhead. For a data owner, the communication overhead is not constant and is relevant to the number of data recipients, since the data owner has to transport one encrypted message and

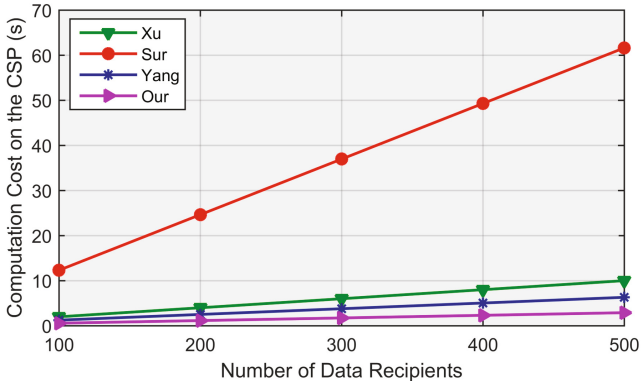

Fig. 3. The Computation Cost of the CSP

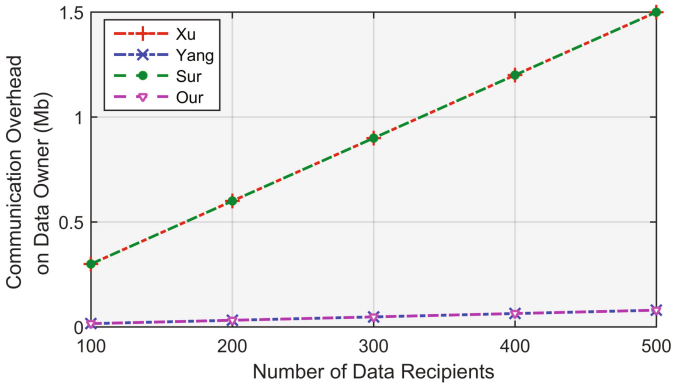

Fig. 4. The Communication Overhead of the Data Owner

one proxy re-encryption key for each recipient to the CSP. Fig. 4 shows the comparison of communication overhead for a data owner. Compared with Xu *et al.*'s [6] scheme and Sur *et al.*'s [4] scheme, our scheme has great advantage. In addition, the data owner not only has to keep its own public/private key pair but also has to store all the generated proxy re-encryption keys. Therefore, Fig. 4 also reflects the storage overhead of the data owner.

For one data sharing operation, the CSP transports one re-encrypted message for one recipient, i.e., the communication overhead is proportional to the number of recipients. In addition, the CSP should store the data owner's encrypted message, proxy re-encryption keys and re-encrypted messages for numbers of data recipients. Therefore, the storage overhead of the CSP is linear. We evaluate the storage overhead of the CSP with different CL-PRE schemes in Fig. 5.

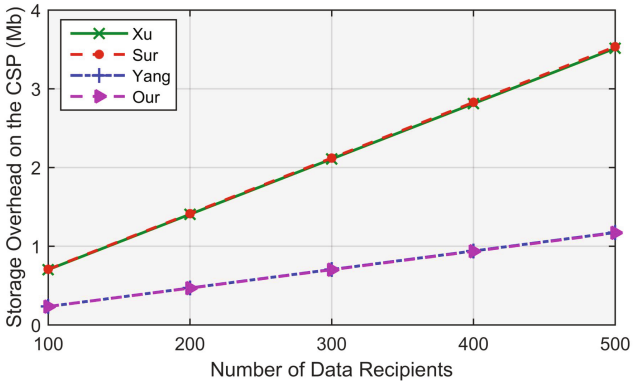

Fig. 5. The Storage Overhead of the CSP

In cloud based data sharing of CL-PRE, a data recipient has to obtain one re-encrypted message from the CSP only, so the communication overhead and storage overhead of the data recipient are not significant.

5 Conclusions

In this paper, we have proposed a strongly secure and efficient certificateless proxy re-encryption scheme (CL-PRE) without pairings for cloud based data sharing scenario. The proposed scheme is provably IND-CCA secure in a stronger security model, where a Type I adversary is allowed to replace the public key associated with the challenge identity (before challenge phase). The simulation results demonstrate that our scheme is strongly secure and practical for cloud based data sharing in terms of computation cost and communication overhead for data owner, CSP and data recipient.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China under Grant 61003230, Grant 61370026, and Grant 61202445, in part by the Fundamental Research Funds for the Central Universities under Grant ZYGX2013J073, and in part by the Applied Basic Research Program of Sichuan Province under Grant 2014JY0041.

References

1. Subashini, S., Kavitha, V.: A survey on security issues in service delivery models of cloud computing. Journal of network and computer applications. **34**(1), 1–11 (2011)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless public key cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)

3. Sun, Y., Zhang, F.T., Baek, J.: Strongly secure certificateless public key encryption without pairing. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 194–208. Springer, Heidelberg (2007)
4. Sur, C., Jung, C.D., Park, Y., Rhee, K.H.: Chosen-ciphertext secure certificateless proxy re-encryption. In: De Decker, B., Schaumüller-Bichl, I. (eds.) CMS 2010. LNCS, vol. 6109, pp. 214–232. Springer, Heidelberg (2010)
5. Libert, B., Quisquater, J.-J.: On constructing certificateless cryptosystems from identity based encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 474–490. Springer, Heidelberg (2006)
6. Xu, L., Wu, X., Zhang, X.: CL-PRE: a certificateless proxy re-encryption scheme for secure data sharing with public cloud. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2012), pp. 87–88. ACM, Seoul (2012)
7. Yang, K., Xu, J., Zhang, Z.: Certificateless proxy re-encryption without pairings-date. In: Lee, H.-S., Han, D.-G. (eds.) ICISC 2013. LNCS, vol. 8565, pp. 67–88. Springer, Heidelberg (2014)
8. Baek, J., Safavi-Naini, R., Susilo, W.: Certificateless public key encryption without pairing. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 134–148. Springer, Heidelberg (2005)
9. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of the 14h ACM conference on Computer and Communications Security (CCS 2007), pp. 185–194. ACM, Alexandria (2007)
10. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
11. Chow, S.S.M., Weng, J., Yang, Y., Deng, R.H.: Efficient unidirectional proxy re-encryption. In: Bernstein, D.J., Lange, T. (eds.) AFRICACRYPT 2010. LNCS, vol. 6055, pp. 316–332. Springer, Heidelberg (2010)
12. Wang, L., Chen, K., Mao, X., Wang, Y.: Efficient and provably-secure certificateless proxy re-encryption scheme for secure cloud data sharing. Journal of Shanghai Jiaotong University (Science) **19**, 398–405 (2014)
13. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
14. Shamus Software Ltd., Miracl library. http://www.shamus.ie/index.php?page=home