Hierarchical Attribute-Based Encryption for Fine-Grained Access Control in Cloud Storage Services

Guojun Wang, Qin Liu
School of Information Science and Engineering
Central South University
Changsha, Hunan Province, P. R. China, 410083
csgjwang@mail.csu.edu.cn

Jie Wu
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA 19122, USA
jiewu@temple.edu

ABSTRACT

Cloud computing, as an emerging computing paradigm, enables users to remotely store their data into a cloud so as to enjoy scalable services on-demand. Especially for small and medium-sized enterprises with limited budgets, they can achieve cost savings and productivity enhancements by using cloud-based services to manage projects, to make collaborations, and the like. However, allowing cloud service providers (CSPs), which are not in the same trusted domains as enterprise users, to take care of confidential data, may raise potential security and privacy issues. To keep the sensitive user data confidential against untrusted CSPs, a natural way is to apply cryptographic approaches, by disclosing decryption keys only to authorized users. However, when enterprise users outsource confidential data for sharing on cloud servers, the adopted encryption system should not only support fine-grained access control, but also provide high performance, full delegation, and scalability, so as to best serve the needs of accessing data anytime and anywhere, delegating within enterprises, and achieving a dynamic set of users. In this paper, we propose a scheme to help enterprises to efficiently share confidential data on cloud servers. We achieve this goal by first combining the hierarchical identity-based encryption (HIBE) system and the ciphertext-policy attribute-based encryption (CP-ABE) system, and then making a performance-expressivity tradeoff, finally applying proxy re-encryption and lazy re-encryption to our scheme.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Security, Algorithms, Design

Keywords

cloud computing, hierarchical attribute-based encryption, fine-grained access control, scalability

1. INTRODUCTION

With the emergence of sharing confidential corporate data on cloud servers, it is imperative to adopt an efficient encryption system with a fine-grained access control to encrypt outsourced data. Ciphertext-policy attribute-based encryption (CP-ABE), as one of the most promising encryption systems in this field, allows the encryption of data by specifying an access control policy over attributes, so that only users with a set of attributes satisfying this policy can decrypt the corresponding data. However, a CP-ABE system may not work well when enterprise users outsource their data for sharing on cloud servers, due to the following reasons:

First, one of the biggest merits of cloud computing is that users can access data stored in the cloud anytime and anywhere using any device, such as thin clients with limited bandwidth, CPU, and memory capabilities. Therefore, the encryption system should provide high performance.

Second, in the case of a large-scale industry, a delegation mechanism in the generation of keys inside an enterprise is needed. Although some CP-ABE schemes support delegation between users, which enables a user to generate attribute secret keys containing a subset of his own attribute secret keys for other users, we hope to achieve a full delegation, that is, a delegation mechanism between attribute authorities (AAs), which independently make decisions on the structure and semantics of their attributes.

Third, in case of a large-scale industry with a high turnover rate, a scalable revocation mechanism is a must. The existing CP-ABE schemes usually demand users to heavily depend on AAs and maintain a large amount of secret keys storage, which lacks flexibility and scalability.

Motivation. Our main design goal is to help the enterprise users to efficiently share confidential data on cloud servers. Specifically, we want to make our scheme more applicable in cloud computing by simultaneously achieving fine-grained access control, high performance, practicability, and scalability.

Our Contribution. In this paper, we first propose a hierarchical attribute-based encryption (HABE) model by combining a HIBE system and a CP-ABE system, to provide fine-grained access control and full delegation. Based on the HABE model, we construct a HABE scheme by making a performance-expressivity tradeoff, to achieve high performance. Finally, we propose a scalable revocation scheme by delegating to the CSP most of the computing tasks in revocation, to achieve a dynamic set of users efficiently.

2. THE HABE MODEL

The HABE model (see Figure 1) consists of a root master (RM) that corresponds to the third trusted party (TTP), multiple domain masters (DMs) in which the top-level DMs correspond to multiple enterprise users, and numerous users that correspond to all personnel in an enterprise.

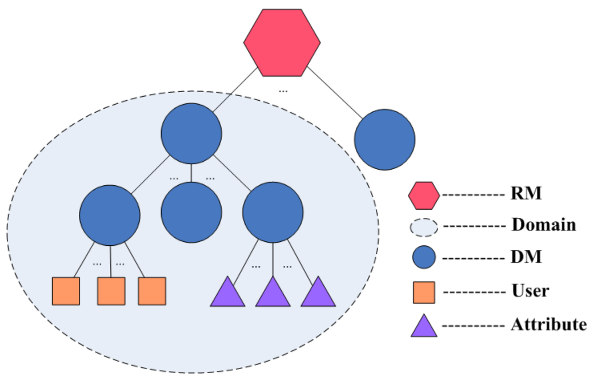

Figure 1: A three-level HABE model

The RM, whose role closely follows the root private key generator (PKG) in a HIBE system, is responsible for the generation and distribution of system parameters and domain keys. The DM, whose role integrates both the properties of the domain PKG in a HIBE system and AA in a CP-ABE system, is responsible for delegating keys to DMs at the next level and distributing keys to users. Specifically, we enable the leftmost DM at the second level to administer all the users in a domain, just as the personnel office administers all personnel in an enterprise, and not to administer any attribute. Notice that other DMs administer an arbitrary number of disjoint attributes, and have full control over the structure and semantics of their attributes.

In the HABE model, we first mark each DM and attribute with a unique identifier (ID), but mark each user with both an ID and a set of descriptive attributes. Then, as Gentry et al [1], we enable an entity's secret key to be extracted from the DM administering itself, and an entity's public key, which denotes its position in the HABE model, to be an ID-tuple consisting of the public key of the DM administering itself and its ID, e.g., the public key of DM_i with ID_i is in the form of $(\mathrm{PK}_{i-1}, \mathrm{ID}_i)$, the public key of user $\mathcal{U}$ with ID_u is in the form of $(\mathrm{PK}_\diamond, \mathrm{ID}_u)$, and the public key of attribute a with ID_a is in the form of $(\mathrm{PK}_i, \mathrm{ID}_a)$, where PK_{i-1}, $\mathrm{PK}_\diamond$, and PK_i are assumed to be the public keys of the DMs that administer DM_i, $\mathcal{U}$, and a, respectively.

3. CONSTRUCTION

Based on the proposed HABE model, we construct the HABE scheme using the bilinear map [1]. As Muller et al [2] sacrificing the expressivity of access structure to achieve better performance, we also use disjunctive normal form (DNF) policy. We assume that all attributes in one conjunctive clause are administered by the same DM. The HABE scheme consists of the following five algorithms:

$Setup(K) \to (params, \mathrm{MK}_0)$: The RM first picks $mk_0 \in \mathbb{Z}_q$, and then chooses groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, two random oracles H_1: $\{0,1\}^* \to \mathbb{G}_1$ and H_2: $\mathbb{G}_2 \to \{0,1\}^n$ for some n, and a random generator $P_0 \in \mathbb{G}_1$. Let $Q_0 = mk_0 P_0 \in \mathbb{G}_1$. The system parameters $params = (q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, n, P_0, Q_0, H_1, H_2)$ will be publicly available, while $\mathrm{MK}_0 = (mk_0)$ will be kept secret.

$CreateDM(params, \mathrm{MK}_i, \mathrm{PK}_{i+1}) \to (\mathrm{MK}_{i+1})$: To generate the master key for DM_{i+1} with PK_{i+1}, the RM or DM_i first picks a random element $mk_{i+1} \in \mathbb{Z}_q$, and then computes $\mathrm{SK}_{i+1} = \mathrm{SK}_i + mk_i P_{i+1}$ where $P_{i+1} = H_1(\mathrm{PK}_{i+1}) \in \mathbb{G}_1$, and $Q_{i+1} = mk_{i+1} P_0 \in \mathbb{G}_1$, finally sets $\mathrm{MK}_{i+1} = (mk_{i+1}, \mathrm{SK}_{i+1}, \text{Q-tuple}_{i+1})$ where $\text{Q-tuple}_{i+1} = (\text{Q-tuple}_i, Q_{i+1})$, and gives the random oracle $H_A\colon \{0,1\} \to \mathbb{Z}_q$ that is chosen by the RM and shared in a domain. Here, we assume that SK_0 is an identity element of $\mathbb{G}_1$, and $\text{Q-tuple}_0 = (Q_0)$.

$CreateUser(params, \mathrm{MK}_i, \mathrm{PK}_u, \mathrm{PK}_a) \to (\mathrm{SK}_{i,u}, \mathrm{SK}_{i,u,a})$: To generate a secret key for user $\mathcal{U}$ with PK_u on attribute a with PK_a, DM_i first checks whether $\mathcal{U}$ is eligible for a, and a is administered by itself. If so, it first sets $mk_u = H_A(\mathrm{PK}_u) \in \mathbb{Z}_q$, $\mathrm{SK}_{i,u} = mk_i mk_u P_0 \in \mathbb{G}_1$, and $\mathrm{SK}_{i,u,a} = \mathrm{SK}_i + mk_i mk_u P_a \in \mathbb{G}_1$, where $P_a = H_1(\mathrm{PK}_a) \in \mathbb{G}_1$, and then gives Q-tuple_i. Otherwise, it outputs "NULL".

$Encrypt(params, \mathbb{A}, \{\mathrm{PK}_{a_{ij}} | 1 \le i \le N, 1 \le j \le n_i\}, f) \to (CT)$: Given a DNF access control policy $\mathbb{A} = \bigvee_{i=1}^{N} (CC_i) = \bigvee_{i=1}^{N} (\bigwedge_{j=1}^{n_i} a_{ij})$, where $N \in \mathbb{Z}^+$ is the number of conjunctive clause in $\mathbb{A}$, $n_i \in \mathbb{Z}^+$ is the number of attributes in the *i-th* conjunctive clause CC_i, and a_{ij} is the *j-th* attribute in CC_i. Let DM_{it_i} with $(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i})$ be the DM at level t_i, administering all attributes in CC_i, where ID_{ik} for $1 \le k < t_i$ are IDs of DM_{it_i}'s ancestors. The sender:

1. For $1 \le i \le N$: Computes $P_{ij} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{ij}) \in \mathbb{G}_1$ for $1 \le j \le t_i$, and $P_{a_{ij}} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i}, \mathrm{ID}_{a_{ij}}) \in \mathbb{G}_1$ for $1 \le j \le n_i$.

2. Picks a random element $r \in \mathbb{Z}_q$, sets $n_\mathbb{A}$ to be the lowest common multiple (LCM) of $n_1, \ldots, n_N$, and computes $U_0 = rP_0$, $U_{12} = rP_{12}$, $\ldots$, $U_{1t_1} = rP_{1t_1}$, $U_1 = r\sum_{j=1}^{n_1} P_{a_{1j}}$, $\ldots$, $U_{N2} = rP_{N2}$, $\ldots$, $U_{Nt_N} = rP_{Nt_N}$, $U_N = r\sum_{j=1}^{n_N} P_{a_{Nj}}$, and $V = f \oplus H_2(\hat{e}(Q_0, rn_\mathbb{A} P_1))$. The ciphertext is $CT = (\mathbb{A}, C_f)$, where $C_f = [U_0, U_{12}, \ldots, U_{1t_1}, U_1, \ldots, U_{N2}, \ldots, U_{Nt_N}, U_N, V]$.

$Decrypt(params, CT, \mathrm{SK}_{it_i,u}, \{\mathrm{SK}_{it_i,u,a_{ij}} | 1 \le j \le n_i\}, \text{Q-tuple}_{i(t_i-1)}) \to (f)$: User $\mathcal{U}$, whose attributes satisfy CC_i, computes $V \oplus H_2(\frac{\hat{e}(U_0, \frac{n_\mathbb{A}}{n_i}\sum_{j=1}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}})}{\hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_\mathbb{A}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_\mathbb{A} Q_{i(j-1)})})$ to recover f. Observe that:

$$
\begin{aligned}
& V \oplus H_2(\frac{\hat{e}(U_0, \frac{n_\mathbb{A}}{n_i}\sum_{j=1}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}})}{\hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_\mathbb{A}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_\mathbb{A} Q_{i(j-1)})}) \\
&= V \oplus H_2(\frac{\hat{e}(Q_0, n_\mathbb{A} rP_1) \prod_{k=2}^{t_i} \hat{e}(Q_{i(k-1)}, n_\mathbb{A} U_{ik}) \hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_\mathbb{A}}{n_i} U_i)}{\hat{e}(\mathrm{SK}_{it_i,u}, \frac{n_\mathbb{A}}{n_i} U_i) \prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_\mathbb{A} Q_{i(j-1)})}) \\
&= V \oplus H_2(\hat{e}(Q_0, n_\mathbb{A} rP_1)) \qquad \text{as required.}
\end{aligned}
$$

Remark. To achieve better performance, we enable user $\mathcal{U}$ to send the value of $\text{Q-tuple}_{i(t_i-1)}$ to the CSP before decrypting data, so that the CSP can help to calculate the value of $\prod_{j=2}^{t_i} \hat{e}(U_{ij}, n_\mathbb{A} Q_{i(j-1)})$). Given this value, $\mathcal{U}$ executes the bilinear map operations for two times to recover the file.

Performance. The HABE scheme eliminates the on-line inquiry for authenticated attribute public keys. When there is only the first level DM to administer all attributes, a user needs to execute one bilinear map and $O(N)$ number of point multiplication operations to output a ciphertext of $O(N)$ length, and $O(1)$ bilinear map operations to recover a file.

4. SECURITY INTUITION

Recall that a confidential file f is encrypted in the form of $f \oplus H_2(\hat{e}(Q_0, rn_{\mathbb{A}}P_1))$. Therefore, an adversary $\mathcal{A}$ needs to construct $\hat{e}(Q_0, rn_{\mathbb{A}}P_1) = \hat{e}(U_0, \mathrm{SK}_1)^{n_{\mathbb{A}}}$ to decrypt C_f. $\mathcal{A}$ can request any user key of his choice other than possessing a sufficient set of attribute keys to decrypt C_f.

For ease of presentation, we have the following assumptions: $\mathcal{A}$ has requested attribute secret keys for user $\mathcal{U}$ on all but one of the attributes $a_{i1}, \ldots, a_{i(k-1)}, a_{i(k+1)}, \ldots, a_{in_i}$ in CC_i, and for user $\mathcal{U}^{'}$ on the missing attribute a_{ik}.

The only occurrence of SK_1 is in the user attribute secret key, therefore adversary $\mathcal{A}$ has to use the attribute secret keys requested for $\mathcal{U}$ and $\mathcal{U}^{'}$ for the bilinear map, yielding:

$$\begin{aligned} &\hat{e}(U_0, \tfrac{n_{\mathbb{A}}}{n_i} \sum_{j=1, j\neq k}^{n_i} \mathrm{SK}_{it_i,u,a_{ij}} + \tfrac{n_{\mathbb{A}}}{n_i} \mathrm{SK}_{it_i,u',a_{ik}} + \alpha) \\ &= \hat{e}(U_0, \mathrm{SK}_1)^{n_{\mathbb{A}}} \prod_{t=2}^{t_i} \hat{e}(Q_{i(t-1)}, U_{it})^{n_{\mathbb{A}}} \hat{e}(rP_0, \alpha) \\ &\hat{e}(mk_{u'} mk_{it_i} P_0, rP_{a_{ik}})^{\frac{n_{\mathbb{A}}}{n_i}} \hat{e}(mk_u mk_{it_i} P_0, r \sum_{j=1, j\neq k}^{n_i} P_{a_{ij}})^{\frac{n_{\mathbb{A}}}{n_i}} \end{aligned}$$

for some α. To obtain $\hat{e}(U_0, \mathrm{SK}_1)^{n_{\mathbb{A}}}$, the last four elements have to be eliminated. However, the values of the last two elements are unknown to the adversary, and cannot be constructed. Therefore, $\mathcal{A}$ cannot recover the file.

5. REVOCATION

Inspired by Yu et al [3], we make slight alterations into our HABE scheme, and apply proxy re-encryption and lazy re-encryption into our scheme.

Modifications in keys: We enable each attribute a with ID_a to be bound to a version number, which increases by one whenever a user associated with a is revoked. Therefore, an attribute public key is the form of PK_a^t=(v_a^t, PK_i, ID_a), where $t \in \mathbb{Z}_q$ is the version number of the attribute public key, and $v_a^t \in \{0,1\}^*$ is a string corresponding to t.

Modifications in algorithms: First, we enable the *CreateDM* algorithm to uniformly and randomly generate a hash function $H_{mk_i} : \{0,1\}^* \rightarrow \mathbb{Z}_q$ for DM_i, where H_{mk_i} is a random oracle. Second, we construct another algorithm *CreateAttribute*(PK_a^t, mk_i), which is executed by DM_i whenever it receives a request for P_a^t, and outputs $H_{mk_i}(\mathrm{PK}_a^t)P_0 \in \mathbb{G}_1$. Therefore, the first step in the *Encrypt* algorithm turns into requesting P-values of all attributes in $\mathbb{A}$ from the DMs.

When a user is revoked, denoted $\mathcal{V}$, it is imperative to update public keys of attributes in $\mathrm{S}_{\mathcal{V}}$, and attribute secret keys for remaining users who possess at least one attribute in $\mathrm{S}_{\mathcal{V}}$, and re-encrypt data whose access structure specifies at least one attribute in $\mathrm{S}_{\mathcal{V}}$, where the set $\mathrm{S}_{\mathcal{V}}$ contains all attributes associated with $\mathcal{V}$. If all these tasks are performed by the DMs themselves, it would introduce a heavy computing overhead and may also require the DMs to always be online. Therefore, we get the idea to take advantage of the abundant resources in a cloud by delegating to CSPs most of the computing tasks in revocation.

The main process is as follows: First, for each attribute a in $\mathrm{S}_{\mathcal{V}}$, we enable the DM to update PK_a^t with PK_a^{t+1} by adding each version number to 1, and compute the the PRE key with $\mathrm{Pkey}_a^{t+1} = H_{mk_i}(\mathrm{PK}_a^{t+1}) - H_{mk_i}(\mathrm{PK}_a^t)$. After sending the update messages to the CSP, the DMs can go off-line. Second, for every attribute in $\mathrm{S}_{\mathcal{V}}$, the CSP stores a new attribute public key and PRE key received in a proper position in an attribute history list (AHL). Whenever an access request is received from a user, denoted $\mathcal{U}$, it first checks whether all attributes in the data access structure are the latest ones. If so, it returns the data directly. Otherwise, it re-encrypts the data, and tells the user to update his overdue attribute secret keys, using related PRE keys, as follows:

Data Re-encryption. Suppose $CT^{'} = (\mathbb{A}^{'}, [U_0^{'}, U_{12}^{'}, \ldots, U_{1t_1}^{'}, U_1^{'}, \ldots, U_{N2}^{'}, \ldots, U_{Nt_N}^{'}, U_N^{'}, V^{'}])$ is the original ciphertext. The CSP inquires the AHL and re-encrypts $CT^{'}$ to CT by setting $V = V^{'}$, $U_0 = U_0^{'}$, and $U_i = U_i^{'} + \sum_{a \in \mathrm{S}_{\mathcal{V}} \wedge CC_i^{'}}$ $(\mathrm{Pkey}_a^{t^{'}+1} + \ldots + \mathrm{Pkey}_a^t)U_0$, for $1 \leq i \leq N$, where $t^{'}$ and t are the version numbers of a's public key in $\mathbb{A}^{'}$ and the latest version number of a's public key in AHL, respectively.

Key Update. For every attribute a in $S_{\mathcal{U}}$, which is a set of overdue attributes associated with user $\mathcal{U}$, $\mathcal{U}$ updates corresponding attribute secret key $\mathrm{SK}_{i,u,a}^{t^{'}}$ to $\mathrm{SK}_{i,u,a}^t$ by setting $\mathrm{SK}_{i,u,a}^t = \mathrm{SK}_{i,u,a}^{t^{'}} + \mathrm{Pkey}_a \mathrm{SK}_{i,u}$, in which Pkey_a, taken from the CSP, is set to be $\mathrm{Pkey}_a^{t^{'}+1} + \ldots + \mathrm{Pkey}_a^t = H_{mk_i}(\mathrm{PK}_a^t) - H_{mk_i}(\mathrm{PK}_a^{t^{'}})$, where t and $t^{'}$ are the version numbers of a's public key for $\mathcal{U}$ and the latest version number of a's public key in AHL, respectively.

6. CONCLUSION AND FUTURE WORK

In this paper, we construct a scheme, which has several traits: (1) high performance; (2) fine-grained access control; (3) scalability; (4) full delegation. Our HABE scheme, which is also collusion resistant, can be proven to be semantically secure against adaptive chosen plaintext attacks under the BDH assumption and the random oracle model [1].

In future work, we will work towards designing a more expressive scheme, which can be proved to have full security under the standard model, with better performance.

7. ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under Grant Nos. 90718034 & 60773013, the Program for New Century Excellent Talents in University under Grant No. NCET-06-0686, and the Program for Changjiang Scholars and Innovative Research Team in University under Grant No. IRT0661.

8. REFERENCES

[1] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. In *Proceedings of ASIACRYPT 2002*, pages 548-566.

[2] S. Muller, S. Katzenbeisser, and C. Eckert. Distributed Attribute-Based Encryption. In *Proceedings of ICISC 2008*, pages 20-36.

[3] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. In *Proceedings of IEEE INFOCOM 2010*, pages 534-542.