POSTER: A Certificateless Proxy Re-Encryption Scheme for Cloud-based Data Sharing

Xiaoxin Wu†, Lei Xu†, and Xinwen Zhang‡
†Huawei Information Technology Lab, Beijing, China
‡Huawei America Research Center, Santa Clara, CA, USA
{wuxiaoxin, stone.xu, xinwen.zhang}@huawei.com

ABSTRACT

We propose CL-PRE, a certificateless proxy re-encryption scheme for data sharing with 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 for 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 with identity based cryptography and the need of certificate . While preserving data and key privacy from semi-trusted cloud, CL-PRE maximumly leverages cloud resources to reduce the computing and communication cost for data owner. We implement CL-PRE and evaluate its 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 security should be mainly relied on cloud customers instead of cloud service providers [1, 5].

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 to 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 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 the keys by itself, e.g., through sending the data encryption key to individual recipients with assumption that each user is equipped with a private/public key pair and data encryption key is encrypted with a recipient's public key. This introduces heavy computing load at data owner side, and relies on a PKI system, which does not scale 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 interest for any 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 an 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 respecting 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.

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 the

CCS'11, October 17–21, 2011, Chicago, Illinois, USA.
ACM 978-1-4503-0948-6/11/10.

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.

Towards a 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, which enables an attacker to perform data re-encryption for any unauthorized user. To address this, we further extend CL-PRE for multi-proxy encryption, with multiple proxies deployed in different cloud providers. An attacker has to compromise all of them in order to achieve the attacking goal, therefore the security is significantly enhanced.

Our proxy re-encryption algorithm is similar to [3], yet we consider more specifically flexible data sharing within cloud. An important novelty of our scheme is using certificateless public key cryptography 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 public key, while eliminates the key escrow problem in traditional identity-based encryption (IBE) [4], and does not require certificates to guarantee the authenticity of public keys. In addition, CL-PRE does not have a problem in [3] that the weak secret can be released if the proxy colludes with users.

2. CL-PRE SCHEME

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, 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 launched by an administrator, or by any outside attacks that take advantage of any security vulnerabilities of the cloud. However, we assume that cloud is able to achieve security over critical data. The critical data includes integrity of public keys and access control policies. We further assume that the operation righteousness over these critical data is guaranteed.

We assume a cloud client has basic capabilities on generating and managing different type 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 protect data.

We further assume that there exists a private key generator (PKG) that is able to generate part of private keys based on user identities and securely deliver these keys to cloud users.

2.2 Overview

As Figure 1 shows, a cloud user, named data owner, shares data to a number of other cloud users (recipients). The data is first encrypted with a symmetric data encryption key (DEK), and then stored in the cloud, along with an access control list (ACL) indicating the recipient group. It also encrypts the DEK using its public key, and sends the encrypted DEK to the cloud as well. Upon access request from a recipient, based on the ACL, the cloud uses a re-encryption algorithm to transfer the encrypted DEK into the format that can be decrypted by the recipient's private

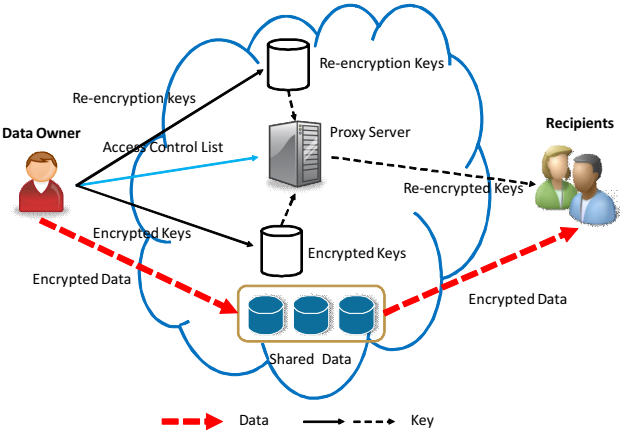

Figure 1: Overview of CL-PRE for data sharing.

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.

In our scheme, 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. Therefore certificateless based encryption [2] is adopted in our scheme.

2.3 Basic Algorithm

In following description, we denote user `A` the data owner, user `B` a data recipient, and proxy a cloud-resident service. Both user `A` and `B` obtains public/private key pairs as the following describes.

PKG Setup: Let G_1, G_2 be two cyclic groups of prime order p, and $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map. The message space is G_2, the cipher space is $G_1 \times G_2$. 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.

Private Key Extraction: User `A` with identity ID_A asks the PKG for partial private key extraction. The PKG calculates $g_A = H_1(ID_A), D_A = g_A^s$ 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:

$$sk_A = D_A^{x_A} = g_A^{s \cdot x_A},$$

where sk_A is kept secret. The same operation is performed by user `B`. To achieve decryption delegation (to allow others to decrypt data that is originally encrypted by `A`), `A` also chooses a random integer t.

Public Key Generation: User A computes her public key:

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

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.

Encryption: To encrypt message $m \in G_1$ that can only be decrypted by herself, user A randomly chooses integer r and calculates

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

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

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

In our data sharing scenario illustrated in Section 2, m is the DEK 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: If user A wants to delegate decryption right to user B, user A randomly chooses $x \in G_2$, and computes the proxy re-encryption key by

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

which is then sent to the proxy by user A.

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

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

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:

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

2.4 Security Analysis

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 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 if T is a random element of G_2.

For CL-PRE, assume an adversary can extract partial private keys or private keys of a recipient, or both, for identities of their choice. The adversary also can replace the public key of any entity with a value of its 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, this adversary can be used to 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 the data owner. Due to space limit we omit the proof.

2.5 Multi-Proxy Re-encryption

As residing in public cloud, the proxy in CL-PRE can be compromised, e.g., by cloud system administrators, or external attacker with virtualization vulnerability in cloud computing platforms. Towards a security enhanced solution, we extend CL-PRE to support multi-proxy scheme, where multiple proxies can be deployed on different clouds.

Assume there are n proxies, user A (data owner) divides the first part of a re-encryption key into n randomly partitioned pieces, i.e.,

$$g_A^{-s \cdot x_A} \cdot H_2^t(x) = rk_1 \cdot rk_2 \cdot \ldots \cdot rk_n,$$

and sends rk_i to proxy i, respectively.

To re-encrypt a ciphertext $C'_A = c' = (g^{tr}, g^r, m \cdot e(g_A, g^{s \cdot x_A})^r)$, proxy i calculates $c_i = e(rk_i, g^r)$ and sends

$$(g^{tr}, m \cdot e(g_A, g^{s \cdot x_A})^r, c_i, C_B(x))$$

to user B.

After receiving all of the n re-encrypted ciphertext, user B first computes

$$\prod_{i=1}^{n} c_i = \prod_{i=1}^{n} e(rk_i, g^r) = e(\prod_{i=1}^{n} rk_i, g^r) = e(g_A^{-s \cdot x_A} \cdot H_2^t(x), g^r),$$

and then decrypts it with his private key.

With this multi-proxy scheme, an attacker needs to compromise multiple cloud platforms in different cloud providers, which significantly increases the attack time and computation cost.

3. PERFORMANCE EVALUATION

We implement CL-PRE 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. Our experiments are carried out on one-core,1GB memory Linux virtual machine residing on a PC with Intel i5 3.4GHz processor and 4GB memory. We accelerate the re-encryption process by caching interim results of pairing computation, which makes each of the re-encryption 20% faster.

Our primitive 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. Our result also confirms that CL-PRE achieves a much lower computing overhead at data owner side by shifting the computation to cloud.

4. 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 *ASIACRYPT*, 2003.

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

[4] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, 2001.

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