An Access Control Scheme for Partially Ordered Set Hierarchy with Provable Security

Jiang Wu and Ruizhong Wei*

Department of Computer Science, Lakehead University
Thunder Bay, Ontario P7B 5E1, Canada
jwu1@lakeheadu.ca
wei@ccc.cs.lakeheadu.ca

Abstract. In a hierarchical structure, an entity has access to another if and only if the former is a superior of the later. The access control scheme for a hierarchy represented by a partially ordered set (poset) has been researched intensively in the past years. In this paper, we propose a new scheme that achieves the best performance of previous schemes and is provably secure under a comprehensive security model.

1 Introduction

In many situations, the hierarchical systems can be represented by a partially ordered set (poset). In such a hierarchy, all users are allocated into a number of disjoint sets of security classes $p_1, p_2, \cdots, p_n$. A binary relation $\leq$ partially orders the set $\mathcal{P} = \{p_1, p_2, \cdots, p_n, \}$. The users in p_j have access to the information held by users in p_i if and only if the relation $p_i \leq p_j$ held in the poset $(\mathcal{P}, \leq)$. If $p_i \leq p_j$, p_i is called a successor of p_j, and p_j is called a predecessor of p_i. If there is no p_k such that $p_i \leq p_k \leq p_j$, the p_i is called an immediate successor of p_j, and p_j is called an immediate predecessor of C_i.

A straightforwawrd access control scheme for poset hierarchy is to assign each class with a key, and let a class have the keys of all its successors. The information belonging to a class is encrypted with the key assigned to that class, therefore the predecessors have access to the information of their successors. This is awkward because the classes in higher hierarchy have to store a large number of keys. In the past two decades, many schemes based on cryptography have been proposed to ease the key management in the hierarchy. Generally, these schemes are aimed to fully or partly achieve the following goals:

- *Support any arbitrary poset.* It is desirable that any arbitrary poset is supported. Some schemes only support special cases of poset such as a tree. Such schemes are considered restrictive in application.

* Research supported by NSERC grant 239135-01

– *Be secure under attacks.* The schemes are supposed to withstand attacks. For example, a user may try to derive the key of a class that is not his/her successor. The schemes should be secure under all possible attacks.
– *Require small storage space.* Any scheme needs a user in a class to store a certain amount of secret or public parameters for key derivation. All the schemes tried to reduce the amount of parameters stored.
– *Support dynamic poset structures.* The structure of a hierarchy may change. Classes may be added to or deleted from the hierarchy. In these cases the users in the classes (not only the ones added and deleted) need to update the parameters they store. It is desirable that when a change takes place, the number of classes involved in updating their parameters is as small as possible.

Several hierarchical access control schemes have been proposed in the last two decades. [1–3] are direct access schemes based on the RSA problem. In a direct access scheme, a predecessor can derive the key of a successor directly from the public parameters of that successor. These scheme are proven secure under a general security model. The disadvantages of this group of schemes include large storage spaces and lack of dynamics. [4–6] are indirect access schemes based on one-way functions. In these schemes, to derive the key of a successor, a predecessor has to derive the key of each class between them. The indirect schemes achieve smaller storage spaces and better dynamics. However, [4] only supports tree hierarchies. None of the indirect access schemes provided formal security proof under a general secure model, except in [5] a sketch was given, but [6] indicated that the implementation of the scheme in [5] is difficult. For these existing hierarchical access control schemes, to achieve the above four requirements simultaneously is still a challenge.

In this paper, we propose a new scheme that is superior to the previous schemes in that it provides both good performance and provable security, and is easy to implement. When we talk about security of the hierarchical access control scheme, we refer to the following security model:

Definition 1. *A hierarchical access control scheme for poset hierarchy is secure if for any group of classes in the poset, it is computationally infeasible to derive the key of any class that is not a member of that group, nor a successor of any member of that group.*

Our scheme is an indirect access scheme, which has similar performance in storage and dynamics to other indirect access schemes. The significant part of our scheme is its formal security proof under this comprehensive security model, which the previous indirect access schemes did not provide.

The rest of this paper is organized as follows: Section 2 presents the scheme, Section 3 analyzes its security, Section 4 compares the performance of the schemes, and Section 5 concludes this paper.

2 Proposed Scheme

2.1 Preliminary

Poset Representation For a given hierarchy structure, its corresponding poset $(\mathcal{P}, \leq)$ can be represented by a Hasse diagram, which is a graph whose nodes are classes of $\mathcal{P}$ and the edges correspond to the $\leq$ relation (in the rest of the paper we use "node" and "class" interchangeably). An edge from $p_j \in \mathcal{P}$ to $p_i \in \mathcal{P}$ is present if $p_i < p_j$ and there is no $p_k \in \mathcal{P}$ such that $p_i < p_k$ and $p_k < p_j$. If $p_i < p_j$, then p_j is drawn higher than p_i. Because of that, the direction of the edges is not indicated in a Hasse diagram. Fig. 1 shows an example of poset represented as a Hasse diagram.

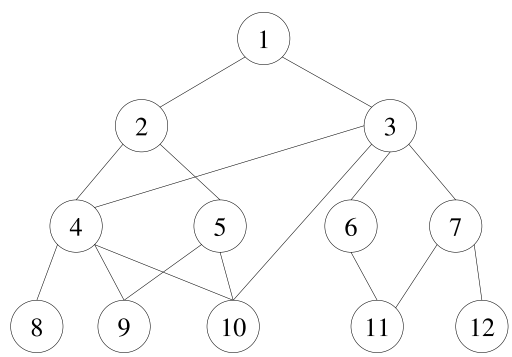

Fig. 1. Example of a Hasse diagram

Auxiliary Function We introduce a function that will be used in our scheme below. Let $p = 2q + 1$ where p, q are all odd primes. Let $\mathbb{G}$ be the subgroup of $\mathbb{Z}_p^*$ of order q. We define a function $f : \mathbb{G} \to [1, q]$ as follows:

$$f(x) = \begin{cases} x; & x \leq q \\ p - x; & x > q \end{cases} \tag{1}$$

For any $x \in \mathbb{Z}_p^*$, if $x \in \mathbb{G}$, then $-x \notin \mathbb{G}$. So the above function is a bijection. If x is a random variable uniformly distributed on $\mathbb{G}$, $f(x)$ is uniformly distributed on $[1, q]$.

2.2 Key Management

The key management of the scheme consists of two procedures: the key generation and the key derivation.

Key Generation

1. The central authority (CA) chooses a group $\mathbb{Z}_p^*$, where $p = 2q + 1$, p and q are both large primes. $\mathbb{G}$ is the subgroup of $\mathbb{Z}_p^*$ of order q.
2. From the top-level classes, the CA traverses the Hasse diagram of the hierarchy with width-first algorithm. For each node p_i, run the following key assignment algorithm to assign its public parameters g_i, $h_{i,j}$ and a secret key k_i:

Algorithm 1 Key Assignment

---

set g_i to be a unique generator of $\mathbb{G}$
if p_i does not have any immediate predecessor **then**
  set k_i to be a number chosen from $[1, q]$ at random
else if p_i has only one immediate predecessor p_j **then**
  $k_i = f(g_i^{k_j})$
else
  {comment: p_i has more than one immediate predecessors}
  let $\mathcal{X}$ be the set of keys of p_i's immediate predecessors
  $x = \prod_{x_i \in \mathcal{X}} x_i$
  $k_i = f(g_i^x)$
  for all $x_j \in \mathcal{X}$ **do**
    $h_{i,j} = g_i^{x/x_j}$
  end for
end if

For example, the nodes in Fig. (1) will be assigned with the following secret key and public parameters:

Node ID	secret key	public parameters
1	k_1	-
2	$k_2 = f(g_2^{k_1})$	g_2
3	$k_3 = f(g_3^{k_1})$	g_3
4	$k_4 = f(g_4^{k_2 k_3})$	$h_{4,2} = g_4^{k_3}, h_{4,3} = g_4^{k_2}$
5	$k_5 = f(g_5^{k_2})$	g_5
6	$k_6 = f(g_6^{k_3})$	g_6
7	$k_7 = f(g_7^{k_3})$	g_7
8	$k_8 = f(g_8^{k_4})$	g_8
9	$k_9 = f(g_9^{k_4 k_5})$	$h_{9,4} = g_9^{k_5}, h_{9,5} = g_9^{k_4}$
10	$k_{10} = f(g_{10}^{k_3 k_4 k_5})$	$h_{10,3} = g_{10}^{k_4 k_5}, h_{10,4} = g_{10}^{k_3 k_5}, h_{10,5} = g_{10}^{k_3 k_4}$
11	$k_{11} = f(g_{11}^{k_6 k_7})$	$h_{11,6} = g_{11}^{k_7}, h_{11,7} = g_{11}^{k_6}$
12	$k_{12} = f(g_{12}^{k_7})$	g_{12}

Key Derivation When a node needs to compute the key of one successor, it finds a path from itself to the successor in the Hasse diagram of the hierarchy.

Starting from its immediate successor in the path, the node go through the path, and computes k_i of every successor p_i along the path with the following algorithm:

Algorithm 2 Key Derivation

if p_i has only one predecessor p_j **then**
  $k_i = f(g_i^{k_j})$
else
  {comment: p_j is the predecessor of p_i that is on the path}
  $k_i = f(h_{i,j}^{k_j})$
end if

For example, in Fig. 1, node 1 is to derive the key of node 10. It finds the path $1 \rightarrow 3 \rightarrow 10$, and does the following computations:

$k_3 = f(g_3^{k_1})$
$k_{10} = f(h_{10,3}^{k_3})$

The correctness of the scheme is easy to be verified by reviewing the procedures in key generation and key derivation.

3 Security Analysis

3.1 Preliminary

On the group $\mathbb{G}$ used in our scheme, two standard assumptions, the discrete logarithm (DL) assumption and decisional Diffie-Hellman (DDH) assumption are believed to hold [7]. Another assumption, named group decisional Diffie-Hellman (GDDH) assumption is proven to hold based on DDH assumption on $\mathbb{G}$ too [8, 9]. To be concrete, let g be a generater of $\mathbb{G}$, a, b, c be random variables uniform on $[1, q]$, $\mathcal{X}$ be a set of random variables uniform on $[1, q]$, l be the binary length of q. Suppose $|\mathcal{X}|$ is polynomially bounded by l. Let $\prod(S)$ indicate the product of all elements in the set S. For any probabilistic polynomial time (in l) algorithms A, any polynomial Q, for l large enough, the three assumptions are formally expressed as follows:

DL assumption:

$$P_r[A(g, g^a) = a] < \frac{1}{Q(l)}$$

DDH assumption:

$$|P_r[A(g, g^a, g^b, g^{ab}) = 1] - P_r[A(g, g^a, g^b, g^c) = 1]| < \frac{1}{Q(l)}$$

For convenience, we use the notation from [8] to simplify the expression. We say that the probabilistic distributions (g, g^a, g^b, g^{ab}) and (g, g^a, g^b, g^c) are polynomially indistinguishable, and denote them as

$$(g, g^a, g^b, g^{ab}) \approx_{poly} (g, g^a, g^b, g^c)$$

GDDH assumption:

$$|P_r[A(g, g^{\prod(\mathcal{X})}, g^{\prod(S)}|S \subset \mathcal{X}) = 1] - P_r[A(g, g^c, g^{\prod(S)}|S \subset \mathcal{X}) = 1]| < \frac{1}{Q(l)}$$

or denoted as

$$(g, g^{\prod(\mathcal{X})}, g^{\prod(S)}|S \subset \mathcal{X}) \approx_{poly} (g, g^c, g^{\prod(S)}|S \subset \mathcal{X})$$

3.2 Security Proof

The security of our scheme is based on the above three assumptions. In the following parts, we prove the scheme is secure under Definition 1. We suppose the number of nodes in $\mathcal{P}$ is polynomially bounded by l (the binary length of $|\mathbb{G}|$), and all the algorithms considered below are polynomial time (in l) algorithms.

We choose an arbitrary node $p_t \in \mathcal{P}$ and suppose its secret key is k_t. Let $\mathcal{A}$ be the set of predecessors of p_t. We need to prove that, even when all the nodes in $\mathcal{P} - \mathcal{A} - \{p_t\}$ conspire, it is computationally intractable for them to derive k_t.

We group the set $\mathcal{P} - \mathcal{A} - \{p_t\}$ into three subsets: $\mathcal{B}$ the set of nodes in $\mathcal{P} - \mathcal{A}$ which do not have predecessors in $\mathcal{P} - \mathcal{A}$, and which is not p_t; $\mathcal{D}$ the set of nodes that are immediate successors of p_t; $\mathcal{R} = \mathcal{P} - \mathcal{A} - \{p_t\} - \mathcal{B} - \mathcal{D}$. The followings relations between $\mathcal{B}$, $\mathcal{D}$ and $\mathcal{R}$ are direct from their definitions:

- $\mathcal{B} \cup \mathcal{D} \cup \mathcal{R} = \mathcal{P} - \mathcal{A} - \{p_t\}$
- $\mathcal{B} \cap \mathcal{D} = \emptyset$, $\mathcal{R} \cap \mathcal{B} = \emptyset$ and $\mathcal{R} \cap \mathcal{D} = \emptyset$
- the nodes in $\mathcal{R}$ are successors of the nodes in $\mathcal{B}$, or $\mathcal{D}$, or both

An example of the above partition is as follows: in Fig. 1, suppose node 4 is the one we choose as the node p_t, then $\mathcal{A} = \{1, 2, 3\}, \mathcal{B} = \{5, 6, 7\}, \mathcal{D} = \{8, 9, 10\}, \mathcal{R} = \{11, 12\}$.

First we consider when all nodes in $\mathcal{B}$ conspire, what information about k_t they can learn. Suppose the generator assigned to node p_t is g_t, $\mathcal{X}$ is the set of secret keys of the immediate predecessors of node p_t. Let $\prod(S)$ be the product of all elements in the set S. Let $x = \prod(\mathcal{X})$, then $k_t = g_t^x$. The public parameters of p_t are

$$\{g_t, g_t^{\prod(S)}|S \subset \mathcal{X} \text{ and } |S| = |\mathcal{X}| - 1\}$$

The nodes $b_i \in \mathcal{B}$ with generators g_{b_i}, $i \in [1, n]$ may share the same predecessors with node p_t, thus may hold a subset of $\{g_{b_i}^{\prod(S)}|S \subseteq \mathcal{X}\}$ as their public parameters or secret keys. We assume that

$$\{g_{b_i}, g_{b_i}^{\prod(S)}|S \subseteq \mathcal{X}, i \in [1, n]\}$$

is all the information possibly held by nodes in $\mathcal{B}$ that is related to k_t. So the public parameters of p_t, plus the information pertaining to k_t held by $\mathcal{B}$ is a subset of

$$\{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\} \cup \{g_{b_i}, g_{b_i}^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}$$

We have the following result showing that even all nodes in $\mathcal{B}$ conspire, with the above information, they can not distinguish k_t from a random number on $[1, q]$. For convenient expression, the following theorem and its proof follow the notation style similar to that in [8].

Theorem 1. *Suppose DDH and GDDH assumptions hold on the group* $\mathbb{G}$*. Let* c *be a random variable uniform on* $[1, q]$*. The two distributions*

$$V_{b_n} = \left(g_t^x, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}, \{g_{b_i}, g_{b_i}^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}\right)$$

and

$$V'_{b_n} = \left(g_t^c, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}, \{g_{b_i}, g_{b_i}^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}\right)$$

are indistinguishable.

Proof. From GDDH assumption we have

$$\left(g_t^x, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}\right) \approx_{poly} \left(g_t^c, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}\right)$$

A polynomial time algorithm can choose z uniformly from $[1, q]$ at random, and reduce the above GDDH distribution pair to

$$V_b = \left(g_t^x, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^z, (g_t^z)^x, \{(g_t^z)^{\prod(S)} | S \subset \mathcal{X}\}\right)$$

$$V'_{im} = \left(g_t^c, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^z, (g_t^z)^c, \{(g_t^z)^{\prod(S)} | S \subset \mathcal{X}\}\right)$$

respectively. It follows that

$$V_b \approx_{poly} V'_{im}. \tag{2}$$

Let c_1 be a random variable uniform on $[1, q]$. Since zc_1 is independent of z and c, from DDH, we have

$$(g_t, g_t^z, g_t^c, g_t^{zc}) \approx_{poly} (g_t, g_t^z, g_t^c, g_t^{zc_1})$$

A polynomial time (in l) algorithm can choose $\mathcal{X}$ that is a set of random variables uniform on $[1, q]$, and whose order is polynomially bounded by l, and reduce the above DDH distribution pair to

$$V'_{im} = \left(g_t^c, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^z, (g_t^z)^c, \{(g_t^z)^{\prod(S)} | S \subset \mathcal{X}\}\right)$$

$$V''_{im} = \left(g_t^c, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^z, (g_t^z)^{c_1}, \{(g_t^z)^{\prod(S)} | S \subset \mathcal{X}\}\right)$$

respectively. It follows that

$$V'_{im} \approx_{poly} V''_{im} \tag{3}$$

Similarly, by choosing z and c uniformly from $[1, q]$ at random, a polynomial time (in l) algorithm can reduce the GDDH distribution pair

$$\left(g_t^{c_1}, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}\right) \approx_{poly} \left(g_t^{x}, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}\right).$$

to

$$V''_{im} = \left(g_t^{c}, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^{z}, (g_t^{z})^{c_1}, \{(g_t^{z})^{\prod(S)} | S \subset \mathcal{X}\}\right)$$
$$V'_b = \left(g_t^{c}, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^{z}, (g_t^{z})^{x}, \{(g_t^{z})^{\prod(S)} | S \subset \mathcal{X}\}\right).$$

respectively. It follows that

$$V''_{im} \approx_{poly} V'_b \tag{4}$$

From (2), (3) and (4), We conclude

$$V_b \approx_{poly} V'_b$$

i.e.,

$$\left(g_t^{x}, \{g_t, g_t^{\prod(S)} | S \subset \mathcal{X}\}, g_t^{z}, \{(g_t^{z})^{\prod(S)} | S \subseteq \mathcal{X}\}\right)$$
$$\approx_{poly} \left(g_t^{c}, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}\}, g_t^{z}, \{(g_t^{z})^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}\}\right).$$

By choosing z_i, $i \in [1, n]$ uniformly from $[1, q]$ at random, a polynomial time algorithm can reduce V_b and V'_b to

$$\left(g_t^{x}, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\}, \{g_t^{zz_i}, (g_t^{zz_i})^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}\right)$$
$$\left(g_t^{c}, \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}, \{g_t^{zz_i}, (g_t^{zz_i})^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}\right)$$

It follows that

$$V_{b_n} \approx_{poly} V'_{b_n}.$$

This completes our proof □

Then we consider when the nodes in $\mathcal{B}$ and $\mathcal{D}$ conspire, what information about k_t they can learn. The nodes $d_i \in \mathcal{D}$ assigned with generator g_{d_i}, $i \in [1, m]$ may hold a subset of the following information pertaining to k_t:

$$\{g_{d_i}, g_{d_i}^{k_t} | i \in [1, m]\}.$$

The following theorem shows that even all nodes in $\mathcal{B}$ and $\mathcal{D}$ conspire, they can not derive k_t:

Theorem 2. *It is intractable for any polynomial time (in l) algorithm to derive g_t^x from*

$$\mathcal{I} = \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\} \cup \{g_{b_i}, g_{b_i}^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\} \cup \{g_{d_i}, g_{d_i}^{f(g_t^x)} | i \in [1, m]\},$$

i.e., for any polynomial time (in l) algorithm A, any polynomial Q, if l is sufficiently large, then

$$P_r\left[A\left(\mathcal{I}\right) = f(g_t^x)\right] < \frac{1}{Q(l)}.$$

Proof. For convenience, let

$$\mathcal{V} = \{g_t, g_t^{\prod(\mathcal{S})} | \mathcal{S} \subset \mathcal{X}\} \cup \{g_{b_i}, g_{b_i}^{\prod(\mathcal{S})} | \mathcal{S} \subseteq \mathcal{X}, i \in [1, n]\}.$$

Step 1. Assume that there exist a polynomial time (in l) algorithm B, a polynomial Q_1 and a number L, for $l > L$

$$P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = f(g_t^x)] \geq \frac{1}{Q_1(l)} \tag{5}$$

where g_d is a generator of $\mathbb{G}$.

Let c be a random variable uniform on $[1, q]$, $Q_2(l) = 2Q_1(l)$. Suppose l is large enough. We consider the following two cases

- **Case 1**: $P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^c)}) = f(g_t^c)] \geq \frac{1}{Q_2(l)}$
  Notice that c is a random variable independent of $\mathcal{V}$. Let $z \in [1, q]$, we define the following algorithm $C(g_d, g_d^z)$:

Algorithm 3 $C(g_d, g_d^z)$

choose a generator of $\mathbb{G}$ as g_t
choose a set of n distinct generators of $\mathbb{G}$ as $\mathcal{B}$
choose a set of random variables uniform on $[1, q]$ as $\mathcal{X}$
compute $\mathcal{V}$ with g_t, $\mathcal{B}$ and $\mathcal{X}$
return $B(\mathcal{V}, g_d, g_d^z)$

  The algorithm C is a polynomial time (in l) algorithm. Since $z = f(g_t^c)$ for some $c \in [1, q]$ (though we do not know c), we have

$$\begin{aligned} P_r[C(g_b, g_b^z) = z] &= P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^c)}) = f(g_t^c)] \\ &\geq \frac{1}{Q_2(l)}. \end{aligned}$$

  This contradicts the DL assumption.
- **Case 2**: $P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^c)}) = f(g_t^c)] < \frac{1}{Q_2(l)}$
  From this inequality and (5), we have

$$\begin{aligned} &P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = f(g_t^x)] - P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^c)}) = f(g_t^c)] \\ \geq &\frac{1}{Q_1(l)} - \frac{1}{Q_2(l)} \\ = &\frac{1}{Q_2(l)} \end{aligned} \tag{6}$$

  Let $z \in \mathbb{G}$, we define the algorithm $D(\mathcal{V}, z)$ in Algorithm 4.

Algorithm 4 $D(\mathcal{V}, z)$

choose a generator of $\mathbb{G}$ as g_b
if $B(\mathcal{V}, g_d, g_d^{f(z)}) = f(z)$ **then**
  return 1
else
  return 0
end if

D is a polynomial time (in l) algorithm. From (6), we have

$$\begin{aligned}&P_r[D(\mathcal{V}, g_t^x) = 1] - P_r[D(\mathcal{V}, g_t^c) = 1]\\ =\ &P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = f(g_t^x)] - P_r[B(\mathcal{V}, g_d, g_d^{f(g_t^c)}) = f(g_t^c)]\\ \geq\ &\frac{1}{Q_2(l)}.\end{aligned}$$

That means D can distinguish the two distributions:

$$(\mathcal{V}, g_t^x) \text{ and } (\mathcal{V}, g_t^c).$$

This contradicts to Theorem 1.

Combining Case 1 and Case 2, we conclude that for any polynomial time (in l) algorithm B, any polynomial Q, for sufficiently large l,

$$P_r\left[B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = f(g_t^x)\right] < \frac{1}{Q(l)} \tag{7}$$

Step 2. Assume there exist a polynomial time (in l) algorithm A, a polynomial Q and a number L such that for $l > L$,

$$P_r\left[A\left(\mathcal{V}, \{g_{d_i}, g_{d_i}^{f(g_t^x)} | i \in [1, m]\}\right) = f(g_t^x)\right] \geq \frac{1}{Q(l)}.$$

Let $B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = A(\mathcal{V}, \{g_d^{z_i}, g_d^{z_i f(g_t^x)} | i \in [1, m]\})$ where $z_1, \cdots, z_m$ are random variables uniform on $[1, q]$, and m is polynomially bounded by l. We have

$$\begin{aligned}P_r\left[B(\mathcal{V}, g_d, g_d^{f(g_t^x)}) = f(g_t^x)\right] &= P_r\left[A(\mathcal{V}, \{g_d^{z_i}, (g_d^{z_i})^{f(g_t^x)} | i \in [1, m]\} = f(g_t^x)\right]\\ &\geq \frac{1}{Q(l)}\end{aligned}$$

This contradicts (7). Therefore for any polynomial time (in l) algorithm A, any polynomial Q, for sufficiently large l,

$$P_r\left[A\left(\mathcal{V}, \{g_{d_i}, g_{d_i}^{f(g_t^x)} | i \in [1, m]\}\right) = f(g_t^x)\right] < \frac{1}{Q(l)},$$

i.e.,

$$P_r\left[A\left(\mathcal{I}\right) = f(g_t^x)\right] < \frac{1}{Q(l)}.$$

This completes our proof. $\square$

Finally, we consider when all the nodes in $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{R}$ conspire, whether they are able to derive k_p. Since all the nodes in $\mathcal{R}$ are successors of $\mathcal{B}$ or $\mathcal{D}$ or both, the information held by $\mathcal{R}$ can be derived by a polynomial time (in l) algorithm from the information held by $\mathcal{B}$ and $\mathcal{D}$. Thus if $\mathcal{B} \cup \mathcal{D} \cup \mathcal{R}$ can derive k_p, then $\mathcal{B} \cup \mathcal{D}$ can derive k_p. This contradicts to Theorem (2). Therefore we conclude that the scheme is secure under the security model defined in Definition (1).

4 Performance Analysis

4.1 Storage Requirement

Our scheme is an indirect access scheme, and has similar storage requirement with other indirect schemes. In a hierarchy with N nodes where each node has at most M predecessors, the storage space required for a single node is about M for our scheme and other indirect schemes. For the direct schemes, to store the public information of one node, the maximum storage is about N numbers, or the product of the n numbers. In a real situation, N would be much greater than M, and N will increase as the scale of the hierarchy increases, while M usually keeps constant. So the indirect schemes achieves require less storage than the direct schemes.

4.2 Dynamics

As an indirect hierarchical access scheme, the operation of adding, deleting a node or link in our scheme is similar to other indirect access schemes. When a node is added or deleted, or a link is added to or deleted from a node, only the nodes that are successors of that node will be impacted, i.e., the secret key and public parameters of those nodes need to be updated. The direct schemes are quite different. In Akl-Taylor scheme, when a node is added or deleted, all the nodes except for its successors have to update their secret keys and public parameters. In Harn-Lin scheme, when a node is added or deleted, all its predecessors will be impacted. In addition, for these two schemes, to prevent a deleted node to access its former successors, the keys of these successors have to be changed too. In a practical hierarchy, there are much more low level nodes than high level nodes, and it is more likely that the low level nodes will change. Therefore in an indirect scheme, less nodes are impacted than in a direct scheme when the hierarchy structure changes. The indirect schemes are more suitable than direct schemes for a dynamic hierarchy.

4.3 Performance Summary

In summary, in view of performance in storage and dynamics, although our scheme does not improve previous indirect schemes, it inherits their performances, which are better than those of the direct schemes.

5 Conclusion

In this paper we proposed a new access control scheme for poset hierarchy. This scheme is concrete and practical for implementation. It supports any arbitrary poset, achieves the best performance of previous schemes, and provides a formal security proof under a comprehensive security model. None of the previous schemes achieved the properties as fully as ours does. Our scheme provides a solution with both practice and theoretical significance for the hierarchical access control problem.

Acknowledgment

The authors are grateful to David Wagner and Kristian Gjøteen for their helpful discussions on the security proof of the scheme.

References

1. Selim G. Akl, Peter D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, ACM Transactions on Computer Systems (TOCS) archive Volume 1 , Issue 3 (August 1983)
2. Stephen J. MacKinnon , Peter D. Taylor , Henk Meijer , Selim G. Akl, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy, IEEE Transactions on Computers, v.34 n.9, p.797-802, Sept. 1985
3. L. Harn , H.-Y. Lin, A cryptographic key generation scheme for multilevel data security, Computers and Security, v.9 n.6, p.539-546, Oct. 1990
4. Ravinderpal S. Sandhu, Cryptographic implementation of a tree hierarchy for access control, Information Processing Letters, v.27 n.2, p.95-98, February 29, 1988
5. Y. Zheng, T. Hardjono and J. Pieprzyk, The sibling intractable function family (SIFF): notion, construction and applications, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Vol. E76-A, No. 1, pp. 4-13, January 1993
6. Sheng Zhong, A practical key management scheme for access control in a user hierarchy, Computers & Security, Volume 21, Issue 8, November 2002, Pages 750-759
7. D. Boneh, The decision Diffie-Hellman problem. In Proceedings of the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science, Vol. 1423, Springer-Verlag, pp. 48–63, 1998.
8. Michael Steiner, Gene Tsudik, and Michael Waidner, Diffie-Hellman key distribution extended to group communication, in ACM Conference on Computer and Communications Security, 1996, pp. 31-37.
9. Emmanuel Bresson, Olivier Chevassut and David Pointcheval, The group Diffie-Hellman problems, Proceedings of Selected Areas in Cryptography (SAC'02), St John's, Newfoundland, Canada, August 15 - 16, 2002