CL-PRE: a Certificateless Proxy Re-Encryption Scheme for Secure Data Sharing with Public Cloud

Lei Xu
Huawei Security & Privacy Lab
Beijing, China
stone.xu@huawei.com

Xiaoxin Wu
Huawei Security & Privacy Lab
Beijing, China
wuxiaoxin@huawei.com

Xinwen Zhang
Huawei Research Center
Santa Clara, CA, USA
xinwen.zhang@huawei.com

ABSTRACT

We propose CL-PRE, a *certificateless proxy re-encryption* scheme for secure data sharing with public cloud. In CL-PRE, a data owner encrypts shared data in cloud with an encryption key, which is further encrypted and transformed by cloud, and then distributed to legitimate recipients in accordance with access control. Uniquely, the cloud-based transformation leverages re-encryption keys derived from private key of data owner and public keys of receipts, and eliminates the key escrow problem in identity based cryptography and the need of certificate. While preserving data and key privacy from semi-trusted cloud, CL-PRE leverages maximal cloud resources to reduce the computing and communication cost for data owner. Towards running proxy in public cloud environment, we further propose *multi-proxy CL-PRE* and *randomized CL-PRE*, which enhance the security and robustness of CL-PRE. We implement all CL-PRE schemes and evaluate their security and performance.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems; D.4.6 [**Security and Protection**]: Cryptographic controls

General Terms

Security, Algorithms

Keywords

Cloud computing, cloud storage, certificateless public key cryptography, proxy re-encryption, access control

1. INTRODUCTION

Security has been considered as one of the critical concerns that hinder public cloud to be widely used. With the separation of data ownership and storage, a data owner has strong motivation to preserve its control of access and usage of shared data while leverage storage, computation, and distribution functions provided by cloud, and desire that a public cloud should not learn any clear data. It has been widely recognized that data confidentiality should be mainly relied on cloud customers instead of cloud service providers [1, 31].

A typical approach for data confidentiality protection is to encrypt a data with a (usually symmetric) key before storing it to cloud. However, encryption makes it difficult for flexibly sharing data between different users. On one side, sharing the data encryption key to all users easily enables a user to access all data that stored in cloud of a data owner, which violates the least privilege principle. On the other side, any change of access control policy either demands decryption and re-encryption of the data in cloud, which exposes clear data in cloud, or the data owner has to re-encrypt the data and re-upload to cloud, which brings computation and bandwidth cost to the owner. Furthermore, collusion between a legitimate user and the cloud easily allows unauthorized data sharing and distribution.

Key management is another burdensome task for encryption based access control. A data owner may manage keys by itself, e.g., through sending the data encryption key to individual recipients with the assumption that each user is equipped with a private/public key pair and the data encryption key is encrypted with a recipient's public key. This introduces heavy computing load at data owner side, and relies on trust and key management systems such as PKI which may not be scale and flexible well. Broadcast encryption and group key management can be used for sharing data in group manner. However, managing group is complicated in particular for today's pervasive data sharing such as cloud-based collaborations and social networks, where the number of groups of interests for an individual user is large. In addition, the size of a group can also be large, and the membership usually changes frequently, which makes group key management very tedious for a common user.

Since cloud is a resource pool for computation, storage, and networking, and provides elastic and pay-as-you-go resource consumption model, our motivation is to leverage cloud for encryption-based access control and key management. Towards end-to-end data confidentiality, we consider the cloud is *semi-trusted*, that is, clear data and encryption keys should never be exposed in cloud. Such a cloud based approach should be able to deliver data encryption keys with respect to pre-defined access control policies from the data owner, and introduce minor overhead on cloud users by eliminating any direct interaction between a data owner and its recipients.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

To achieve these goals, we develop CL-PRE, a new *proxy-based re-encryption* scheme augmented with *certificateless public key cryptography*, which leverages cloud not only for data storage but also for secure key distribution for data sharing. With CL-PRE, a data is first encrypted with a symmetric data encryption key (DEK) before stored in cloud by its owner. The data owner then generates *proxy re-encryption keys* with all of its potential recipients and sends to a cloud resident proxy service, along with the encrypted DEK with its public key. Using the re-encryption keys, the cloud is then able to transform the encrypted DEK to one that can be decrypted using an individual recipient' private key. In this way, the cloud works only as a proxy for key management. CL-PRE ensures that the cloud cannot get the clear DEK during the transformation.

An important novelty of CL-PRE is using certificateless public key cryptography [2] for proxy re-encryption, which is the first attempt to our best knowledge. Uniquely, CL-PRE leverages the identity of a recipient as an ingredient of its public key, while eliminates the key escrow problem in traditional identity-based encryption (IBE) [8], and does not require the use of certificates to guarantee the authenticity of public keys. Furthermore, a recipient generates its public key without interacting with PKG. In the process of private key generation, the recipient only needs to interact with PKG for one time. In order to update the public/private key pair later, it does not need to contact PKG. So the burden of PKG is reduced greatly.

The novel properties of CL-PRE satisfy security requirements for many cloud-based information sharing applications such as social network service (SNS), where traditional PKI-like key management mechanism is not suitable because of lack of flexibility and scalability. In SNS, people often share pictures, videos, and personal comments with friends. Since these contents usually contain privacy related information, it is desirable to protect them from the service provider and unauthorized users. Specifically, each user can generate her own public key and publish it on her homepage of SNS. Users who want to share data with others establish relationships by generating re-encryption keys. With re-encryption provided by SNS or other web services, individual users can share personal contents securely and conveniently.

Towards a public cloud-based solution, the proxy in CL-PRE runs in potentially malicious execution environment, where any vulnerability in cloud platform (e.g., in virtualization layer of a cloud node) can compromise the proxy and enable an attacker to perform data re-encryption for any unauthorized user. To address this, we propose two extensions for CL-PRE. First, we introduce *multi-proxy CL-PRE*, where multiple proxies are deployed in different cloud providers. An attacker has to compromise partial or all of them in order to achieve the attacking goal, therefore the security is significantly enhanced. Second, we introduce *randomized CL-PRE*, which further reduces the trust on a proxy by randomizing re-encryption key for each request, such that a proxy cannot perform re-encryption without obtaining a new re-encryption key from the data owner, therefore cannot abuse the data sharing capability.

We have implemented all CL-PRE schemes and evaluated their performance for data owner, proxy in cloud, and data recipient. Our results confirm that CL-PRE is a practical solution towards flexible and large-scale data sharing with cloud.

In summary, our contribution is 3-fold in this paper.

1. We propose a new certificateless proxy re-encryption scheme called CL-PRE for flexible data sharing with public cloud, which preserves final access control for data owner without completely trusting cloud infrastructure. CL-PRE uniquely leverages identity as partial of a user's public key and eliminates key escrow problem. We prove that CL-PRE is CPA-secure in random oracle model.

2. We extend CL-PRE to support multiple proxies deployed on different cloud providers, and randomized re-encryption keys for different data sharing sessions. Our extensions further enhance security and reduce trust for proxy running in public cloud.

3. We propose a method to accelerate re-encryption operations in cloud. We conduct communication and performance evaluation for data owner, proxy, and data recipient. Our results confirm that our schemes can satisfy performance requirement for large scale data sharing.

The remainder of this paper is organized as follows. In Section 2 we describe the trust model and overall architecture for cloud based data sharing. Section 3 presents the basic CL-PRE scheme with security analysis, and Section 4 presents two security enhanced extensions for CL-PRE. We give performance evaluations in Section 5 and summarize related work in Section 6. Section 7 concludes this paper.

2. OVERVIEW

In this section we describe security assumptions and the overall architecture of CL-PRE for cloud based data sharing.

2.1 Trust Model and Assumptions

We assume cloud is semi-trusted. This means that cloud works fairly by following pre-defined protocols and policies between end users and cloud services, e.g., upon client agreement. Yet with the high complexity of public cloud environment, cloud is not able to guarantee data confidentiality. The corruption of data security may be caused by social attacks towards cloud administrators, or by attacks that take advantage of security vulnerabilities of cloud infrastructure. However, we assume that cloud is able to achieve security over critical data, which includes the integrity and availability of public keys and access control policies.

We further assume that there exists a private key generator (PKG) that is able to generate part of private keys based on users' identities and securely deliver these keys to cloud users. We also assume a cloud client has basic capabilities on generating and managing different types of keys. In addition, a client is able to make its own data secure. We do not consider data re-dissemination after a legitimate user successfully decrypts a protected data.

2.2 Overall Architecture

As Figure 1 shows, a cloud user, named *data owner*, shares data to a number of other cloud users called *recipients*. A data is first encrypted with a symmetric data encryption key (DEK) by its owner, and then stored in the cloud, along with an access control list (ACL) indicating the recipient group. The data owner also encrypts the DEK using its public key,

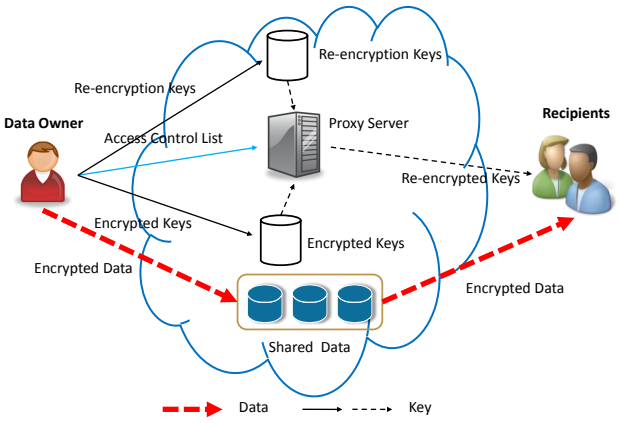

Figure 1: Overview of CL-PRE for data sharing.

and sends the encrypted DEK to the cloud as well. Upon access request from a recipient, based on the ACL, a *proxy* server in the cloud takes a re-encryption key sent from the data owner, and uses a re-encryption algorithm to transfer the encrypted DEK into the format that can be decrypted by the recipient's private key. The recipient then can download the encrypted data from the cloud and use the DEK for decryption.

A data owner may share different files with different *recipient groups*. For each of these groups, it uses a unique DEK. Therefore, a recipient cannot read data for a group it does not belong to. The cloud, on the other hand, acts as an intermediate proxy making data understood among cloud users. It cannot read the data as it cannot get DEKs.

A re-encryption key is generated from the data owner's private key and a recipient's public key. Since the number of cloud users participating in file sharing may be large, traditional PKI based approach has the public key management issue, and IBE based approach has the private key escrow problem. Uniquely, we adopt certificateless based encryption [2] in our re-encryption scheme.

3. CL-PRE SCHEME

This section first presents the basic CL-PRE scheme, then proves that this scheme is CPA-secure, and then analyzes its security properties and key update issue.

3.1 Main Algorithm

We denote user `A` the data owner, user `B` a data recipient, and proxy a cloud-resident service.

PKG Setup. Let G_1, G_2 be two cyclic groups of prime order p, and $e : G_1 \times G_1 \to G_2$ be a bilinear map. The message space is G_2. H_1 is a hash function from $\{0,1\}^*$ to G_1, and H_2 is a hash function from G_2 to G_1. A random generator $g \in G_1$ is chosen. The PKG randomly picks an integer $s \in \mathbb{Z}_p^*$ as the `master key`, and publishes g^s.

Partial Private Key Extraction. User `A` with identity $\texttt{ID}_A$ asks the PKG for partial private key extraction. The PKG calculates $g_A = H_1(\texttt{ID}_A), D_A = g_A^s$ and sends D_A to user `A`. The same operation with user `B`.

Secret Value Generation. User `A` randomly chooses an integer $x_A \in \mathbb{Z}_p^*$. The same with user `B`.

Private Key Generation. User `A` computes private key:

$$sk_A = D_A^{x_A} = g_A^{s \cdot x_A}. \quad (1)$$

The same operation is performed by user `B`. To achieve decryption delegation (to allow others to decrypt data that is originally encrypted with `A`'s public key), `A` also chooses a random integer t. sk_A and t are kept secret.

Public Key Generation. User `A` computes her public key:

$$pk_A = (g_A, g^{s \cdot x_A}), \quad (2)$$

where $g_A = H_1(\texttt{ID}_A)$. pk_A is published and anyone who wants to send `A` a message can use this for encryption. Note that g_A can be calculated by everyone from user `A`'s identity. For decryption delegation, user `A` also publishes g^t as part of her public key. The same operation is performed by `B`.

Note that a public key can be derived by user independently, i.e., the user does not need to contact PKG for public key generation.

Encryption. To encrypt a message $m \in G_2$ that can only be decrypted by herself, user `A` randomly chooses integer r and calculates

$$c = C_A(m) = (g^r, m \cdot e(g_A^r, g^{s \cdot x_A})).$$

For decryption delegation, user `A` also randomly chooses integers r and calculates

$$c' = C'_A(m) = (g^{tr}, g^r, m \cdot e(g_A^r, g^{s \cdot x_A})).$$

In our data sharing architecture illustrated in Section 2, m is the DEK (or DEK can be derived from m) for a sharing group generated by user `A`.

Decryption. To decrypt $C_A(m) = (u, v)$ under sk_A, user `A` calculates

$$v / e(sk_A, u) = m \cdot e(g_A, g^{s \cdot x_A})^r / e(g_A^{s \cdot x_A}, g^r) = m.$$

Proxy Re-encryption Key Generation. To delegate decryption right to user `B`, user `A` randomly chooses $x \in G_2$, and computes a proxy re-encryption key:

$$rk_{A \to B} = (g_A^{-s \cdot x_A} \cdot H_2^t(x), C_B(x)), \quad (3)$$

which is then sent to the proxy by user `A`.

Proxy Re-encryption. To re-encrypt a ciphertext $C'_A(m)$ under re-encryption key $rk_{A \to B}$, the proxy computes

$$\begin{aligned} c'' &= m \cdot e(g_A, g^{s \cdot x_A})^r \cdot e(g_A^{-s \cdot x_A} \cdot H_2^t(x), g^r) \\ &= m \cdot e(H_2^t(x), g^r), \end{aligned}$$

and then sends $(g^{tr}, c'', C_B(x))$ to user `B`.

Re-encryption Decryption. After receiving $(g^{tr}, c'', C_B(x))$, user `B` decrypts $C_B(x)$ to get x, and then gets the message by computing

$$c'' / e(H_2(x), g^{tr}) = m.$$

Hash Functions H_1 and H_2. The CL-PRE scheme in this section uses a hash function $H_1 : \{0,1\}^* \to G_1$ and a hash function $H_2 : G_2 \to G_1$. For a concrete system, G_1 is usually a subgroup of points on elliptic curve and G_2 a finite field. In order to construct H_1 and H_2, we introduce the concept of *Admissible Encoding* [8]. A concrete admissible encoding function `MapToPoint` for elliptic curve $y^2 = x^3 + 1$ over $\mathbb{F}_p$ was given in [8], which maps an element of $\mathbb{F}_p$ to G_1.

For hash function H_1 in our scheme, we choose a hash function with appropriate output length (for example, if $|p| = 512$, we choose `SHA-512`) and make the `ID` as input, and reduce the output by p to get an element of $\mathbb{F}_p$. We then use `MapToPoint` to map to G_1. For hash function H_2, we first map element of G_2 to $\{0,1\}^*$ by representing element of G_2 in binary form and then use above method to map to G_1. As described in [8], these techniques do not affect the security property of the scheme.

3.2 Security Analysis of CL-PRE

The security of CL-PRE is based on the assumed intractability of the Decisional Bilinear Diffie-Hellman problem (DBDH), which is defined as follows: Let (G_1, G_2) be a pair of bilinear groups of prime order p and with an efficiently computable bilinear pairing $e : G_1 \times G_1 \to G_2$, and let g be a random generator of G_1. The DBDH problem is to decide, when given a tuple of values $(g, g^a, g^b, g^c, T) \in G_1^4 \times G_2$ (where $a, b, c \in_R \mathbb{Z}_p$), whether $T = e(g,g)^{abc}$ or T is a random element of G_2.

For CL-PRE, we assume an adversary can extract partial private key or private key of a recipient, or both, for identities of their choice. As a proxy re-encryption scheme, the adversary has access to re-encryption oracle and re-encryption key generation oracle. If an adversary can break our scheme, it can solve the DBDH problem. This also implies that CL-PRE is collusion resistant, i.e., collusion between a recipient and the proxy cannot recover the private key of a data owner.

We adopt the security proof techniques from [21] and [8]. Let $\mathcal{A}$ be an p.p.t algorithm that has non-negligible advantage ϵ in attacking the scheme. We use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ to solve DBDH problem. Algorithm $\mathcal{B}$ accepts a tuple (g^a, g^b, g^c, T) as input and outputs 1 if $T = e(g,g)^{abc}$.

Oracle Queries.

$\mathcal{B}$ simulates the random oracle $H_1 : \{0,1\}^* \to G_1$ as follows: For a query `ID` which does not have been queried, $\mathcal{B}$ picks $x, z, t \overset{R}{\leftarrow} \mathbb{Z}_p^*$ and randomly flips a weighted coin to set $\alpha \leftarrow 1$ with probability γ, and $\alpha \leftarrow 0$ otherwise. If $\alpha = 0$, set $h \leftarrow (g^c)^z$, else compute $h \leftarrow g^z$. $\mathcal{B}$ records the tuple $(\texttt{ID}, h, x, z, t, \alpha)$ and returns h as the result.

$\mathcal{B}$ simulates the random oracle $H_2 : G_2 \to G_1$ by returning a random element of G_1 as a respond to a query.

Simulation Process.

1. `Setup.` $\mathcal{B}$ generates parameters $(G_1, G_2, H_1, H_2, g, g^a)$ and gives this tuple to $\mathcal{A}$. Here a is corresponding to the master secret of PKG.

2. `Query Phase I.`

   (a) When $\mathcal{A}$ submits the query `(extractPartialPrivateKey, ID)`, $\mathcal{B}$ evaluates $H_1(\texttt{ID})$ to obtain $(\texttt{ID}, h, x, z, t, \alpha)$, and returns $((g^a)^z)$ to $\mathcal{A}$.

   (b) When $\mathcal{A}$ submits the query `(extractPrivateKey, ID)`, $\mathcal{B}$ evaluates $H_1(\texttt{ID})$ to obtain $(\texttt{ID}, h, x, z, t, \alpha)$, and returns $((g^a)^z)^x$ to $\mathcal{A}$.

   (c) When $\mathcal{A}$ submits the query `(extractPublicKey, ID)`, $\mathcal{B}$ evaluates $H_1(\texttt{ID})$ to obtain $(\texttt{ID}, h, x, z, t, \alpha)$, and returns $(h, (g^a)^x)$ to $\mathcal{A}$.

   (d) When $\mathcal{A}$ submits the query `(extractReEncryptionKey, ID`$_1$`, ID`$_2$`)`, $\mathcal{B}$ evaluates $H_1(\texttt{ID}_1)$ and $H_1(\texttt{ID}_2)$ to get two tuples

   $$(\texttt{ID}_1, h_1, x_1, z_1, t_1, \alpha_1), (\texttt{ID}_2, h_2, x_2, z_2, t_2, \alpha_2).$$

   $\mathcal{B}$ also picks $r \overset{R}{\leftarrow} \mathbb{Z}_p^*, x \overset{R}{\leftarrow} G_1$ and $X \overset{R}{\leftarrow} G_2$. If $\alpha_1 = 0$, $\mathcal{B}$ returns the re-encryption key

   $$rk_{\texttt{ID}_1 \to \texttt{ID}_2} = (x, (g^b)^r, X \cdot T^{r z_2 x_2})$$

   to $\mathcal{A}$. If $\alpha_1 = 1$, $\mathcal{B}$ returns the re-encryption key

   $$rk_{\texttt{ID}_1 \to \texttt{ID}_2} = ((g^a)^{-z_1 x_1} \cdot H_2(X)^{t_1}, C_{\texttt{ID}_2}(X))$$

   to $\mathcal{A}$.

3. `Challenge Phase.` $\mathcal{A}$ chooses $(\texttt{ID}^*, m_0, m_1)$ where $\texttt{ID}^*$ should not be trivial. We say $\texttt{ID}^*$ is trivial if in `Query Phase I`, $\mathcal{A}$ has extracted private key for $\texttt{ID}'$ and a re-encryption key from $\texttt{ID}^*$ to $\texttt{ID}'$. Then $\mathcal{B}$ evaluates $H_1(\texttt{ID}^*)$ and gets $(\texttt{ID}^*, h, x, z, t, \alpha)$. $\mathcal{B}$ chooses $i \in_R \{0,1\}$ and returns $(g^b, m_i \cdot T^{zx})$ to $\mathcal{A}$.

4. `Query Phase II.` $\mathcal{A}$ makes queries as in the `Query Phase I`, but $\mathcal{A}$ is not allowed to make any query that will make the challenge trivial.

5. `Guess Phase.` $\mathcal{A}$ guesses which message is corresponding to the cipher, and outputs its guess $i' \in \{0,1\}$.

   Let α_i be the value of α generated by $H_1(\texttt{ID}_i)$. Prior to outputting a value, $\mathcal{B}$ verifies following conditions:

   (a) The value α corresponding to $\texttt{ID}^*$ is 0.

   (b) For each of $\mathcal{A}$'s queries `(extractPrivateKey, ID`$_i$`)` and `(extractPublicKey, ID`$_i$`)`, $\alpha_i = 1$.

   (c) For each of $\mathcal{A}$'s queries `(extractReEncryptionKey, ID`$_i$`, ID`$_j$`)`, where $\texttt{ID}_i \to \texttt{ID}_j$ lies along a path leading from $\texttt{ID}^*$, $\alpha_j = 0$.

   (d) For each of $\mathcal{A}$'s queries `(extractReEncryptionKey, ID`$_i$`, ID`$_j$`)`, where $\texttt{ID}_i \to \texttt{ID}_j$ does not lie along a path leading from $\texttt{ID}^*$, $\alpha_i = 1$.

   If any of the above conditions are false, $\mathcal{B}$ aborts the simulation. If $\mathcal{B}$ does not abort, then $\mathcal{B}$ outputs 1 if $i' = i$, or else outputs 0.

THEOREM 1. *If $\mathcal{B}$ does not abort during the simulation game, $\mathcal{B}$ can solve DBDH problem with non-negligible advantage. Therefore CL-PRE is CPA-secure in random oracle model.*

PROOF. First we analyze the probability that $\mathcal{B}$ does not abort during the simulation game. Suppose $\mathcal{A}$ makes n_{key} public/private key queries in `Query Phase I` and `Query Phase II`, then the probability that $\mathcal{B}$ does not abort during this process is $\gamma^{n_{key}}$. The probability that $\mathcal{B}$ does not abort during the `Challenge Phase` is $1 - \gamma$. Also suppose $\mathcal{A}$ makes n_{rk} re-encryption key queries that the re-encryption keys do not lie along path leading from $\texttt{ID}^*$ and n_{rk}^* re-encryption key queries that the re-encryption keys lie along path leading from $\texttt{ID}^*$. During these processes, the probability that $\mathcal{B}$ does not abort is $(1-\gamma)^{n_{rk}} \gamma^{n_{rk}^*}$. So finally the probability that $\mathcal{B}$ does not abort in the whole process is $\gamma^{n_{key}+n_{rk}^*}(1-\gamma)^{1+n_{rk}}$. Let $n_{max} = max\{n_{key}, n_{rk}, n_{rk}^*\}$, because $0 < \gamma < 1$, it is easy to see that the probability that $\mathcal{B}$ does

not abort is greater than $\gamma^{2n_{max}}(1-\gamma)^{1+n_{max}}$. When $\gamma = (2n_{max})/(3n_{max}+1)$, the probability that $\mathcal{B}$ does not abort achieves the maxim value.

Next we show that $\mathcal{A}$'s view is identical to the real attack. If $\mathcal{B}$ does not abort in the simulation above, every public/private key and re-encryption key that does not lie along a path from $\mathtt{ID}^*$ is correctly formed, and only the re-encryption keys that lie along a path from $\mathtt{ID}^*$ are incorrectly formed. However, the adversary $\mathcal{A}$ cannot distinguish our simulation from a real world interaction in which the re-encryption keys have correct form. The heuristic argument for this is simple. Note that each correctly-formed re-encryption key $rk_{\mathtt{ID}_1 \to \mathtt{ID}_2}$ consists a value $(sk_{\mathtt{ID}_1})^{-1}H_2(X)^t$, where $X \in_R G_2$, and a ciphertext of X under $\mathtt{ID}_2$'s public key. An incorrectly formed re-encryption key replaces $(sk_{\mathtt{ID}_1})^{-1}H_2(X)^t$ with some value $x \in_R G_1$. This x can be expressed as $(sk_{\mathtt{ID}_1})^{-1} \cdot y$ for some unknown $y \in G_1$. So an adversary who can distinguish incorrectly formed re-encryption keys can be used to solve DBDH problem.

In conclusion, if $\mathcal{B}$ does not abort during the game, then $\mathcal{A}$'s view is identical to the real attack. Hence, when the input to $\mathcal{B}$ is a DBDH tuple, the challenge ciphertext C^* is a correct encryption of m_i under $\mathtt{ID}^*$ and hence $|Pr[i = i'] - 1/2| \geq \epsilon$. So the CL-PRE scheme is CPA-secure. □

3.3 Properties of CL-PRE

CL-PRE possesses several properties that satisfy security requirements for cloud based data sharing scenario.

1. Unidirectionality. This means a re-encryption key $rk_{A\to B}$ from user A to user B cannot be used to re-encrypt a message encrypted under user B's public key. For CL-PRE, the result of re-encrypting $C'_B(m)$ under $rk_{A\to B}$ is in the form of $m \cdot e(g_B, g^{s\cdot x_B})^r \cdot e(g_A^{-s\cdot x_A} \cdot H_2^t(x), g^r)$, which cannot be decrypted using A's private key.

   In cloud based data sharing scenario, this is a nature requirement because sharing is also unidirectional. That is, if user A wants to obtain data of B, she must ask B for a re-encryption key rather than generate an re-encryption key by herself.

2. Non-interactivity. This means in order to generate a re-encryption key from A to B, user A does not need to interact with B. For CL-PRE, it is easy to see that user A can generate a re-encryption key for user B with only B's public key. So the re-encryption keys can be generated while the recipients are off-line.

   This property is also very useful in data sharing scenario because it is not practical to require users are always on-line for re-encryption key generation.

3. Nontransitive. This means it is infeasible to construct $rk_{A\to C}$ from $rk_{A\to B}$ and $rk_{B\to C}$. In CL-PRE, an adversary (proxy or ordinary users) can generate the second part of an re-encryption key at will, but cannot generate the first part and make the two parts match.

   This property is desirable in our scenario because the proxy manages all existing re-encryption keys, and if the algorithm is transitive, the proxy can create re-encryption keys without the permission of legitimate users.

4. Single-use. This means that an re-encrypted message cannot be further re-encrypted. Due to the structure of the re-encrypted message in CL-PRE, it cannot be further re-encrypted.

   This property is useful because if the scheme is not single-use, an adversary who holds proper re-encryption key and re-encrypted message can first re-encrypt and then decrypt to recover the original message without permission of data owner.

3.4 Key Update

Key update is an essential part for data sharing applications. Occasionally, users may update their public/private key pairs. Key update also offers a way for revocation of existing re-encryption keys. Because the identity of a user rarely changes, in most cases key update means the update of the *Secret Value*.

Public/private key update results in related re-encryption key update. If many re-encryption keys have been generated using old keys, re-encryption key update is a big burden for users. In this section we give a method to shift this burden to cloud.

Note that the second part of a re-encryption key is in fact an encryption of a random element from G_2 under a recipient's public key, and this part does not participate the re-encryption operation. Furthermore, this part can be cached by the recipient after the first time it receives a re-encrypted message. So we do not consider the update of this part.

If user A wants to update her public/private key pair, she picks $x'_A \in_R \mathbb{Z}_p^*$, sets new private key $sk'_A = D_A^{x'_A} = g_A^{s\cdot x'_A}$ and new public key $pk'_A = (g_A, g^{sx'_A}, g^{tx'_A/x_A})$.

In order to utilize the cloud to update related re-encryption keys, user A calculates a number $r_{A'/A} = x'_A/x_A \ mod\ p$ and sends to the proxy.

After receiving $r_{A'/A}$, the proxy updates related re-encryption key $rk_{A\to B}$ as follows:

$$\begin{aligned} rk'_{A\to B} &= ((g_A^{-s\cdot x_A} \cdot H_2(x))^{t\cdot r_{A'/A}}, C_B(x)) \\ &= (g_A^{-s\cdot x'_A} \cdot H_2(x)^{t\cdot x'_A/x_A}, C_B(x)). \end{aligned}$$

User A should keep a list of her *Secret Value*s and when she receives ciphertexts encrypted under her old public keys, she can still decrypt them correctly.

Note that no matter how many re-encryption keys user A has established, she only needs to compute and send one value $r_{A'/A}$. The remainder update work is done by the cloud.

Next we show the correctness of the updated re-encryption key. Consider one encrypted message under the new public key of user A in the form

$$(g^{tr\cdot x'_A/x_A}, g^r, m \cdot e(g_A^r, g^{sx'_A})).$$

For re-encryption, proxy computes

$$\begin{aligned} &(m \cdot e(g_A^r, g^{sx'_A})) \cdot e(g_A^{-s\cdot x'_A} \cdot H_2(x)^{t\cdot x'_A/x_A}, g^r) \\ =&m \cdot e(H_2(x)^{t\cdot x'_A/x_A}, g^r). \end{aligned}$$

Then the recipient B can decrypt it by computing

$$m \cdot e(H_2(x)^{t\cdot x'_A/x_A}, g^r)/e(H_2(x), g^{tr\cdot x'_A/x_A}) = m.$$

4. EXTENSIONS OF CL-PRE

In this section, we propose two variants of CL-PRE that offer enhanced security and better computation performance

of re-encryption at cloud. We then give a method to accelerate the re-encryption operation at cloud proxy side.

4.1 Multi-Proxy CL-PRE

As residing in public cloud, the proxy in CL-PRE runs in untrusted environment, e.g., it may be exposed to attacks due to misconfiguration of cloud system administrators, or compromised by external attacker with virtualization vulnerability in cloud computing platforms. Towards a security enhanced solution, we propose *multi-proxy CL-PRE*, an extension of CL-PRE to support multiple proxies deployed on different clouds, such that exploiting the system requires compromising multiple proxies.

Assume there are n proxies in the system. User A (data owner) can choose a parameter k $(1 < k < n)$ s.t., when more than k proxies do the re-encryption correctly, the recipient can get a valid message. In other words, user A can achieve higher security level by choosing larger k. We use techniques in [32] to construct multi-proxy CL-PRE. Note that in our scheme, what we want to split are elements of abelian group other than finite field, so we cannot adopt the method of [32] directly. There are also some secret sharing schemes that can work on abelian group [14, 15], but these methods can neither be adopted directly because we have to execute bilinear pairing on the split elements.

We use Lagrange polynomial interpolation method for constructing multi-proxy CL-PRE. Suppose we have a univariate polynomial $f(x)$ of degree k, and $k+1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_k, y_k)$ s.t. $y_i = f(x_i)$ and $x_i \neq x_j$ for $i \neq j$. Then $f(x) = \sum_{i=0}^{k} y_i L_i(x)$, where

$$L_i(x) = \prod_{0 \leq m \leq k, m \neq i} \frac{x - x_m}{x_i - x_m}$$

For the multi-proxy CL-PRE, the algorithms for *Setup, Private Key Extraction, Secret Value Generation, Private Key Generation, Public Key Generation, Encryption, Decryption* are the same as that in CL-PRE. The others are as follows.

Proxy Re-encryption Key Generation. If user A wants to delegate decryption right to user B, user A chooses $x \in_R G_2$ and two random polynomials $f(x), h(x)$. $f(x)$ and $h(x)$ satisfy that $\deg(f) = \deg(h) = k$ and $f(0) = -sx_A, h(0) = t$. A also generates n random number $n_i \neq 0 (1 \leq i \leq n)$. These numbers are public information. Then A evaluates $f(x)$ and $h(x)$ at these numbers to gain two sets of points $(n_i, f(n_i)), (n_i, h(n_i))$. For the ith proxy the re-encryption key is

$$rk_{A \to B}^{(i)} = (g_A^{f(n_i)} H_2(x)^{h(n_i)}, C_B(x)).$$

Proxy Re-encryption. To re-encrypt a ciphertext $C'_A(m)$ under re-encryption key $rk_{A \to B}^{(i)}$, the ith proxy computes

$$c_i = e(g_A^{f(n_i)} H_2(x)^{h(n_i)}, g^r),$$

and sends to user B.

Re-encryption Decryption. After receiving $k+1$ re-encrypted message (without lose of generality, suppose received messages are $c_1, c_2, \ldots, c_{k+1}$) and the original cipher $C'_A(m)$, user B decrypts as follows:

1. Computes

$$L_i(0) = \prod_{0 < m \leq k+1, m \neq i} \frac{-n_m}{n_i - n_m}, 1 \leq i \leq k+1.$$

Note that $L_i(0)$ is independent of concrete polynomial.

2. Computes

$$\begin{aligned} c_{part} &= \prod_{i=1}^{k+1} c_i^{L_i(0)} = \prod_{i=1}^{k+1} e(g_A^{f(n_i)} H_2(x)^{h(n_i)}, g^r)^{L_i(0)} \\ &= \prod_{i=1}^{k+1} e(g_A, g)^{r f(n_i) L_i(0)} e(H_2(x), g)^{r h(n_i) L_i(0)} \\ &= e(g_A, g)^{-r s x_A} \cdot e(H_2(x), g)^{rt}. \end{aligned}$$

The last equation holds because with Lagrange polynomial interpolation, we have $\sum_{i=1}^{k+1} f(n_i) L_i(0) = f(0) = -sx_A$ and $\sum_{i=1}^{k+1} h(n_i) L_i(0) = h(0) = t$.

3. Finally, B computes

$$\begin{aligned} &m \cdot e(g_A, g^{s \cdot x_A})^r \cdot c_{part} / e(H_2(x), g^{tr}) \\ &= m \cdot e(H_2(x)^t, g^r) / e(H_2(x), g^{tr}) = m. \end{aligned}$$

Optimization. If user A needs to generate re-encryption keys for many recipients, she can use the same $f(x), h(x)$ and the same set of the n random numbers for all of them. In this case, A needs to compute $g_A^{f(n_i)}$ and $h(n_i)$ only once. The proxies can also save time by computing $L_i(0)$s once.

Other Advantages. Besides offering improved security, multi-proxy CL-PRE has other advantages. First, it makes the whole system more robust, as data sharing can be accomplished if more than k proxies work properly. Secondly, recipients can choose the proxies to fetch re-encrypted data according to physical location or working load. This increases system deployment flexibility and content delivery efficiency.

4.2 Randomized CL-PRE

Multi-proxy CL-PRE does not solve the *semi-trust* problem completely. If the number of compromised proxies is big enough, the system can be compromised. Instead of building complete secure running environment for proxy, our next extension of CL-PRE takes a different approach.

We note that the fundamental reason that we have to put some trust on proxy is that the proxy has full control of all re-encryption keys after receiving them from a data owner, and the data owner is lack of capabilities to restrict the usage of re-encryption keys. If the data owner manages re-encryption keys by herself and does not expose them to the proxy, we can further reduce trust on proxy. Towards this, our idea is similar to the pairing computation delegation [33].

Specifically, after generating a re-encryption key, user A does not send it to proxy. Instead, every time A wants to share data to a recipient, she generates a one-time re-encryption key for this session. The one-time key is a randomization of the original re-encryption key, and the cost of generating one-time re-encryption key is lower than that of generating the original re-encryption key. Then A sends the randomized key along with the data to the proxy. The randomized key can only be used to re-encrypt shared data in the same session. So it is useless for the proxy to control randomized re-encryption keys. We call this scheme *randomized CL-PRE*. The algorithms are explained as follows.

Encryption. User A picks $r_1, r_2 \in_R \mathbb{Z}_p$ and computes

$$c' = (g^{t r_1 r_2}, g^{r_1}, m \cdot e(g_A^{r_1}, g^{s x_A})).$$

Proxy Re-encryption. User A uses r_2 to randomize the re-encryption key. For original re-encryption key

$$rk_{A\to B} = (g_A^{-sx_A} \cdot H(x)^t, C_B(x)),$$

user A randomizes it to

$$rk'_{A\to B} = (g_A^{-sx_A} \cdot H(x)^{tr_2}, C_B(x)), \quad (4)$$

and sends $rk'_{A\to B}$ to proxy.

Proxy re-encrypts c' using $rk'_{A\to B}$ in the following way:

$$\begin{aligned} c'' &= m \cdot e(g_A^{r_1}, g^{sx_A}) \cdot e(g_A^{-sx_A} \cdot H(x)^{tr_2}, g^{r_1}) \\ &= m \cdot e(H_2(x)^{tr_2}, g^{r_1}). \end{aligned}$$

Decryption. User B decrypts c'' by computing

$$c''/e(H_2(x), g^{tr_1r_2}) = m.$$

Using above method, proxy does not possess any "long term" re-encryption key but a one-time randomized one. The proxy cannot complete a re-encryption of any new message with previous re-encryption keys. Suppose the proxy receives a ciphertext $c' = (g^{tr_1r_2}, g^{r_1}, m \cdot e(g_A^{r_1}, g^{sx_A}))$, where r_1, r_2 are random numbers different from previous encryption, but only has an older re-encryption key $rk'_{A\to B} = (g_A^{-sx_A} \cdot H(x)^{t \cdot r_2^{old}}, C_B(x))$. The re-encryption result is

$$\begin{aligned} c'' &= m \cdot e(g_A^{r_1}, g^{sx_A}) \cdot e(g_A^{-sx_A} \cdot H(x)^{t \cdot r_2^{old}}, g^{r_1}) \\ &= m \cdot e(H_2(x)^{t \cdot r_2^{old}}, g^{r_1}). \end{aligned}$$

The recipient cannot decrypt because it does not possess $g^{tr_1r_2^{old}}$.

A data owner can choose this method and does not need to worry about abusing of re-encryption keys by the proxy. However, this security enhancement is not free. Each time the data owner shares data with m users, besides the encryption of one message, she has to randomize m re-encryption keys and sends these re-encryption keys to the proxy. We include the communication and performance analysis in Section 5. Some methods can be applied to reduce the computation cost for the data owner. For example, if user A shares with a fixed group of users for a period of time, she does not need to generate new randomized re-encryption key every time. Instead, she generates a new randomized re-encryption key only when the group of recipients changes.

4.3 Acceleration of Proxy Re-encryption

In CL-PRE, for one re-encryption operation, a bilinear pairing $e(rk_i, g^r)$ has to be performed by the cloud proxy, where rk_i is part of the re-encryption key from the data owner to user i, and g^r is part of the encrypted message to be re-encrypted. When the data owner shares with m users, m bilinear pairings $e(rk_i, g^r), i = 1, 2, \ldots, m$ have to be computed, where the second input of the bilinear pairing is fixed, and the first input varies for different users. We can also fix the first input using property of bilinear pairing, specifically, $e(rk_i, g^r) = e(g^r, rk_i)^{-1}$.

No matter which type of pairing is chosen, Miller's algorithm [28] or its variants such as [18] can be used to compute the pairing value. Miller's algorithm is an iterative algorithm, and the first input is used to construct linear functions $\ell_{r,s}(\cdot)$ and $\ell_t(\cdot)$ (in some situations the computation of $\ell_t(\cdot)$ may be eliminated [4]), where $\ell_{r,s}$ is a line passing through r, s, ℓ_t is a vertical line passing through t, and r, s, t only depend on g^r. Then the algorithm evaluates these two functions on rk_i. So for our scheme, as the first input of the bilinear pairing is the same for all m re-encryptions, $\ell_{r,s}(\cdot)$ and $\ell_t(\cdot)$ has to be computed only once and evaluated at different points. We also includes some experimental results in Section 5.

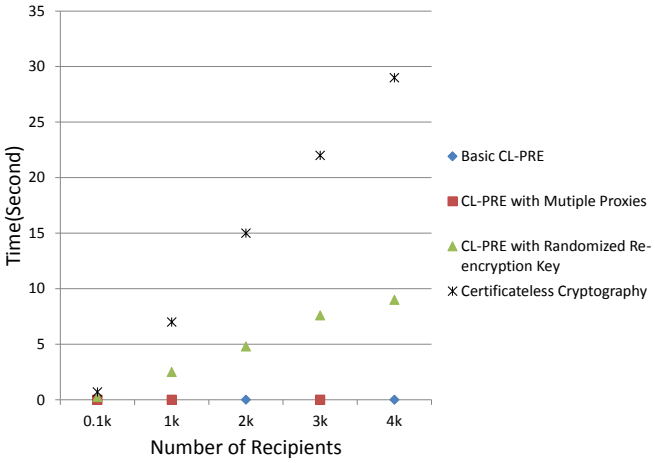

Figure 2: Performance overhead for data owner.

5. PERFORMANCE EVALUATION

In this section, we evaluate the performance of data owner, cloud proxy, and recipient in CL-PRE and its variants, which are implemented on an elliptic curve defined on 512 bits prime field with a generator of order 160 bits. The embedded degree of the curve is 2. We also implement the basic version of certificateless public key encryption scheme [2], which is referred as *certificateless cryptography* in the reminder of this section. Our experiments are carried out on one-core, 1GB RAM Linux virtual machine residing on a PC with Intel i5 3.4GHz processor and 4GB RAM.

5.1 Re-encryption Performance of CL-PRE

We accelerate the re-encryption process using acceleration techniques in Section 4.3, which makes each of the re-encryption 20% faster. Table 1 summarize the cost of the basic CL-PRE scheme. Our experimental result shows that with 3k bits of both re-encryption key size and ciphertext size, the proxy re-encryption time is about 7-8 ms. With elastic computing resources in cloud, we believe the proxy is not a performance bottle neck.

Table 1: Re-encryption performance summary

Re-encryption time	Re-encryption key size	Ciphertext size
7 to 8 ms	3K bits	3K bits

5.2 Performance of Data Owner

Figure 2 shows the computation overhead of data owner in variant schemes with different numbers of data recipients. As shown, the computation overhead of the data owner does not increase with the number of recipients for basic CL-PRE and multi-proxy CL-PRE, as it only does one encryption in these two cases.

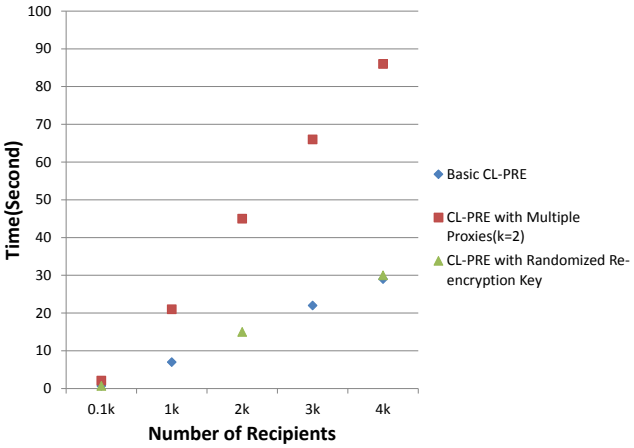

Figure 3: Performance overhead for proxy.

With certificateless cryptography and randomized CL-PRE, the data owner has to do operations for each recipient. So the cost increases linearly with the number of recipients. However, as the encryption needs one bilinear pairing while the randomization needs only one scalar multiplication, encrypting a message is more complex and costly than randomizing one re-encryption key, as indicated in Figure 2. We note that the cost of randomizing re-encryption keys in our experiment is the worst case since we choose the largest number (160 bits long) as scalar. In practical, random number should be used and the average cost should be 50% lower.

Next we analyze communication and storage overhead of data owner in different cases. For the basic CL-PRE and multi-proxy CL-PRE, the communication overhead is constant and is not relevant to the number of recipients, and the data owner only has to keep its own public/private key pair. For randomized CL-PRE, the data owner has to transport one encrypted message and one randomized re-encryption key for each recipient. Note that the second part of randomized re-encryption key in randomized CL-PRE (cf. Equation 4) is not changed and not sent to the proxy every time. In addition, the data owner has to store all the generated re-encryption keys.

5.3 Performance of Proxy

Figure 3 shows the computation overhead of the proxy server in different schemes. We omit the case of certificateless cryptography as in this case the cloud does not need to do any computation. In all three cases, the overhead increases linearly with the number of recipients. The basic CL-PRE and randomized CL-PRE have the same overhead in proxy re-encryption, so the curves coincide. For multi-proxy CL-PRE, the computation cost increases not only accompanied by the number of recipients but also the parameter k (cf. Section 4.1). However, the re-encryption process can be parallelized in this scheme. If there are $k+1$ computation units, the computation time is only related to the number of recipients. Figure 3 reflects the situation that only one computation unit is available.

Next we analyze the communication and storage overhead of cloud proxy in different cases. For the basic CL-PRE, the cloud stores $O(m^2)$ re-encryption keys, where m is the number of users in the system. For one data sharing operation, the proxy transports one re-encrypted message for one recipient, i.e., the communication overhead is proportional to the number of recipients. For randomized CL-PRE, if the proxy does not cache randomized re-encryption keys, there is no need of re-encryption storage, and the communication overhead is the same as that of basic CL-PRE. For multi-proxy CL-PRE, the proxy stores about $O(n \cdot m^2)$ re-encryption keys (n is the number of proxies in the system). This is because for one sharing relationship between two users, the proxy needs to store n re-encryption keys and there are totally $O(m^2)$ relationships. The communication overhead depends on parameter k. If there are m recipients, $(k+1)m$ re-encrypted messages are transported. We summarize communication and transportation cost of proxy for *one* recipient in Table 2 [1].

Table 2: Communication and transportation cost for proxy in cloud

Basic CL-PRE	Multi-proxy CL-PRE	Randomized CL-PRE
2K bits	6K bits (3×2K)	2K bits

5.4 Performance of Recipient

In case of the basic CL-PRE and randomized CL-PRE, there is no difference for a recipient – it obtains re-encrypted messages from the proxy and decrypts in the same way. For multi-proxy CL-PRE, the overhead increases with the parameter k. The extra overhead includes the computation of $L_i(0)$ (this can be cached if the recipient always uses the same set of proxies), $k+1$ exponentiations (scalar multiplication for elliptic curve), and k multiplications (point addition for elliptic curve). As a recipient has to do one decryption only, we believe this cost is not significant.

6. RELATED WORK

The research of proxy re-encryption has been started in [26] and [7] and the original aim is to find a method to transform ciphertext with one encryption key to another without decryption. In [23] the authors propose a formal modeling of proxy cryptography, which generalizes, simplifies, and clarifies the model of "atomic proxy" in [7]. In [3] the authors give criteria for proxy re-encryption scheme and propose a *unidirectional*, *noninteractive*, *proxy invisibility*, and *original access* proxy re-encryption scheme. But this scheme has a defect that malicious user can derive *weak secret* of a data owner from re-encryption keys and its private key. CL-PRE is similar to [3], yet considers more specifically the security for flexible data sharing with cloud. In addition, CL-PRE does not have the weak secret problem in [3].

Many other proxy re-encryption schemes with different security properties are constructed, such as identity based re-encryption scheme without random oracles [13], chosen-ciphertext secure proxy re-encryption [10], and unidirectional chosen-ciphertext secure proxy re-encryption [25].

Another line of related research is secure file system. The original work in this area is the Cryptographic File System

[1]Note that the third part of a re-encrypted message is not transported every time, and no compression method of elliptic curve point is used. We also suppose $k = 2$ for multi-proxy CL-PRE.

(CFS) [6], which uses a single key to encrypt an entire directory of files and relies on the underlying file system for authorization of writes. Later variants includes Cepheus [16], SNAD [27], SiRiUS [19], and Plutus [24].

Attribute based cryptography also offers a way for secure data sharing when *attributes* can be used to identify users. The original work of this area is fuzzy identity based encryption [30], which generalizes previous identity based encryption [8]. Later variants includes attribute-based encryption (ABE) schemes, such as KP-ABE (binding access policy to keys [20]), CP-ABE (binding access policy to ciphertext [5]), and distributed ABE [29]. Chase [11] provides a construction for a multi-authority ABE system, where each authority administers a domain of attributes. Chase and Chow [12] provide a more practice-oriented multi-authority ABE system, which removes the trusted central authority while preserving user privacy. There are other practices taking advantage of ABE [34]. But these techniques are unsuitable for scenarios with very decentralized trust relationships such as SNS.

Broadcast encryption is another way for secure data sharing. Fiat and Naor[17] first introduce the concept of broadcast encryption. Later many research efforts have been carried out on this topic. But the efficiency of the best known schemes such as [22] and [9], is not only dependent on the size of authorized user set, but also requires the broadcaster to refer to its database of user authorizations. Therefore broadcast encryption is more suitable for situation where the set of revoked users is small, which contradicts our scenario.

7. CONCLUSION

This paper proposes CL-PRE, a certificateless proxy re-encryption scheme for cloud-based data sharing. CL-PRE uniquely integrates identity-based public key into proxy re-encryption, eliminates the key escrow problem in traditional identity-based encryption, and does not require the use of certificates to guarantee the authenticity of public keys. CL-PRE satisfies security requirements for large-scale and flexible information sharing with cloud such as online social networks. Consider a proxy running in public cloud to leverage elastic cloud storage and computing resources, we further propose multi-proxy CL-PRE to deploy intermediate proxies in multiple cloud service providers in order to improve the robustness of the system, and randomized CL-PRE to randomize the re-encryption key each time of data sharing in order to reduce the trust on the proxy. Our performance evaluations confirm that our proposed schemes are practical for cloud-based applications.

8. REFERENCES

[1] AWS Customer Agreement `http://aws.amazon.com/agreement/`. 2011.

[2] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In C.-S. Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *LNCS*, pages 452 – 473. Springer - Verlag, 2003.

[3] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, 9:1 – 30, 2006.

[4] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In M. Matsui and R. J. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *LNCS*, pages 17 – 25. Springer - Verlag, 2003.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy – S&P 2007*, pages 321–334. IEEE Computer Society, 2007.

[6] M. Blaze. A cryptographic file system for unix. In *Proceedings of the 1st ACM conference on Computer and communications security – CCS 1993*, pages 9 –16. ACM, 1993.

[7] M. Blaze, G. Bleumer, and M. Strauss. Divertible protocols and atomic proxy cryptography. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Advances in Cryptology - EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 127 – 144. Springer-Verlag, 1998.

[8] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In J. Kilian, editor, *Advance in Cryptology –CRYPTO 2001*, volume 2139 of *LNCS*, pages 213 – 229. Springer-Verlag, 2001.

[9] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *LNCS*, pages 258 – 275. Springer - Verlag, 2005.

[10] R. Canetti and S. Hohenberger. Chosen-ciphertext secure proxy re-encryption. In D. C. di Vimercati, Sabrina, and P. Syverson, editors, *Proceedings of the 14th ACM conference on Computer and communications security – CCS 2007*, pages 185 –194. ACM, 2007.

[11] M. Chase. Multi-authority attribute based encryption. In S. P. Vadhan, editor, *Theory of Cryptography – TCC 2007*, volume 4392 of *LNCS*, pages 515 – 534. Springer - Verlag, 2007.

[12] M. Chase and S. S. Chow. Improving privacy and security in multi-authority attribute-based encryption. In E. Al-Shaer, S. Jha, and A. D. Keromytis, editors, *Proceedings of the 16th ACM conference on Computer and communications security – CCS 2009*, pages 121 –130. ACM, 2009.

[13] C. K. Chu and W. G. Tzeng. Identity-based proxy re-encryption without random oracles. In J. Garay, A. K. Lenstra, M. Mambo, and R. Peralta, editors, *Information Security – ISC 2007*, volume 4779, pages 189 – 202. Springer - Verlag, 2007.

[14] R. Cramer and S. Fehr. Optimal black-box secret sharing over arbitrary abelian groups. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 272 – 287. Springer - Verlag, 2002.

[15] Y. Desmedt, B. King, W. Kishimoto, and K. Kurosawa. A comment on the efficiency of secret sharing scheme over any finite abelian group. In C. Boyd and E. Dawson, editors, *Information Security and Privacy – ACISP 1998*, volume 1438 of *LNCS*, pages 391 – 402. Springer - Verlag, 1998.

[16] K. E.Fu. Group sharing and random access in cryptanalytic storage file syetems. Master's thesis, MIT, 1999.

[17] A. Fiat and M. Naor. Broadcast encryption. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO 1993*, volume 773 of *LNCS*, pages 480 – 491. Springer - Verlag, 1993.

[18] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the tate pairing. In C. Fieker and D. R. Kohel, editors, *Proceedings of the 5th International Symposium on Algorithmic Number Theory - ANTS V*, volume 2369 of *LNCS*, pages 324 – 337. Springer-Verlag, 2002.

[19] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh:. Sirius: Securing remote untrusted storage. In *Proceedings of the Network and Distributed System Security Symposium – NDSS 2003*, pages 131 – 45. Internet Society, 2003.

[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *Proceedings of the 13th ACM conference on Computer and communications security – CCS 2006*, pages 89 – 98. ACM, 2006.

[21] M. Green and G. Ateniese. Identity-based proxy re-encryption. In J. Katz and M. Yung, editors, *Applied Cryptography and Network Security – ACNS 2007*, volume 4251 of *LNCS*, pages 288 – 306. Springer - Verlag, 2007.

[22] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, number 2442 in LNCS, pages 47 – 60. Springer - Verlag, 2002.

[23] A. Ivan and Y. Dodis. Proxy cryptography revisited. In *Proceedings of the Network and Distributed System Security Symposium – NDSS 2003*. Internet Society, 2003.

[24] M. Kallahalla, E. Riedely, R. Swaminathan, Q. Wangz, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings 2nd USENIX Conference on File and Storage Technologies – FAST 2003:*, pages 29 – 42. USENIX, 2003.

[25] B. Libert and D. Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In R. Cramer, editor, *Public Key Cryptography – PKC 2008*, volume 4939 of *LNCS*, pages 360 – 379. Springer - Verlag, 2008.

[26] M. Mambo and E. Okamoto. Proxy cryptosystems: Delegation of the power to decrypt ciphertexts. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, E80-A:54 – 63, 1997.

[27] E. Miller, D. Long, W. Freeman, and B. Reed. Strong security for network-attached storage. In D. D. E. Long, editor, *Proceedings of the USENIX Conference on File and Storage Technologies – FAST 2002*, pages 1 – 13. USENIX, 2002.

[28] V. S. Miller. The weil pairing, and its efficient calculation. *Journal of Cryptology*, 17(4):235 – 261, September 2004.

[29] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In L. Chen, M. D. Ryan, and G. Wang, editors, *Information Security and Cryptology – ICISC 2008*, volume 5461 of *LNCS*, pages 20 – 36. Springer - Verlag, 2008.

[30] A. Sahai and B. Waters. Fuzzy identity-based encryption. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457 –473. Springer - Verlag, 2005.

[31] C. Security Alliance. Security Guidance for Critical Areas of Focus in Cloud Computing V2.1, 2009. `https://cloudsecurityalliance.org/csaguide.pdf`.

[32] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612 – 613, 1979.

[33] P. P. Tsang, S. S. M. Chow, and S. W. Smith. Batch pairing delegation. In A. Miyaji, H. Kikuchi, and K. Rannenberg, editors, *Advances in Information and Computer Security - IWSEC 2007*, volume 4752 of *LNCS*, pages 74 – 90. Springer - Verlag, 2007.

[34] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proceedings of the 29th conference on Information communications - INFOCOM 2010*, pages 534 – 542. IEEE Press, 2010.