An ID-based Hierarchical Access Control Scheme with Constant Size Public Parameter

Rang Zhou, Chunxiang Xu, Wanpeng Li, Jining Zhao
(Corresponding author: Rang Zhou)

Department of Computer Science and Engineering, University of Electronic Science and Technology of China
No.2006, Xiyuan Avenue, West Hi-tech Zone, Chengdu, 611731, P.R. China
(Email: zhour1987@sohu.com)

(Received Mar. 26, 2013; revised and accepted Jan. 15 & July 13, 2015)

Abstract

Many ways are proposed to reduce the secret storage space of access control in hierarchy, but no one optimizes the public parameters which are only modified by CA. Length of each public parameter is one important factor for the size and utilization of storage space. The frequent changes on the maximum length of public parameter will be a weakness for stability, in dynamic key management. Number and length of changed public parameter are considered for the interaction between CA and trusted public platform in dynamic management. The paper proposes an improved scheme to optimize storage space of public parameter for each class from variable linear size to a small constant size. Our scheme has higher utilization, stability and efficiency on storage space of trusted public platform and needs less interaction between CA and trusted public platform. The security of this improved scheme is proved on key recovery model.

Keywords: Hierarchical access control, ID-based cryptographic, key management, public parameters

1 Introduction

The technology of access control, which is one of the most important components in computer system, is a central issue in computer research. With the great development of information technology, to improve the efficiency of access control and protect the data confidentiality, security access control technology becomes the main research objective recently [5].

To achieve the goal of classifying users and information resource, researchers proposed multi-level security access control model [14]. Each two users are linked by the relationship of binary partial order. The users in high security layer can access the secretly information possessed by low layer users. It is infeasible on the contrary.

To reduce the key storage, Akl and Taylor [1] firstly proposed an approach that the public key cryptography is used on multi-level security access control, in 1983. The approach optimizes key management greatly, which separates key assignment into a public key cryptosystem for the management of all classes' privilege and a symmetric cryptosystem for data protection.

The actual scene for hierarchical access control: In the past, hierarchical access control schemes always are assumed that trusted public platform [4, 8] is a simple storage space only for users' corruption and free for whole system. In fact, we have to admit that the trusted public platform is provided by money. CA buys the trusted public platform to store lots of public parameters, so the money for the management of public parameter is related with the whole system space for public parameter. So, it is necessary to reduce the scale of public parameters of the whole hierarchical access control scheme. Further, we can consider that the trusted public platform has the computation ability to reduce the computation which is done by CA and users as the model of cloud computing.

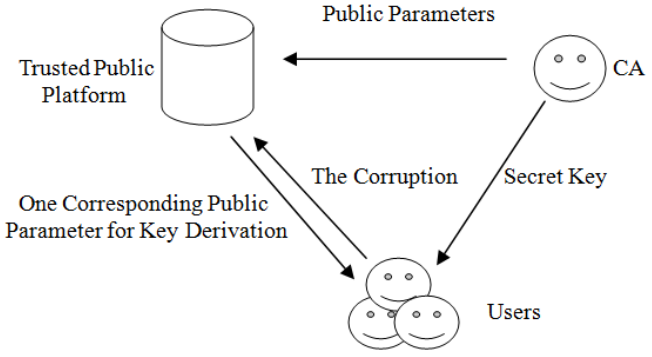

Figure 1: An example of using the proposed scheme

Now, we consider the scene where the whole system is constructed by CA, trusted public platform and users as Figure 1. The trusted platform can provide the computation and storage service [25]. It is necessary to assume that the trusted platform for public parameter is as

honest and curious as the cloud service provider in the model of cloud computing. When a high layer user wants to derive the low layer key, the trusted platform sends only one corresponding public parameter which is used for key derivation simply to the authorized user. Then, the authorized user computes the low layer key by using its secret key and the corresponding public parameter.

Related work: In Akl-Taylor's scheme [1], a CA is needed and if two security classes have a partial order relationship, the low security class's key can be derived from the high security class's key. Though the scheme is simple to understand, but two problems are inevitable in application. The first problem is the storage for the great amount of integers and the other is the computation load that all system needs to update in every change of authorized relationship. To reduce the storage of integers, Mackinnon [21] proposed a typical and improved key allocation scheme. Unfortunately, a lot of integers are needed in this scheme.

Harn and Lin [12] proposed a hierarchical encryption access control scheme based on the hardness of large integer factorization. In contrast with Akl-Taylor's scheme, Harn-Lin's scheme generates key from the low layer to high layer. Though the scheme reduces the time of public parameters generation, the number of integers is same to Akl-Taylor's scheme.

Combining with the security assumptions of Akl-Taylor's scheme and Harn-Lin's scheme, Hwang-Yang [15] proposed an efficient hierarchical access control scheme. The scheme effectively reduces the number of integers, but it is not secure under collusion attack [28].

A YCN scheme are proposed to solve the key assignment problem with a matrix model by Yeh et al. [30]. To protect the data security, in their scheme, two key are used which are a derivation key and an encrypted data key. However, Hwang shown that several user classes can collaborate to derive the derivation keys and encryption keys in some cases under YCN scheme [13, 17]. In 2003, to fix the collusive attack problem, Yeh et al. and Lin proposed their improved schemes, respectively [17, 31].

To solve the problem of key security, Tzeng proposed a time-bound cryptographic key assignment scheme in a partially ordered hierarchy in 2002 [27]. Each authorized user can access the specified data by the legal key during the authorized period only. However, Yi and Ye point out the insecurity of Tzeng's scheme, that the secret keys of some classes can be derived by any three users' collaboration [32]. To solve the problem, Chien and Santis proposed their schemes about time-bound cryptographic key assignment scheme in a partially ordered hierarchy [6, 24].

Lo [20] proposed a new efficient hybrid key assignment scheme in 2011. The security of Lo's scheme is based on the hardness of one-way function and large integer factorization. The scheme is proved more efficiently than the previous schemes with same type. However, two efficiency problems of Lo's scheme must be considered on the actual scene. They are described as following: 1). To resist the partial key exposure attack, every e_i satisfies the condition $m^{7/8} \leq e_i \leq m$ [3]. The public parameters of high layer class are constructed by the e_i's product of all child nodes. Obviously, the length of public parameter from low layer class to high layer class gradually increases. So, the public parameter of high class will be large. It is bad for the cost on buying the space of public platform. 2). In the dynamic key management phase, every modification about class adding or deleting corresponds to a great mount of modification on public parameters.

Nikooghadam [22], Wu [29], and Odelu [23] proposed their schemes which are constructed on ECC. Lo [19] and Lee [16] proposed their schemes Based on Polynomial. These schemes have high efficiency on key derivation and key management, but each authorized relationship needs a public parameter to be stored on the public trusted platform. The amount of public parameters in these schemes largely surpasses the number of the secure classes. These schemes not only don't take full advantage of hierarchy access control, but increase the cost of public parameters storage and maintenance.

Many other schemes have been proposed to solve the problem of access control in a hierarchy [9, 10, 11, 26]. However, two weaknesses are inevitably in these independent key management schemes. The first one is path searching from starting class to target class. The other is the information interaction [33], which contains all public parameters on the path, between the public trusted platform and users.

Our contribution: The contributions of this paper can be summarized as follows:

1) Firstly, to resolve the two problems proposed in Lo's scheme, the paper proposes an improved scheme with constant size of public parameters. The improved scheme optimizes every public parameter, which stores on trusted public platform, to constant size. So, comparing with Lo's scheme, the improved scheme has the advantage as following: To reduce the storage of public parameters greatly; to improve the storage utilization of public parameters greatly; to decrease the number of modification on public parameters in dynamic key management; to strengthen the system stability. At the same time, the optimization of public parameters makes the key derivation process more efficiently than Lo's scheme.

2) Combining with Lo's scheme, the paper introduces the security classes' ID to a part of public key and proposes an ID-based hierarchical access control scheme. By changing every public parameter to a simple random value in constant size, the scheme reduces the size of storage for public parameter smaller. As a result, the scheme optimizes the storage space and utilization of public parameter further.

Roadmap: The remainder of the paper is structured as follows. In Section 2, the preliminaries for security analysis are introduced. Section 3 presents the new improved scheme including key generation, key derivation

and dynamic key management. The performance comparing with the original scheme is provided in Section 4. The security proof of the new improved scheme is discussed in Section 5. Finally, Section 6 concludes the paper.

2 Preliminaries

2.1 The Key Recovery Attack Model for Hierarchical Access Control Scheme

Definition 1. *(Key Recovery Model) [2]. A key allocation scheme is secure in key recovery if no polynomial time adversary A has a non-negligible advantage (in the security parameter ρ) against the challenger in the following game:*

- *Setup: The challenger C runs Setup(1^ρ,G), and gives the resulting public information Pub to the adversary A.*
- *Attack: The adversary issues, in any adaptively chosen order, a polynomial number of Corrupt(v_i) queries, which the challenger C answers by retrieving $k_i \leftarrow Sec(v_i)$ and giving k_i to A.*
- *Break: The adversary outputs a node v^*, subject to $v^* \notin Des(v_i, G)$ for any v_i asked in Attack Phase, along with her best guess k'_{v^*} to the cryptographic key k_{v^*} associated with node v^*.*

We define the adversary's advantage in attacking the scheme as: $Adv_A^{REC} = Pr[k'_{v^*} = k_{v^*}]$.

The Corrupt(v_i), which is proposed by adversary A, is equivalent to collusion attack within the system. The union of adversary's keys is same as the collusive behavior which the low security classes do. They have the same object to get the high security classes' keys. So, it is the static security model in hierarchical system.

2.2 Hard Problems and Assumption

Definition 2. Discrete log-problem: *G is a finite cyclic group, whose generator is g and has order of $n = |G|$, where it is easy to compute $g^a = A$ from a and hard to compute $a', 0 \le a' \le n$ which satisfies the condition $g^{a'}$, from A.*

Definition 3. Strongly Hash function:

1) The input of $H(.)$ is no any restriction on length,

2) The output of $H(.)$ is constant length and can resist the birthday attack,

3) It is simple to compute the value of $H(x)$ from the known x, but it is computationally infeasible on the contrary,

4) No one can feasibly finds two different values of x that give the same $H(x)$.

Definition 4. Large integer factorization problem: *For the given odd composite number N constructed by two prime factor, it is hard to compute the prime p which satisfies the condition $p|N$ in appropriate environment.*

Definition 5. RSA problem: *The number $N = p \cdot q$ is known, where the number p and q both are primes. The number e is an integer and satisfies the condition $\gcd(e, (p-1)\cdot(q-1)) = 1$. It is infeasible to compute the unique integer $m \in Z_n^*$, where m satisfies the condition $m^e = c(modN)$ from the fixed $c \in Z_n^*$.*

3 The Proposed Scheme

The paper proposes an improved ID-based hierarchical access control scheme with constant size public parameter. The scheme is comprised of key generation, key derivation and dynamic key management.

3.1 Key Generation Phase

Firstly, a CA is needed to do the work of key computation and assignment for the authorized users. CA executes the following steps:

Step 1. CA chooses two large primes p and q, and computes the public large number $m = p \cdot q$ and secret parameter $\varphi(m) = (p-1)\cdot(q-1)$, where $\varphi(m)$ must be kept secretly by CA and m is kept on the trusted public platform. At last, CA destroys p and q.

Step 2. CA generates a random number g, which is coprime with m and $2 < g < (m-1)$. CA chooses two public one-way hash function $H_1(.)$, $H_2(.)$ and $m^{7/8} \le H_1(.) \le m$, $\gcd(H_1(.), \varphi(m)) = 1$, $H_2(.) \le m$ [3].

Step 3. In the hierarchical access control, each class C_i has an ID_i.

Step 4. For every class C_i, which is a non-leaf class or a leaf class with two or more immediate ancestors in the hierarchy, CA computes $e_i = H_1(ID_i)$. Then, CA computes private key exponent $d_i = e_i^{-1} mod \varphi(m)$. The pair (e_i, d_i) is corresponding to C_i, where the secret key d_i is only kept secretly by CA.

Step 5. Key generation.

- For every class C_i, which is a non-leaf class or a leaf class with two or more immediate ancestors in the hierarchy, CA computes the secret key $K_i = g^{d_i \prod_{all C_l} d_l mod \varphi(m)} \bmod m$, where the all C_l is the successor of C_i and no one is a leaf class with only one immediate ancestor.
- For every leaf class C_i, which has only one immediate ancestor C_j in the hierarchy, CA randomly generates a secret key K_i for the corresponding class C_i and calculates a public parameter $PB_i = K_i \oplus H_2(K_j, ID_i, ID_j)$.

Step 6. CA passes every secret key K_i to corresponding class C_i through a secure channel individually and publishes all public parameters and authorized relationships on public platform of trusted.

For a clear description for the initialization and key generation, an example of the proposed scheme, where the classes have the authorized relationships, is shown in Figure 2. The prime pairs, secret keys and public parameters about dependent key are generated as Table 1. The secret keys and public parameters about independent key are generated as Table 2.

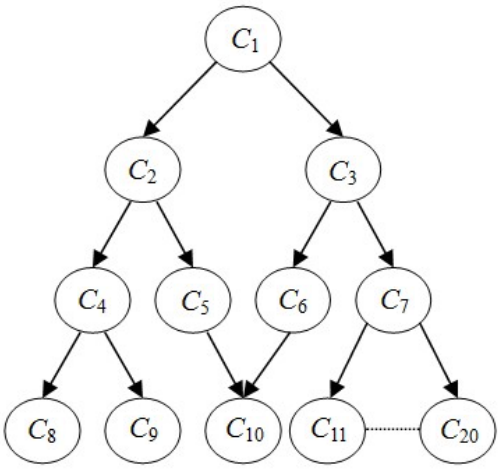

Figure 2: An example of using the proposed scheme

Table 1: The dependent key in the example

Class (C_i)	Secret key (K_i)	Prime pair (e_i, d_i)
C_1	$g^{d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_{10} mod\varphi(m)} \bmod m$	(e_1, d_1)
C_2	$g^{d_2 d_4 d_5 d_{10} mod\varphi(m)} \bmod m$	(e_2, d_2)
C_3	$g^{d_3 d_6 d_7 d_{10} mod\varphi(m)} \bmod m$	(e_3, d_3)
C_4	$g^{d_4 mod\varphi(m)} \bmod m$	(e_4, d_4)
C_5	$g^{d_5 d_{10} mod\varphi(m)} \bmod m$	(e_5, d_5)
C_6	$g^{d_6 d_{10} mod\varphi(m)} \bmod m$	(e_6, d_6)
C_7	$g^{d_7 mod\varphi(m)} \bmod m$	(e_7, d_7)
C_{10}	$g^{d_{10} mod\varphi(m)} \bmod m$	(e_{10}, d_{10})

3.2 Key Derivation Phase

3.2.1 The Formula of Key Derivation

C_i and C_j are in the hierarchy with relationship $C_i \preceq C_j$. Secret key K_i, K_j are corresponding to classes C_i, C_j, respectively. The formula of key derivation from K_j to K_i is used as following:

Table 2: The independent key in the example

Class(C_i)	Public parameter(PB_i)	Secret key(K_i)
C_8	$K_8 \oplus H_2(K_4, ID_8, ID_4)$	K_8
C_9	$K_9 \oplus H_2(K_4, ID_9, ID_4)$	K_9
C_{11}	$K_{11} \oplus H_2(K_7, ID_{11}, ID_7)$	K_{11}
$\cdots$	$\cdots$	$\cdots$
C_{20}	$K_{20} \oplus H_2(K_7, ID_{20}, ID_7)$	K_{20}

1) C_i is a leaf class with only one immediate ancestor C_j, $K_i = PB_i \oplus H_2(K_j, ID_i, ID_j)$.

2) If C_i is a leaf class with only one immediate ancestor C_k, $K_i = PB_i \oplus H_2(K_j^{\prod_{allC_l, C_l \preceq C_j \wedge C_l \npreceq C_k} H_1(ID_l)}, ID_i, ID_k)$.

3) Otherwise, $K_i = K_j^{\prod_{allC_l, C_l \preceq C_j \wedge C_l \npreceq C_i} H_1(ID_l)}$.

3.2.2 Correctness Proof

Theorem 1. *For two classes C_i and C_j with the relationship $C_i \preceq C_j$, C_j can derive the secret key K_i of C_i from the above formula.*

Proof. In a hierarchical access control system with n nodes, we have the public parameters $PB_1, PB_2, ...PB_n$, the one-way hash function $H_1(.)$, $H_2(.)$ and $m^{7/8} \leq H_1(.) \leq m$, $H_1(.) \in prime$, $gcd(H_1(.)\varphi(m)) = 1$, $H_2(.) \leq m$.

1) Class C_i is a leaf node with only one immediate ancestor C_j. It is simple to get the key from the equation $K_i = PB_i \oplus H_2(K_j, ID_i, ID_j)$.

2) Class C_i is a leaf node with only one immediate ancestor C_k and the relationship of $C_i \preceq C_k \preceq C_j$ is satisfied, where C_k is the immediate ancestor of C_i. Because we do not know the secret key K_k, we shall process the following steps. Every C_l is the successor of C_j, except the leaf class with only one immediate ancestor.

$$\begin{aligned} K_i &= PB_i \oplus H_2(K_k, ID_i, ID_k) \\ &= PB_i \oplus H_2(g^{d_k \prod_{allC_l, C_l \preceq C_k} d_l} \bmod m, \\ &\qquad ID_i, ID_k) \\ &= PB_i \oplus H_2(K_j^{(\frac{\prod_{allC_l, C_l \preceq C_j} e_l}{\prod_{allC_l, C_l \preceq C_k} e_l})} \bmod m, \\ &\qquad ID_i, ID_k) \\ &= PB_i \oplus H_2(K_j^{\prod_{allC_l, C_l \preceq C_j \wedge C_l \npreceq C_k} H_1(ID_l)} \bmod m, \\ &\qquad ID_i, ID_k). \end{aligned}$$

3) Class C_i is a non-leaf class or a leaf class with two or

more immediate ancestors in the hierarchy.

$$\begin{aligned} K_i &= g^{d_i \prod_{allC_l, C_l \preceq C_i} d_l} \mod m \\ &= K_j^{(\prod_{allC_l, C_l \preceq C_j} e_l / (\prod_{allC_l, C_l \preceq C_i} e_l))} \mod m \\ &= K_j^{\prod_{allC_l, C_l \preceq C_j \wedge C_l \not\preceq C_i} H_1(ID_l)} \mod m \end{aligned}$$

□

3.2.3 Example of Key Derivation

To describe the key derivation clearly, the example as Figure 2 is used.

1) The key derivation from C_4 to C_9:

$$K_9 = PB_9 \oplus H_2(K_4, ID_9, ID_4).$$

2) Because of the relationship of $C_{11} \preceq C_7 \preceq C_1$ in the hierarchy, the key derivation from C_1 to C_{11} is

$$K_{11} = PB_{11} \oplus H_2(K_1^{\prod_{allC_l, C_l \preceq C_1 \wedge C_l \not\preceq C_7} H_1(ID_l)} \mod m, ID_{11}, ID_7).$$

The proof is stated as follows:

$$\begin{aligned} K_{11} &= PB_{11} \oplus H_2(K_7, ID_{11}, ID_7) \\ &= PB_{11} \oplus H_2(g^{d_7} \mod m, ID_{11}, ID_7) \\ &= PB_{11} \oplus H_2(g^{(d_7 d_1 d_2 d_3 d_4 d_5 d_6 d_{10})(e_1 e_2 e_3 e_4 e_5 e_6 e_{10})} \mod m, ID_{11}, ID_7) \\ &= PB_{11} \oplus H_2(K_1^{\prod_{allC_l, C_l \preceq C_1 \wedge C_l \not\preceq C_7} H_1(ID_l)} \mod m, ID_{11}, ID_7). \end{aligned}$$

3) The key derivation from C_1 to C_4 is the formula

$$K_4 = K_1^{\prod_{allC_l, C_l \preceq C_1 \wedge C_l \not\preceq C_4} H_1(ID_l)} \mod m.$$

The proof is stated as follows:

$$\begin{aligned} K_4 &= g^{d_4} \mod m \\ &= g^{(d_4 d_1 d_2 d_3 d_5 d_6 d_7 d_{10})(e_1 e_2 e_3 e_5 e_6 e_7 e_{10})} \mod m \\ &= K_1^{\prod_{allC_l, C_l \preceq C_1 \wedge C_l \not\preceq C_4} H_1(ID_l)} \mod m. \end{aligned}$$

3.3 Dynamic Key Management

It is necessary to provide the dynamic key management ability for a hierarchical key assignment scheme. This section discusses the key changes of adding and deleting classes.

3.3.1 Adding a New Class with ID_{new}

1) Adding a New Leaf Class with Only One Immediate Ancestor:

   a. The immediate ancestor of the new class is not a leaf class with only one immediate ancestor before the adding.

   The new class C_{new} is the immediate successor of C_r. CA generates random key K_{new} and XOR public parameter PB_{new} as Step 5 in key generation phase. At last, CA modifies the authorized relationship to add C_{new}.

   b. The immediate ancestor of the new class is a leaf class with only one immediate ancestor before the adding.

   The new class C_{new} is an immediate successor of C_r. CA generates pair (e_r, d_r) as Step 4, and new key K_r and random key K_{new} for C_r and C_{new} as Step 5 in key generation phase, respectively. Then, CA modifies the secret keys of C_r's ancestors, and the public parameters of C_r and C_{new}. At last CA modifies the authorized relationship to add C_{new}.

2) Adding a New Non-leaf Class with Only One Immediate Ancestor or a Class with Multiple Immediate Ancestors in the Hierarchy:

   a. The new class is a non-leaf class with only one immediate ancestor.

   The new class C_{new} is the immediate successor of C_{r_1} and the immediate ancestor of C_{r_2}. CA generates pair (e_{new}, d_{new}) as Step 4, and new key K_{new} as Step 5 in key generation phase. Then, CA modifies the secret keys of C_{new}'s ancestors. At last CA modifies the authorized relationship to add C_{new}.

   b. The new class is a class with multiple immediate ancestors in the hierarchy.

   The new class C_{new} is the immediate successor of C_{r_1} and C_{r_2}. CA generates pair (e_{new}, d_{new}) as Step 4, and new key K_{new} as Step 5 in key generation phase. If C_{r_1} is a leaf class with only one immediate ancestor, CA generates pair (e_{r_1}, d_{r_1}) as Step 4. If C_{r_2} is a leaf class with only one immediate ancestor, CA does the same work for C_{r_2} as the scene of C_{r_1}. Then, CA modifies the secret keys of C_{new}'s ancestors. At last CA modifies the authorized relationship to add C_{new}.

3.3.2 Removing a Class with ID_{del}

1) Removing a Leaf Class with Only One Immediate Ancestor:

   a. The immediate ancestor of the removing class is not a leaf class with only one immediate ancestor after the deleting.

   The removing class C_{del} is a leaf class. CA deletes the key K_{del} and public parameter PB_{del} of C_{del}. At last CA modifies the authorized relationship to remove C_{del}.

b. The immediate ancestor of the removing class is a leaf class with only one immediate ancestor after the removing.

The removing class C_{del} is a leaf class with only one immediate ancestor C_r. CA deletes the key K_{del} and public parameter PB_{del} of C_{del}. CA deletes the key K_r and the prime pair (e_r, d_r) and modifies the secret keys of C_r's ancestors. Then, CA generates random key and XOR public parameter for C_r as Step 5 in the key generation phase. At last CA modifies the authorized relationship to remove C_{del}.

2) Removing a Leaf Class with Multiple Immediate Ancestors in the Hierarchy:

The removing class C_{del} is a leaf class with multiple immediate ancestor $C_{r_1}, \cdots, C_{r_t}$. CA deletes the key K_{del} and the prime pair (e_{del}, d_{del}) of C_{del}. Then, CA does a test about the class C_{r_1}. If C_{r_1} is a leaf class with only one immediate ancestor after the removing, CA deletes the secret key and prime pair of C_{r_1}, and generates random key K_{r_1} and modifies public parameter PB_{r_1} for C_{r_1} as Step 5 in key generation phase. Then, CA modifies the secret keys of C_{del}'s ancestors. If C_{r_1} is not a leaf class with only one immediate ancestor after the removing, CA only modifies the secret keys of C_{del}'s ancestors. CA does the same test on the class $C_{r_2}, \cdots, C_{r_t}$. At last CA modifies the authorized relationship to remove C_{del}.

3) Removing a Non-leaf Class:

a. All immediate successors of the removing class are not leaf classes with only one immediate ancestor after the deleting.

The removing class C_{del} is a non-leaf class. CA deletes the key K_{del} and the prime pair (e_{del}, d_{del}) of C_{del}. Then, CA modifies the secret keys of C_{del}'s ancestors. At last CA modifies the authorized relationship to remove C_{del}.

b. Any immediate successor of the removing class is a leaf class with only one immediate ancestor.

The removing class C_{del} is a non-leaf class and $C_{r_1}, \cdots, C_{r_t}$ are the immediate successors of C_{del} with only one immediate ancestor after the removing. CA deletes the key K_{del} and the prime pair (e_{del}, d_{del}) of C_{del} and modifies the secret keys of C_{del}'s ancestors. Then, CA generates new random keys and XOR public parameters for $C_{r_1}, \cdots, C_{r_t}$ as Step 5 in key generation. At last CA modifies the authorized relationship to remove C_{del}.

4 The Efficiency Analysis Between Lo's and Our Scheme

The efficiency comparisons are comprised of the number of changed public parameters in dynamic key management, space complexity of storage for public parameters and time complexity of public key generation in key derivation.

4.1 The Number of Changed Public Parameters in Dynamic Key Management

In general scene, CA modifies the public parameters of the changed node and its all ancestors in Lo's scheme, but only the public parameters of the changed node and its immediate ancestors in our improved scheme. So, our scheme has less modification about public parameters in the dynamic management than Lo's scheme. Firstly, the computation of changed public parameters which are executed on CA are reduced. Then, it is easily inferred that the bottle of the interaction is alleviated between CA and trusted public platform.

4.2 The Efficiency Comparison on Space Complexity of Public Parameters Storage

In Lo's scheme, the public parameter is a product from a series of large prime numbers, where every one is coprime with $\varphi(m)$ and satisfies the condition $m^{7/8} \leq e_i \leq m$ [3]. Two disadvantages on storage are inevitable. The first one is that the storage of public parameter for each class must be provided as the largest product $PB_i = \prod_{allC_l, C_l \preceq C_i} e_l$. Obviously, the storage space is very large and more wasteful because many public parameters are shorter than the largest public parameters. The other one is the stability. When the hierarchical access control layer adds in dynamic key management, the length of largest public parameter adds. CA must modify the length of storage on public parameter for each class. With the operation, the storage of public parameters changes deeply. Not only more interaction between CA and trusted public platform but also lower storage stability on trusted public platform have to be considered.

In our improved scheme, the public parameters are divided into two parts as Table 3. The first part is about identity, which is the input of Hash function, so the length of output about Hash function is fixed. So, the storage on public parameter for non-leaf class and leaf class with two or more immediate ancestors are cancelled. The other part is XOR public parameter, which is related with the length of private key, but the private key is no more than m bits. So, the storage on public parameter for each class is m bits. Now, only one base table are considered for the classes with only one immediate ancestor. Because of the same length public parameters, the utilization of storage space is improved greatly. The public parameter is constant size, so the re-operation about the length of storage on public parameter for each node does not exist in dynamic key management. It is good for the system stability.

Table 3: The comparison on length of public parameter

Schemes	The public parameter on leaf class with only one immediate ancestor	The public parameter on non-leaf class or leaf class with more immediate ancestors
Lo's	m bits	$t_i \cdot m$ bits, where t_i denotes the number of C_i and C_i's successors except leaves with only one immediate ancestor
Our	m bits	0 bits

4.3 The Efficiency Comparison on Time Complexity of Key Derivation

For a simple description, a POSET is defined as following: C_j, C_i has t_j, t_i child nodes, respectively, and they have the relationship $C_i \preceq C_j$ and $t_i \leq t_j$. A division must be executed between C_j's and C_i's public parameters for the derivation key in Lo's scheme. It is a division of $t_j e$ bits, because of the relationship $C_i \preceq C_j$ and $PB_j = \prod_{allC_l, C_l \preceq C_j} e_l$. The latest research transforms one time division to the equal length multiplication in the time complexity $O(n^{\log_2 3})$ [18], so this computation is $t_j e$ bits multiplication in the time complexity $O(n^{\log_2 3})$. In our improved scheme, it is an e bits accumulative multiplication in $t_j - t_i$ times, because of the derivation key $z_{ji} = \prod_{allC_l, C_l \preceq C_j \wedge C_l \npreceq C_i} e_l$. So this computation is no more than $(t_j - t_i)e$ bits multiplication in the time complexity $O(n)$. Obviously, our improved scheme is more efficient than Lo's scheme because of $O(n) < O(n^{\log_2 3})$ and $(t_j - t_i)e \leq t_j e$ as Table 4.

Table 4: The comparison of time complexity

Schemes	The length of lager number	The time complexity of computation
Lo's	$t_j \cdot e$ bits	$O(n^{log_2 3})$
Our	$(t_j - t_i) \cdot e$ bits	$O(n)$

5 Proof of Security

Theorem 2. *The improved scheme is secure on the key recovery model in Definition 1.*

Proof. Challenger C constructs K'_{v^*}, but attacker A only have a negligible advantage to distinguish the K'_{v^*} and K_{v^*}.

Algorithm $STAT_v^{our}(1^\tau, G', ID, corr'_v)$:

1) Let us construct an input for $STAT_v^{our}(1^\tau, G', ID, corr'_v)$, where $g_{HL} = g_{our}$.
   - $G = G'$;
   - $pr = H(ID)$;
   - $u = v$ is the attacked class;
   - $corr_u = corr'_v$ are the keys of the classes which are corrupted.
2) Let the input of $STAT_v^{our}$ is $(1^\tau, G', ID, corr'_v)$ and the output is K_u^{our}.
3) The output of the classes except the leaf class with only one immediate ancestor are $K_u^{HL} = K_u^{our}$, which is corresponding to $K_{v*}^{HL'} = K_{v*}^{our'}$. Thus,
   $$Pr_{our}[k'_{v*} = k_{v*}] = Pr_{HL}[k'_{v*} = k_{v*}].$$
   So, we can conclude that our scheme is have the same security with Harn-Lin scheme.
4) Assume that we have the result of $K_{v*}^{HL'} = K_{v*}^{our'}$ as 3). Concluding the front result and the random secret key K_i of leaf class which attacker A owns, we modify the public parameter of leaf class to $PB_i = K_i \oplus H_2(K_{v*}^{our'}, ID_i, ID_u) = K_i \oplus H_2(K_{v*}^{HL'}, ID_i, ID_u)$. Now, it is a negligible advantage to distinguish the K'_{v*} and K_{v*} for the attacker who only has the leaf class key . Thus,
   $$Pr_{our}[K'_{v*} = K_{v*}] = |Pr_{HL}[K'_{v*} = K_{v*}] + \varepsilon_{H^{-1}}|.$$
   So, we can conclude that our scheme is have the same security with Harn-Lin scheme.

Combining the above result, we can conclude that our scheme has the same security with Harn-Lin scheme. The security proof of Harn-Lin scheme on the key recovery model is provided in [7]. So, our improved scheme is secure on the key recovery model. This concludes the Theorem 2. □

6 Conclusion

The paper proposes an improved ID-based hierarchical cryptography access control scheme which the public platform only store constant size of public parameter for the leaf classes with only one immediate ancestor. The improved scheme does a great optimization as following: The first one is the storage space of public parameters. The second one is the system stability on public platform part in key dynamic management, the third one is the information interaction between CA and the public platform and the last is the efficiency of key derivation. Then, to reduce the storage space further, the paper introduces ID as a part of public parameter. Comparing with the same type schemes, the improved ID-based scheme has high efficiency on space and time complexity and less interaction between CA and the public platform of trusted. At last, the paper does the work of security analysis for the improved ID-based scheme on the key recovery model.

7 Acknowledgments

This work is supported by Science and Technology on Communication Security Laboratory Foundation (NO.9140C110301110C1103) and The Weaponry Equipment Pre-Research Foundation, the PLA General Armament Department (NO.9140A04020311DZ02).

References

[1] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *ACM Transactions on Computer Systems*, vol. 1, no. 3, pp. 239–248, 1983.

[2] G. Ateniese, A. D. Santis, L. A. Ferrara, et al. "Provably-secure time-bound hierarchical key assignment schemes," *Journal of Cryptology*, vol. 25, no. 2, pp. 243–270, 2012.

[3] J. Blömer, A. May, "New partial key exposure attacks on RSA," in *Advances in Cryptology (Crypto'03)*, pp. 27–43, Springer, 2003.

[4] M. Y. Chen, C. W. Liu, and M. S. Hwang, "Securedropbox: a file encryption system suitable for cloud storage services," in *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, pp. 21, 2013.

[5] M. Y. Chen, C. C. Yang, and M. S. Hwang, "Privacy protection data access control," *International Journal of Electronics and Information Engineering*, vol. 15, no. 6, pp. 411–419, 2013.

[6] H. Y. Chien, "Efficient time-bound hierarchical key assignment scheme," *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, no. 16, pp. 1301–1304, 2004.

[7] P. D'Arco, A. De Santis, A. L. Ferrar, et al. "Variations on a theme by Akl and Taylor: Security and tradeoffs," *Theoretical Computer Science*, vol. 411, no. 1, pp. 213–227, 2010.

[8] S. M. El-Sayed, H. M. A. Kader, M. M. Hadhoud, and D. S. AbdElminaam, "Mobile cloud computing framework for elastic partitioned/modularized applications mobility," *International Journal of Electronics and Information Engineering*, vol. 1, no. 2, pp. 53–63, 2014.

[9] D. Giri and P. D. Srivastava, "A cryptographic key assignment scheme for access control in poset ordered hierarchies with enhanced security," *International Journal of Network Security*, vol. 7, no. 2, pp. 223–234, 2008.

[10] D. Giri and P. D. Srivastava, "An asymmetric cryptographic key assignment scheme for access control in tree structural hierarchies," *International Journal of Network Security*, vol. 4, no. 3, pp. 348–354, 2007.

[11] D. Giri and P. D. Srivastava, "An asymmetric cryptographic key assignment scheme for access control in tree structural hierarchies," *International Journal of Electronics and Information Engineering*, vol. 4, no. 3, pp. 348–354, 2007.

[12] L. Harn and H. Y. Lin, "A cryptographic key generation scheme for multilevel data security," *Computer & Security*, vol. 9, no. 6, pp. 539–546, 1990.

[13] M. S. Hwang, "Cryptanalysis of ycn key assignment scheme in a hierarchy," *Information Processing Letters*, vol. 3, no. 73, pp. 97–101, 2003.

[14] M. S. Hwang, J. W. Lo, and C. H. Liu, "Improvement on the flexible tree-based key management framework," *Computers & Security*, vol. 24, no. 6, pp. 500–504, 2005.

[15] M. S. Hwang and W. P. Yang, "Controlling access in large partially ordered hierarchies using cryptographic keys," *The Journal of Systems and Software*, vol. 67, no. 2, pp. 99–107, 1990.

[16] C. C. Lee, Y. M. Lai, and C. S. Hsiao, "Cryptanalysis of a simple key assignment for access control based on polynomial," *Journal of Information Security and Applications*, vol. 18, no. 4, pp. 215–218, 2013.

[17] I. C. Lin, M. S. Hwang, and C. C. Chang, "A new key assignment scheme for enforcing complicated access control policies in hierarchy," *Future Generation Computer Systems*, vol. 4, no. 19, pp. 457–462, 2003.

[18] H. Liu and Z. Yu, *The time complexity of the large integer division depending on the large integer multiplication*, 2011. (`http://wenku.baidu.com/view/7d7f17120b4e767f5acfce32`)

[19] J. W. Lo, M. S. Hwang, and C. H. Liu, "A simple key assignment for access control based on polynomial," *The Arabian Journal for Science and Engineering*, vol. 38, no. 6, pp. 1397–1403, 2013.

[20] J. W. Lo, M. S. Hwang, and C. H. Liu, "An efficient key assignment scheme for access control in a large leaf class hierarchy," *Information Sciences*, vol. 181, no. 4, pp. 917–925, 2011.

[21] S. J. Mackinnon, P. D. Taylor, H. Meijer, and S. G. Akl, "An optimal algorithm for assigning cryptographic keys to control access in a hierarchy," *IEEE Transactions on Computers*, vol. 34, no. 9, pp. 797–802, 1985.

[22] M. Nikooghadam, A. Zakerolhosseini, and M. E. Moghadam, "Efficient utilization of elliptic curve crypto system for hierarchical access control," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1917–1929, 2010.

[23] V. Odelu, A. K. Das, and A. Goswami, "An effective and secure key-management scheme for hierarchical access control in e-medicine system," *Journal of Medical Systems*, vol. 37, no. 2, pp. 1–18, 2013.

[24] A. D. Santis, A. L. Ferrara, and B. Masucci, "Enforcing the security of a time-bound hierarchical key assignment scheme," *Information Sciences*, vol. 12, no. 176, pp. 1684–1694, 2006.

[25] T. H. Sun and M. S. Hwang, "A hierarchical data access and key management in cloud computing," *Innovative Computing, Information and Control Express Letters*, vol. 6, no. 2, pp. 569–574, 2012.

[26] S. F. Tzeng, C. C. Lee, and T. C. Lin, "A novel key management scheme for dynamic access control in a

hierarchy," *International Journal of Network Security*, vol. 12, no. 3, pp. 178–180, 2011.

[27] W. G. Tzeng, "A time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 1, no. 14, pp. 182–188, 2002.

[28] S. Y. Wang and C. S. Laihn, "Cryptanalysis of Hwang–Yang scheme for controlling access in large partially ordered hierarchies," *The Journal of Systems and Software*, vol. 75, no. 1, pp. 189–192, 2005.

[29] S. Wu and K. Chen, "An efficient key-management scheme for hierarchical access control in e-medicine system," *Journal of Medical Systems*, vol. 36, no. 4, pp. 1–13, 2012.

[30] J. Yeh, R. Chow, and R. Newman, "A key assignment for enforcing access control policy exceptions," in *Proceedings on International Symposium on Internet Technology*, pp. 54–59, 1998.

[31] J. Yeh, R. Chow, and R. Newman, "Key assignment for enforcing access control policy exceptions in distributed systems," *Information Sciences*, vol. 152, pp. 63–88, 2003.

[32] X. Yi and Y. Ye, "Security of tzeng's time-bound cryptographic key assignment scheme for access control in a hierarchy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, no. 15, pp. 1054–1055, 2003.

[33] Y. Zhou, J. Liu, H. Deng, et al. "Non-interactive revocable identity-based access control over e-healthcare records," in *Information Security Practice and Experience*, LNCS 9065, pp. 485–498, Springer, 2015.

Rang Zhou is a PH.D. student in the Department of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). He received his B.S. and M.S. degrees from the Department of Computer Science and Engineering, UESTC in 2009 and 2013, respectively, P.R.China. His research interests include: Access control security, cryptography and cloud security.

Chunxiang Xu is a professor in the Department of Computer Science and Engineering, UESTC. She received her B.S., M.S. and PH.D. degrees from XiDian University in 1985, 1988 and 2004, respectively, P.R.China. Her research interests include: Information security, cloud security and cryptography.

Wanpeng Li received his B.S. degree from Xihua University, P.R.China in 2010, and M.S. degree from UESTC P.R.China in 2013. He currently is a PH.D. student in the Department of Royal Holloway, University of London. His research interests include: Forward security and cryptography.

Jining Zhao received his B.S. degree from HeNan Normal University, P.R.China in 2009 and M.S. degree from UESTC P.R.China in 2013. He currently is a PH.D. student in the Department of Computer Science and Engineering, UESTC. His research interests include: Cloud auditing security and cryptography.