Clock-Based Proxy Re-encryption Scheme in Unreliable Clouds

Qin Liu[†‡], Guojun Wang[†], and Jie Wu[‡],
[†]School of Information Science and Engineering, Central South University, P. R. China
[‡]Department of Computer and Information Sciences, Temple University, USA
Email:{qin.liu, jiewu}@temple.edu, csgjwang@mail.csu.edu.cn

***Abstract*—In this paper, we propose a clock-based proxy re-encryption (C-PRE) scheme to achieve fine-grained access control and scalable user revocation in unreliable clouds. Our scheme, which is built on top of ciphertext-policy attribute-based encryption (CP-ABE) and proxy re-encryption (PRE), allows the data owner and the cloud to share a secret key in advance, with which the cloud can be delegated to re-encrypt data on behalf of the data owner. The main merit of our scheme is that the cloud can automatically re-encrypt data based on its internal clock without receiving any command from the data owner.**

***Index Terms*—Cloud computing; clock; proxy re-encryption; ciphertext-policy attribute-based encryption**

I. Introduction

Cloud computing has increasingly become a technology trend due to its key properties, such as cost saving and on-demand provisioning. However, many users still hesitate to move data into a cloud, since they worry about their sensitive information being leaked by a cloud service provider [1]. A fundamental approach for secure data sharing in the cloud is to let the *data owner* store encrypted data, and disclose the decryption keys to authorized *users*. There are two problems related to this approach: *access control on encrypted data* and *user revocation*.

Access control means that the encrypted data can only be decrypted by users with permissions. Ciphertext-policy attribute-based encryption (CP-ABE) [2] is a new cryptographic technique having such a property, where the users are identified by a set of *attributes* rather than an exact identity. Each data is encrypted with an *attribute-based access structure*, such that only the users whose attributes satisfy the access structure can decrypt the data. For example, data encrypted with the access structure $\{(\alpha_1 \wedge \alpha_2) \vee \alpha_3\}$ can be decrypted by either users with attributes α_1 and α_2, or users with attribute α_3. This flexibility makes CP-ABE an attractive choice when selecting an encryption scheme for cloud computing.

User revocation means that the data owner withdraws access rights from users who are no longer entitled to access the data. Existing work requires the data owner to re-encrypt the data and re-distribute new keys to the remaining authorized users, so that the revoked users cannot decrypt the data using their expired keys, but authorized users can successfully access the data. However, this kind of solutions may cause a performance bottleneck on the data owner when there are frequent user revocations.

To reduce the heavy workload on the data owner, our previous work [3] proposed a hierarchical attribute-based encryption (HABE) scheme by combining the proxy re-encryption (PRE) technique [4] with the CP-ABE system. In the HABE scheme, the data owner sends a re-encryption command including PRE keys to the cloud, which can be delegated to re-encrypt the data with these PRE keys, without knowing data contents.

However, the HABE scheme does not consider the underlying system architecture of the cloud environment. First, a cloud is essentially a large scale distributed system, which will experience failures common to such systems, such as server crashes and network outages. Second, the data owner's data is replicated over multiple cloud servers for high availability. As a result, a re-encryption command sent by the data owner may not arrive at all cloud servers timely due to the unreliable communications between servers, thus creating security risks.

In this paper, we propose a clock-based proxy re-encryption (C-PRE) scheme. Our contributions are threefold: (1) It is reliable. The C-PRE scheme allows the data owner and the cloud to share a secret key in advance, with which the cloud can automatically re-encrypt data based on their internal clock. (2) Practical. We apply the PRE technique to reduce the heavy workload on the data owner. The C-PRE scheme is more applicable to the environment where there are frequent user revocations. (3) Efficient. We incorporate time concept to the encryption scheme, so that the user with a small number of secret keys can rapidly recover data.

II. Overview of the C-PRE Scheme

In the C-PRE scheme, we assume that *time* is classified into a *time tree*. The height of the time tree can be changed as required. For ease of presentation, in this paper we only consider a tree-layer time tree as shown in Fig. 1, where time is accurate to day, and the time tree is classified into three layers in order: year, month, and day. We use (y, m, d), (y, m), and (y) to denote a particular day, month, and year, respectively. For example, $(2011, 4, 5)$ denotes April 5, 2011.

The intuition of the C-PRE scheme is to allow the data owner and the cloud to share a secret key s in advance, so that the cloud can use s to calculate the PRE keys based on its internal clock, and re-encrypts the ciphertext with these PRE keys. We use s_a^y, $s_a^{y,m}$, and $s_a^{y,m,d}$ to denote the PRE keys on attribute a in time (y), (y, m), and (y, m, d), respectively.

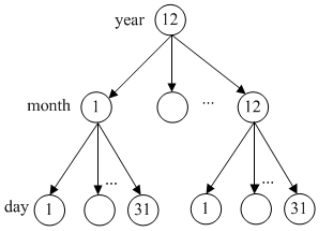

Fig. 1. Sample time tree.

For each attribute a, the data owner calculates the PRE keys in a hierarchical way: $s_a = H_s(PK_a)$, $s_a^y = H_{s_a}(y)$, $s_a^{y,m} = H_{s_a^y}(m)$, and $s_a^{y,m,d} = H_{s_a^{y,m}}(d)$, where PK_a is attribute a's public key; y, m, and d denote a specific year, month, and day, respectively; and $H_s, H_{s_a}, H_{s_a^y}$, and $H_{s_a^{y,m}}$, are hash functions with indexes s, s_a, s_a^y, and $s_a^{y,m}$, respectively.

In the C-PRE scheme, each user is associated with an *attribute set* and an *eligible time*, where the eligible time means how long the user can access the data. Each data is associated with an *attribute-based access structure* and *an access time*. On receiving a file access request, the cloud first determines current time based on its internal clock, say (y, m, d). Then, it uses the shared key s to calculate the PRE keys in time (y), (y, m), and (y, m, d) for all attributes in the access structure, and then uses these PRE keys to re-encrypt the original ciphertext, so that the data can be decrypted by only the users whose attribute set satisfies the access structure, and whose eligible time satisfies the access time.

III. Construction

As in [5], a randomized algorithm $\mathcal{IG}$ is a bilinear Diffie-Hellman (BDH) parameter generator if $\mathcal{IG}$ takes a sufficiently large security parameter K as input, runs in polynomial time in K, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q and the description of a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e})$ be the outputs of $\mathcal{IG}$ under K, and P_0 be the random generator of $\mathbb{G}_1$.

The C-PRE scheme contains following five algorithms:

1. $Setup(K) \to (PK, MK, s)$: The data owner takes security parameter K as input to generate the system public key PK, the system master key MK, and the secret shared key s. Specifically, he first defines the universe attributes $\mathbb{UA}$. Then, for each attribute a in $\mathbb{UA}$, he generates a public/private key pair (sk_a, PK_a), where sk_a is randomly chosen from $\mathbb{Z}_q$ and $PK_a = sk_a P_0$. Next, he computes $Q_0 = mk_0 P_0$ and $SK_1 = mk_0 P_1$, where mk_0 is randomly chosen from $\mathbb{Z}_q$ and P_1 is randomly chosen from $\mathbb{G}_1$. Finally, he randomly chooses mk_1 and s from $\mathbb{Z}_q$, publishes $\{PK_a\}_{a\in\mathbb{UA}}$ and $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, P_1, Q_0)$ as system public key PK, keeps $\{sk_a\}_{a\in\mathbb{UA}}$, mk_0, mk_1, and SK_1 as system master key MK, and sends s to the cloud as a shared secret key.

2. $GenKey(PK, MK, s, PK_u, A_u, T_u) \to (SK_u, \{SK_{u,a}^{T_u}\}_{a\in A_u})$: Suppose user $\mathcal{U}$ with public key PK_u possesses an attribute set A_u with eligible time T_u. Then, the data owner first generates a user identity secret key SK_u, and then for each attribute $a \in A_u$, he generates a user attribute secret key $SK_{u,a}^{T_u}$ for $\mathcal{U}$. Specifically, the data owner calculates user master key with $mk_u = H_{mk_1}(PK_u)$, and sets $SK_u = mk_1 mk_u P_0$, where $H_{mk_1} : \mathbb{G}_1 \to \mathbb{Z}_q$. For each attribute $a \in A_u$, he sets $SK_{u,a}^{T_u} = SK_1 + mk_1 mk_u (PK_a + s_a^{T_u} P_0)$, where $s_a^{T_u}$ is the PRE key on attribute a in time T_u. Note that if T_u is a particular year (y), then $s_a^y = H_{s_a}(y)$; if T_u is a particular month (y, m), then $s_a^{y,m} = H_{s_a^y}(m)$; if T_u is a particular day (y, m, d), then $s_a^{y,m,d} = H_{s_a^{y,m}}(d)$, where $s_a = H_s(PK_a)$, $H_s : \mathbb{G}_1 \to \mathbb{Z}_q$, and $H_{s_a}, H_{s_a^y}, H_{s_a^{y,m}} : \{0,1\}^* \to \mathbb{Z}_q$.

3. $Encrypt(PK, \mathbb{A}, D) \to (C_{\mathbb{A}})$: Given an access structure $\mathbb{A} = \bigvee_{i=1}^{N}(CC_i) = \bigvee_{i=1}^{N}(\bigwedge_{j=1}^{n_i} a_{ij})$ that is in the disjunctive normal form (DNF), the data owner encrypts data D as follows: He first picks a random element $r \in \mathbb{Z}_q$, and then sets $n_{\mathbb{A}}$ to be the lowest common multiple (LCM) of $n_1, \ldots, n_N$. Finally, he calculates Eq. (1) to produce the ciphertext:

$$\begin{aligned} &U_0 = rP_0, V = D \cdot \hat{e}(Q_0, rn_{\mathbb{A}} P_1) \\ &\{U_i = r \sum_{a\in CC_i} PK_a\}_{1\le i\le N} \end{aligned} \quad (1)$$

The ciphertext is set to $C_{\mathbb{A}} = (\mathbb{A}, U_0, \{U_i\}_{1\le i\le N}, V)$.

4. $ReEncrypt(C_{\mathbb{A}}, s, t) \to C_{\mathbb{A}}^t$: On receiving the user's request for data D, the cloud first determines current time $t = (y, m, d)$. Then, it randomly chooses $r' \in \mathbb{Z}_q$, and re-encrypts data D with Eq. (2):

$$\begin{aligned} U_0^t &= U_0 + r' P_0, V^t = V \cdot \hat{e}(Q_0, r' n_{\mathbb{A}} P_1) \\ U_{(y)i}^t &= \textstyle\sum_{a\in CC_i}(U_i + r' PK_a + s_a^y U_0^t) \\ U_{(y,m)i}^t &= \textstyle\sum_{a\in CC_i}(U_i + r' PK_a + s_a^{y,m} U_0^t) \\ U_{(y,m,d)i}^t &= \textstyle\sum_{a\in CC_i}(U_i + r' PK_a + s_a^{y,m,d} U_0^t) \end{aligned} \quad (2)$$

The ciphertext is $C_{\mathbb{A}}^t = (V^t, \mathbb{A}, t, U_0^t, \{U_{(y)i}^t, U_{(y,m)i}^t, U_{(y,m,d)i}^t\}_{1\le i\le N})$.

5. $Decrypt(PK, C_{\mathbb{A}}^t, SK_u, \{SK_{u,a}^{T_u}\}_{a\subseteq\mathbb{A}, T_u\subseteq t}) \to D$: Given ciphertext $C_{\mathbb{A}}^t$, user $\mathcal{U}$, whose attributes satisfy the access structure $\mathbb{A}$, e.g., possessing all attributes in the *i-th* conjunctive clause CC_i, and the eligible time T_u satisfies t, computes Eq. (3) to recover D:

$$V^t / \left(\frac{\hat{e}(U_0^t, \frac{n_{\mathbb{A}}}{n_i} \sum_{a\in CC_i} SK_{u,a}^{T_u})}{\hat{e}(SK_u, \frac{n_{\mathbb{A}}}{n_i} U_{(T_u)i}^t)}\right) \quad (3)$$

References

[1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, and I. Stoica, "A view of cloud computing," *Communications of the ACM*, vol. 53, pp. 50–58, 2010.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, 2007, pp. 321–334.

[3] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (Poster)*, 2010, pp. 735–737.

[4] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," *Advances in Cryptology–EUROCRYPT*, vol. 1403, pp. 127–144, 1998.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology–CRYPTO 2001*, vol. 2139, 2001, pp. 213–229.