A Cryptographic Key Assignment Scheme for Access Control in Poset Ordered Hierarchies with Enhanced Security

Debasis Giri and P. D. Srivastava
(Corresponding author: Debasis Giri)

Department of Mathematics, Indian Institute of Technology, Kharagpur-721302, India
(Email: {dgiri,pds}@maths.iitkgp.ernet.in)
(Received Agu. 01, 2006; revised Nov. 08, 2006; and accepted June 12, 2007)

Abstract

In a hierarchical structure, a user in a security class has access to information items of security classes of lower levels, but not of upper levels. Based upon cryptographic techniques, several schemes have been proposed for solving the problem of access control in hierarchal structures, which are based on only one cryptographic assumption. In this paper, we propose a scheme for access control in hierarchical structures that achieves better security, efficiency, flexibility and generality compared to the schemes previously published.

Keywords: Access control, cryptography, data security, key generation

1 Introduction

The concept of hierarchical access control is that an user of a higher security level class has the ability to access the information items (e.g., a message, data) in users of lower security level classes. Hierarchical structures are used in many applications including military, government, schools and colleges, private corporations, computer network systems [16, 19], operating systems [6] and database management systems [5].

In many situations, the hierarchical systems can be represented by a partially ordered set. We consider an organizational structure in which users and their own information items are divided into a number of disjoint set of security classes, say, C_0, C_1, ..., C_{n-1}, where i represents the identity of the class C_i. For a set $C = \{C_0, C_1, \ldots, C_{n-1}\}$, we call the relation "$\leq$" is partially ordered if it satisfies the following three properties:

1) *Reflexivity property:* For all $C_i \in C$, $C_i \leq C_i$;

2) *Anti-symmetric property:* If $C_i, C_j \in C$, $C_i \leq C_j$ and $C_j \leq C_i$ implies $C_i = C_j$;

3) *Transitivity property:* If $C_i, C_j, C_k \in C$, $C_i \leq C_j$ and $C_j \leq C_k$ implies $C_i \leq C_k$.

A set is partial ordered on "$\leq$" is called partially ordered set (poset, for short). We assume that the set $C = \{C_0, C_1, \ldots, C_{n-1}\}$ is partially ordered with respect to the relation "$\leq$", where $C_i \leq C_j$ means that C_i has security clearance lower than or equal to C_j. In other words, users in C_j can access the encrypted information held by users in C_i. But the opposite is not allowed. Figure1 shows an example of four level hierarchial structure. The top level classes posses the highest security, and security decreases with increase in the level. Users in bottom level classes have the least security. If $C_i \leq C_j$, C_i is called a successor of C_j, and C_j is called a predecessor of C_i. If there is no C_k such that $C_i \leq C_k \leq C_j$, the class C_i is called an immediate successor of C_j and C_j is called an immediate predecessor of C_i. If there is no C_k such that $C_j \leq C_k$, the class C_j is called leaf security class; otherwise, the class Cj is called a non-leaf security class. It is obvious that a predecessor class of any class is a non-leaf security class in a hierarchy.

Assume that a user in the security class C_6 in Figure 1 encrypts a message with his/her own encryption key. Because of access control in a hierarchical structure, only the users in the security class C_6 and his/her predecessors classes (i.e., C_3, C_1, C_0) can decrypt this message. Nobody else can decrypt this message.

A straightforward access control scheme for poset ordered hierarchy is to assign each security class with a key, and each class has the keys of all its successors. The information items belonging to a class is encrypted with the key assigned to that class. As a result, if a class encrypts the information items, its predecessors can only decrypt the encrypted information items. The drawback of such scheme is to store the keys in higher hierarchical classes. Many authors have proposed different methods for solving such type of problem using the concept of master key. In 1983, Akl-Taylor [1] proposed a scheme based on cryptography to access of information in a hier-

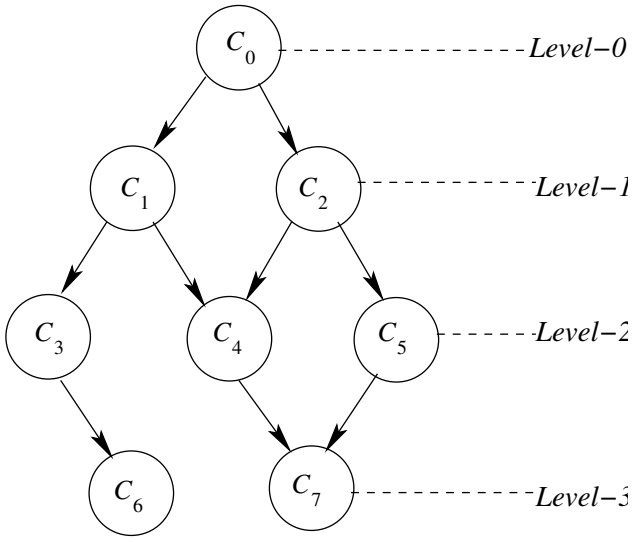

Figure 1: An example of a hierarchical structure

archy. Their solution was based on the RSA cryptosystem [26]. The advantage of this scheme is that the key generation/derivation algorithms are quite simple. In 1985, Mackinnon et al. [17] proposed an improved algorithm for the Akl-Taylor scheme based on top-down approach of poset ordered hierarchy for reducing the value of public parameters. In 1988, Sandhu [27] introduced a cryptographic implementation of a tree structural hierarchy for access control based on one-way function. In 1990, Harn-Lin [7] proposed a scheme which is similar to the scheme of Akl-Taylor, but, it is based on bottom-up approach for key generation. These above mentioned schemes have some drawbacks. Firstly, if the security classes in the hierarchy is large, a large storage space is required for storing the public parameters. Secondly, on the solutions of dynamic access control problems, the key assignment scheme encounters great difficulties in re-updating key. Finally, it is difficult to provide the user with a convenient way to change his/her secret key for the security considerations. To overcome these problems, a number of schemes [3, 13, 14, 15, 30, 32] related to access control have been proposed. In 1992 and 1993, both Chang et al. [3] and Liaw et al. [13, 14] proposed a scheme based on Newton's interpolations method and one-way function. In 2000, Hwang [9] proposed an access control scheme for a totally ordered hierarchy based on asymmetric cryptosystem. In 2001, Wu-Chang [32] proposed a cryptographic key assignment scheme to solve the access control policy using polynomial interpolations. But, this scheme has security flaws as described in [8, 31]. In 2003, Lin-Hwang-Chang [15] proposed a scheme for access control, where each security class contains a secret key SK_i and derivation key DK_i which are kept secret by the class C_i. If $C_i \leq C_j$, the class C_j can derive the secret key of the class C_i using the derivation key DK_j and public parameters. In this scheme requires only small amount of storage space to store public parameters compared to the Akl-Taylor's scheme [1]. In 2002, Shen-Chen [30] proposed a scheme which is based on discrete logarithm problems and the Newton's interpolating polynomials. The drawback of this scheme is that a large number of secret parameters becomes inconvenient to administer and hazardous to keep them secure. To overcome this problem, we propose a scheme for access control in poset ordered hierarchies based on one-way secure hash functions [21], the discrete logarithm problems [11, 18, 22], the factoring problems [12, 23, 24] and the Newton's interpolating polynomials [28]. Our scheme requires less amount of storage space to store secret parameters compared to the Shen and Chen's [30] scheme. Further, our scheme is applicable to a large-scale hierarchical model. This scheme also supports dynamic access control policy. Moreover, our scheme possesses the enhanced security compared to the existing schemes.

The remainder of this paper is organized as follows. Section 2 gives a brief review of the Shen and Chen scheme. In Section 3, we describe our proposed scheme for access control in poset ordered structural hierarchies. Section 4 shows the dynamic key management. In Section 5, we discuss the security analysis. Section 6 shows the space and time complexity of our scheme. In Section 7, our scheme is compared with previously published schemes. Finally, Section 8 concludes the paper.

2 Review of the Shen and Chen Scheme

In this section, we briefly review the Shen and Chen scheme [30]. There is a central authority (CA, for short) in the system. $ID_1, ID_2, \ldots, ID_n$ denote the identifiers of C_1, C_2, $\ldots, C_n$ respectively. CA selects two large primes P and P', such that $P = 2P' + 1$. Next, CA selects a primitive root g over Galois field $GF(P)$. Then, CA publishes g and P as public parameters. Then, CA assigns the secret parameters b_i and SK_i to the class C_i, for $i = 1, 2, \ldots, n$, where n is the number of classes in the hierarchical system, and $\gcd(b_i, P-1) = 1$ and $\gcd(SK_i, P-1) = 1$. CA computes a public parameter $Q_i = SK_i^{b_i^{-1}} \bmod P$, for $i = 1, 2, \ldots, n$. Then, CA computes a Newton's interpolating polynomial $f_i(x)$ over $GF(P)$ by interpolating at all the points $(ID_j || (g^{SK_i} \bmod P), b_j)$, where the index j corresponds to every successor C_j of C_i, ID_j is the identity of C_j and $||$ is a bit concatenation operator. Then, CA publishes the public parameter Q_i of C_i and transmits $(SK_i, f_i(x), b_i)$ to each class C_i in the hierarchy, where SK_i and b_i are transmitted securely to C_i. In the key derivation procedure, suppose $C_j \leq C_i$. Then, C_i can derive C_j's private key SK_j by computing $b_j = f_i(ID_j || (g^{SK_i} \bmod P))$ and $SK_j = Q_j^{b_j} \bmod P$.

3 The Proposed Scheme

In this section, we propose a new key assignment scheme for access control in a poset ordered structure hierarchy. We assume that there is a trusted central authority in the system. The main purpose of CA is to generate keys and distribute those keys to all classes in the hierarchy. Our scheme consists of five following procedures, namely, *system setup procedure*, *relationship building procedure*, *key generation procedure*, *public polynomial generation procedure* and *key derivation procedure*.

3.1 System Setup Procedure

CA chooses a large prime P so that $P = 2P_1 \cdot P_2 + 1$, where P_1 and P_2 are two distinct large primes. P_1 and P_2 are to be chosen at least 512 bits long primes for security considerations. CA computes $R = \frac{P-1}{2}$. CA then chooses a primitive root g over Galois field $GF(P)$. CA selects a prime Q such that $\lceil \log_2 Q \rceil \geq \lceil \log_2 P \rceil + \lceil \log_2 n \rceil$, where n is the number of security classes in the system. CA selects a symmetric cryptosystem (for example AES-256 [20]) in which $E_k(\cdot)$ and $D_k(\cdot)$ are the encryption and decryption algorithms with the key k respectively and a cryptographic one-way hash function $h(\cdot)$ (for example SHA-256 [21]). CA keeps g, P, Q, $h(\cdot)$, and encryption and decryption algorithms as public. In our scheme, we use AES-256 as symmetric cryptosystem and SHA-256 as cryptographic one-way hash function.

It is noted that the AES-256 has block length, cipher length and key length each of $L = 256$-bit. Further, in case of SHA-256, the message digest length of $h(\cdot)$ is L, which is same as the key length of AES-256. As a result, one can use symmetric secret key as the hashed value $h(r)$ of a long message, say, r. However, if r or $h(r)$ is not disclosed to an unauthorized third party or an adversary, it is computationally hard to recover m from c, where $c = E_{h(r)}(m)$.

3.2 Relationship Building Procedure

In this subsection, we construct a relationship list among all classes in a hierarchy in order to store the information regarding those relationships. It is noted that a hierarchy is represented as a directed acyclic graph, say, $G = (C, E)$, where $C = \{C_0, C_1, \ldots, C_{n-1}\}$ and $E = \{e_{j,i} \mid e_{j,i}$ is an edge from C_j to C_i (i.e., there is a directed path from C_j to C_i) with a relation $C_i \leq C_j$ for different C_i and C_j, where $C_i, C_j \in C\}$. C and E represent the vertex set and edge set of the graph G respectively and each $C_i \in C$ is considered as a vertex in the graph G. Then, CA publishes the graph G corresponding to the hierarchy. CA has only access to update the published graph G, i.e., the relationship among the classes $C_0, C_1, \ldots, C_{n-1}$ in that hierarchy.

It is noted that if there exists a relation between two different classes C_i and C_j with $C_i \leq C_j$ in a hierarchy, a path from C_j to C_i exists in graph G corresponding to that hierarchy.

3.3 Key Generation Procedure

In this subsection, we describe the key generation procedure to generate keys for all classes in a hierarchy by CA.

CA randomly chooses a secret key $SK_i \in \{0,1\}^L$ for each class C_i in the hierarchy, where $L = 256$. Then, CA transmits securely the secret key SK_i to each security class C_i in the hierarchy. C_i keeps SK_i as secret.

3.4 Public Polynomial Generation Procedure

In this subsection, we describe the public polynomial generation procedure to generate the Newton's interpolating polynomial [28] for each non-leaf security class in the hierarchy by CA.

The description of the public polynomial generation procedure over $GF(Q)$ is as follows:

1) CA chooses a class $C_i \in C$ from the graph G corresponding to the hierarchy, where i is the identity of the class C_i.

2) To construct the public derivation Newton's interpolating polynomials for the class C_i, CA first constructs the points containing the identities and secret keys of the immediate successors of C_i, and the identity i and the secret key SK_i of C_i. Consider that C_i has k number of immediate successors, say, $C_{i_1}, C_{i_2}, \cdots, C_{i_k}$, where i_u is the identity of the class C_{i_u}, $u \in \{1, 2, \ldots, k\}$. CA constructs the points $(i_u || DK_i, E_{h(i||i_u||SK_i^2)}(SK_{i_u}))$ for all u such that $u \in \{1, 2, \ldots, k\}$, where $||$ is a bit concatenation operator and $DK_i = g^{SK_i^3 \bmod R} \bmod P$ is the derivation key of the class C_i. Then, containing these points, CA derives the Newton's interpolating polynomial for the class C_i, which is denoted by $NIP_{i,i}(x)$ over $GF(Q)$. Next, CA computes the Newton's interpolating polynomial for the class C_i after constructing the points containing the identities and secret keys of the immediate successors of each C_{i_u}, $u \in \{1, 2, \ldots, k\}$, and the identity i and the secret key SK_i of C_i. Now, consider the case for the immediate successor C_{i_1} of C_i. For example, let C_{i_1} have only four immediate successors, say, C_a, C_b, C_c and C_d. Then, CA constructs four points $(a||DK_i, E_{h(i_1||a||SK_i^2)}(SK_a))$, $(b||DK_i, E_{h(i_1||b||SK_i^2)}(SK_b))$, $(c||DK_i, E_{h(i_1||c||SK_i^2)}(SK_c))$ and $(d||DK_i, E_{h(i_1||d||SK_i^2)}(SK_d))$. Then, containing these points, CA derives another Newton's interpolating polynomial for the class C_i, which is denoted by $NIP_{i,i_1}(x)$ over $GF(Q)$. Similarly, CA derives $NIP_{i,i_u}(x)$ for all $u \in \{2, 3, \ldots, k\}$ and then CA computes $NIP_{i,a}(x)$, $NIP_{i,b}(x)$, $NIP_{i,c}(x)$ and

$NIP_{i,d}(x)$ for the class C_i and so on for all successors of C_i, which are non-leaf security classes in the hierarchy. $NIP_{i,j}(x)$ stands for the Newton's interpolating polynomial for the class C_i at the points containing the identities and secret keys of all immediate successors of C_j, and the identity j of C_j, and the secret key SK_i and the derivation key DK_i of C_i.

To construct the public derivation Newton's interpolating polynomials for the class C_i, CA first constructs the points containing the identities and secret keys of the immediate successors of C_i, and the identity i and the secret key SK_i of C_i. Consider that C_i has k number of immediate successors, say, $C_{i_1}, C_{i_2}, \cdots, C_{i_k}$, where i_u is the identity of the class C_{i_u}, $u \in \{1, 2, \ldots, k\}$. CA constructs the points $(i_u||DK_i, E_{h(i||i_u||SK_i^2)}(SK_{i_u}))$ for all u such that $u \in \{1, 2, \ldots, k\}$, where $||$ is a bit concatenation operator and $DK_i = g^{SK_i^3 \bmod R} \bmod P$ is the derivation key of the class C_i. Then, containing these points, CA derives the Newton's interpolating polynomial for the class C_i, which is denoted by $NIP_{i,i}(x)$ over $GF(Q)$. Next, CA computes the Newton's interpolating polynomial for the class C_i after constructing the points containing the identities and secret keys of the immediate successors of each C_{i_u}, $u \in \{1, 2, \ldots, k\}$, and the identity i and the secret key SK_i of C_i. Now, consider the case for the immediate successor C_{i_1} of C_i. For example, let C_{i_1} have only four immediate successors, say, C_a, C_b, C_c and C_d. Then, CA constructs four points $(a||DK_i, E_{h(i_1||a||SK_i^2)}(SK_a))$, $(b||DK_i, E_{h(i_1||b||SK_i^2)}(SK_b))$, $(c||DK_i, E_{h(i_1||c||SK_i^2)}(SK_c))$ and $(d||DK_i, E_{h(i_1||d||SK_i^2)}(SK_d))$. Then, containing these points, CA derives another Newton's interpolating polynomial for the class C_i, which is denoted by $NIP_{i,i_1}(x)$ over $GF(Q)$. Similarly, CA derives $NIP_{i,i_u}(x)$ for all $u \in \{2, 3, \ldots, k\}$ and then CA computes $NIP_{i,a}(x)$, $NIP_{i,b}(x)$, $NIP_{i,c}(x)$ and $NIP_{i,d}(x)$ for the class C_i and so on for all successors of C_i, which are non-leaf security classes in the hierarchy. $NIP_{i,j}(x)$ stands for the Newton's interpolating polynomial for the class C_i at the points containing the identities and secret keys of all immediate successors of C_j, and the identity j of C_j, and the secret key SK_i and the derivation key DK_i of C_i.

Note that if a successor of C_i is a leaf security class, CA does not derive the Newton's interpolating polynomial for that successor.

3) CA repeats Step 2 until each non-leaf security class is taken in the hierarchy.

The above procedure is summarized by the following algorithm.

Algorithm 1.
input:

1) *$G = (C, E)$, a directed acyclic graph (as described in Subsection 3.2).*
2) *SK, an array in the range from 0 to $n-1$, where SK_i contains the secret key of C_i for $i = 0, 1, \ldots, n-1$.*
3) *n, the number of vertices of G, i.e., number of classes in the hierarchy.*

output: *The Newton's interpolating polynomials for every $C_i \in C$, where C_i is a non-leaf security class in G.*

Polynomial_Generation *(G, SK, n)*

```
{
  1. Integer: l, T, DK, X[0:n-1], Y[0:n-1];
        [comment: l, T and DK are three integer
        variables, and X and Y are two arrays of
        integer variables]
  2. while(C ≠ φ) do
        [comment: φ represents null set]
     {  2.1. Choose an element C_i ∈ C;
        2.2. Set IS1 contains all immediate successors
             of C_i;
        2.3. If IS1 = φ then goto step-2.9 ;
        2.4. T = SK_i^2;
        2.5. DK = g^(T·SK_i mod R) mod P;
                [comment: DK = g^(SK_i^3 mod R) mod P]
        2.6. Set S contains all successors of C_i;
        2.7. Set A = S ∪ {C_i};
        2.8. while (A ≠ φ) do
             {  2.8.1. Select an element C_j ∈ A;
                2.8.2. Set IS2 contains all immediate
                       successors of C_j;
                2.8.3. If IS2 = φ then goto step-2.8.8;
                2.8.4. l = 1;
                2.8.5. while (IS2 ≠ φ) do
                       {  2.8.5.1. Choose an element
                                   C_k ∈ IS2;
                          2.8.5.2. X_l = k||DK;
                          2.8.5.3. Y_l = E_h(j||k||T)(SK_k);
                        [comment: Y_l = E_h(j||k||SK_i^2)(SK_k)]
                          2.8.5.4. l = l + 1;
                          2.8.5.5. IS2 = IS2 \ {C_k};
                        [comment: "\" represents set minus]
                       }
                2.8.6. l = l − 1;
                2.8.7. Computes NIP_{i,j} containing the
                       points (X_r, Y_r) for 1 ≤ r ≤ l;
                2.8.8. A = A \ {C_j};
             }
        2.9. C = C \ {C_i};
     }
}
```

CA publishes all the Newton's interpolating polynomials (i.e., the coefficients of all the polynomials) corresponding to each non-leaf security class C_i in the hierarchy. But, only CA owns the authority to update public Newton's interpolating polynomials.

Example 1. *Let us revisit the hierarchical structure presented in Figure 1. Suppose CA runs the algorithm-1 to compute all the Newton's interpolating polynomials for all*

non-leaf security classes in the hierarchy, which are shown below.

The Newton's interpolating polynomials for the class C_0:

$NIP_{0,0}(x)$ is computed containing the following two points $(1||DK_0,\ E_{h(0||1||SK_0^2)}(SK_1))$ and $(2||DK_0,\ E_{h(0||2||SK_0^2)}(SK_2))$.

$NIP_{0,1}(x)$ is computed containing the following two points $(3||DK_0,\ E_{h(1||3||SK_0^2)}(SK_3))$ and $(4||DK_0,\ E_{h(1||4||SK_0^2)}(SK_4))$.

$NIP_{0,2}(x)$ is computed containing the following two points $(4||DK_0,\ E_{h(2||4||SK_0^2)}(SK_4))$ and $(5||DK_0,\ E_{h(2||5||SK_0^2)}(SK_5))$.

$NIP_{0,3}(x)$ is computed containing the point $(6||DK_0,\ E_{h(3||6||SK_0^2)}(SK_6))$.

$NIP_{0,4}(x)$ is computed containing the point $(7||DK_0,\ E_{h(4||7||SK_0^2)}(SK_7))$.

$NIP_{0,5}(x)$ is computed containing the point $(7||DK_0,\ E_{h(5||7||SK_0^2)}(SK_7))$.

The Newton's interpolating polynomials for the class C_1:

$NIP_{1,1}(x)$ is computed containing the following two points $(3||DK_1,\ E_{h(1||3||SK_1^2)}(SK_3))$ and $(4||DK_1,\ E_{h(1||4||SK_1^2)}(SK_4))$.

$NIP_{1,3}(x)$ is computed containing the point $(6||DK_1,\ E_{h(3||6||SK_1^2)}(SK_6))$.

$NIP_{1,4}(x)$ is computed containing the point $(7||DK_1,\ E_{h(4||7||SK_1^2)}(SK_7))$.

The Newton's interpolating polynomials for the class C_2:

$NIP_{2,2}(x)$ is computed containing the following two points $(4||DK_2,\ E_{h(2||4||SK_2^2)}(SK_4))$ and $(5||DK_2,\ E_{h(2||5||SK_2^2)}(SK_5))$.

$NIP_{2,4}(x)$ is computed containing the point $(7||DK_2,\ E_{h(4||7||SK_2^2)}(SK_7))$.

$NIP_{2,5}(x)$ is computed containing the point $(7||DK_2,\ E_{h(5||7||SK_2^2)}(SK_7))$.

The Newton's interpolating polynomial for the class C_3:

$NIP_{3,3}(x)$ is computed containing the point $(6||DK_3,\ E_{h(3||6||SK_3^2)}(SK_6))$.

The Newton's interpolating polynomial for the class C_4:

$NIP_{4,4}(x)$ is computed containing the point $(7||DK_4,\ E_{h(4||7||SK_4^2)}(SK_7))$.

The Newton's interpolating polynomial for the class C_5:

$NIP_{5,5}(x)$ is computed containing the point $(7||DK_5,\ E_{h(5||7||SK_5^2)}(SK_7))$.

3.5 Key Derivation Procedure

When a class, say, C_j, needs to compute the secret key of an another class, say, C_i, where C_i is a successor of C_j (i.e., $C_i \leq C_j$), C_j first finds a path from itself to the class C_i from the graph G. Figure 2 shows an example of a chain, where C_j wants to derive the secret key SK_i of the class C_i and there exists a path from C_j to C_i with some intermediate classes, say, C_{k_1}, C_{k_2}, ..., C_{k_l}. Here $C_i \leq C_{k_1} \leq C_{k_2} \leq \ldots \leq C_{k_l} \leq C_j$, where C_{k_r} is the immediate successor of $C_{k_{r+1}}$ for $r = 1, 2, \ldots, l-1$, and C_i and C_{k_l} are the immediate successors of C_{k_1} and C_j respectively.

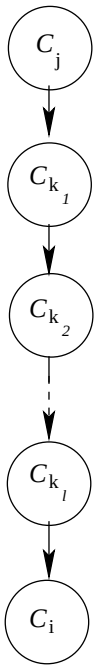

Figure 2: An example of a chain in a hierarchical structure

C_j computes the derivation key DK_j as

$$DK_j = g^{SK_j^3 \bmod R} \bmod P. \tag{1}$$

Using its secret key SK_j. C_j then computes SK_i as follows:

$$\begin{aligned} & NIP_{j,\,k_l}(i||DK_j) = E_{h(k_l||i||SK_j^2)}(SK_i) \\ \Rightarrow\quad & SK_i = D_{h(k_l||i||SK_j^2)}(NIP_{j,\,k_l}(i||DK_j)), \end{aligned} \tag{2}$$

where k_l is the identity of C_{k_l}, C_{k_l} the immediate predecessor of C_i and $NIP_{j,k_l}(x)$ stands for a Newton's interpolating polynomial for the class C_j at the points containing the identities and secret keys of all the immediate successors (including the class C_i) of C_{k_l}, and the identity k_l of C_{k_l}, and the secret key SK_j and the derivation key DK_j of C_j. $NIP_{j,\,k_l}(i||DK_j)$ is the value of the Newton's interpolating polynomial $NIP_{j,\,k_l}(x)$ at the x-coordinate $(i||DK_j)$. If the x-coordinate to the Newton's interpolating polynomial $NIP_{j,\,k_l}(x)$ is known, one gets the y-coordinate corresponding to the x-coordinate. For

an example, if we supply x-coordinate as $i||DK_j$, one gets y-coordinate as $E_{h(k_l||i||SK_j^2)}(SK_i)$ from Equation (2). It is noted that even if the derivation key DK_j of a class C_j is known to an adversary, it is computationally infeasible to compute the secret key SK_j of that class C_j. In order to derive SK_j^3, the adversary needs to solve the discrete logarithm problem over a large prime field $GF(P)$. The secret key SK_j of the class C_j is to be known by the adversary from $SK_j^3 \bmod R$, where $R = \frac{P-1}{2}$. Since R is product of two large prime factors, it is computationally difficult for the adversary to derive SK_j due to the integer factorization problem. Hence, we note that given DK_j, g and P to compute SK_i from the Equation (1) is based on both discrete logarithm as well as integer factorization problems.

Example 2. *Suppose the class C_0 wants to compute the secret key SK_7 of the class C_7 in Figure 1. At first C_0 supplies the x-coordinate as $7||DK_0$ to the Newton's interpolating polynomial $NIP_{0,4}(x)$ (or $NIP_{0,5}(x)$). Then C_0 derives $E_{h(4||7||SK_0^2)}(SK_7)$ (or $E_{h(5||7||SK_0^2)}(SK_7)$) and decrypts that value with the key $h(4||7||SK_0^2)$ (or $h(5||7||SK_0^2)$) to compute the secret key SK_7 corresponding to the class C_7.*

4 Dynamic Key Management

In this section, we present the dynamic key management problems like adding/deleting a class, adding/deleting a relationship and changing a secret key.

4.1 Adding a New Class

Let C_a be a new class to be added as an immediate successor of C_i into the existing system. Then, all the predecessors of C_i will also be the predecessors of C_a. CA does the following steps:

1) CA randomly chooses a secret key $SK_a \in \{0,1\}^L$.

2) CA computes derivation key $DK_a = g^{SK_a^3 \bmod R} \bmod P$.

3) If C_a is a leaf security class, CA constructs $NIP_{k,a}(x)$ for all C_k such that $C_i \leq C_k$ including the point $(a||DK_k, E_{h(i||a||SK_k^2)}(SK_a))$. Then, CA publishes the coefficients of every $NIP_{k,a}(x)$ corresponding to the class C_k.

4) Otherwise, if C_a is not a leaf security class, we proceed as follows. Let $C_j \leq C_a \leq C_i$, where C_a is an immediate successor and immediate predecessor of C_i and C_j respectively. CA constructs $NIP_{a,k}(x)$ for all C_k such that $C_k \leq C_a$ and publishes the coefficients of every $NIP_{a,k}(x)$ corresponding to the class C_a. CA reconstructs $NIP_{l,i}(x)$ for all C_l such that $C_i \leq C_l$ including one more point $(a||DK_l, E_{h(i||a||SK_l^2)}(SK_a))$ and publishes the coefficients of every $NIP_{l,i}(x)$ after deleting the old ones corresponding to the class C_l.

5) CA transmits securely SK_a to the class C_a.

4.2 Deleting a Class

Let C_d be a class to be deleted from the existing system. Then the following steps are required:

1) Let C_i be an immediate predecessor of C_d. CA reconstructs $NIP_{k,i}(x)$ for all C_k such that $C_i \leq C_k$ excluding the point $(d||DK_k, E_{h(i||d||SK_k^2)}(SK_d))$. Then, CA publishes the coefficients of every $NIP_{k,i}(x)$ after deleting the coefficients of old ones corresponding to the class C_k.

2) CA deletes all information corresponding to the class C_d.

4.3 Adding a Relationship

Suppose that a new relationship to be added between two different C_i and C_j such that $C_i \leq C_j$ holds, where C_i is an immediate successor of C_j. CA reconstructs $NIP_{k,i}(x)$ for all C_k such that $C_j \leq C_k$ including the point $(i||DK_k, E_{h(j||i||SK_k^2)}(SK_i))$ and then CA publishes the coefficients of every $NIP_{k,i}(x)$ corresponding to the class C_k.

4.4 Deleting a Relationship

Suppose that a relationship to be deleted between two different C_i and C_j with a relation $C_i \leq C_j$, where C_j is the immediate predecessor of C_i. CA reconstructs $NIP_{k,j}(x)$ for all C_k such that $C_j \leq C_k$ excluding the point $(i||DK_k, E_{h(j||i||SK_k^2)}(SK_i))$ and then publishes the coefficients of every $NIP_{k,j}(x)$ after deleting the coefficients of old ones corresponding to the class C_k.

4.5 Changing a Secret Key

Sometimes for security it is needed to change the secret key of a class. Suppose old secret key SK_i of the class C_i will be changed by a new secret key $SK_i' \in \{0,1\}^L$. CA then performs the following steps:

1) CA recomputes derivation key:

$$DK_i' = g^{(SK_i')^3 \bmod R} \bmod P.$$

2) Using new secret key SK_i' and derivation key DK_i' of C_i, CA reconstructs $NIP_{i,j}(x)$ for all C_j such that $C_j \leq C_i$ and publishes the coefficients of every $NIP_{i,j}(x)$ after deleting the old ones corresponding to the class C_i. Then, using the new secret key SK_i' of C_i, CA also reconstructs $NIP_{k,i}(x)$ for all C_k different from C_i such that $C_i \leq C_k$ and publishes the coefficients of every $NIP_{k,i}(x)$ after deleting the old ones corresponding to the class C_k.

3) CA securely transmits the secret key SK_i' to the class C_i.

5 Security Analysis

In this section, we present the security analysis of our scheme against different kinds of attacks from inside and outside of the system.

Contrary attack: Let us consider $C_i \leq C_j$. Let us verify whether SK_j can be calculated by a user being an adversary at level C_i through the secret key SK_i of its own and all public parameters. If C_k is the immediate predecessor of C_i and $C_k \leq C_j$, SK_i can be computed by C_j as $SK_i = D_{h(k||i||SK_j^2)}(NIP_{j,k}(i||DK_j))$. Since $DK_j = g^{SK_j^3 \bmod R} \bmod P$, SK_j can be computed from the equation $E_{h(k||i||SK_j^2)}(SK_i) = NIP_{j,k}(i||DK_j)$, which is based on the difficulty of computing the discrete logarithm problem over $GF(P)$ and the factoring problem to R even if DK_j is known to the adversary. Also, it is known that the problem of computing n-th root of $x^n \bmod R$ for any integer $n \geq 2$ is as difficult as factoring R, where R is product of two large primes and it has proved in [25] for the case of $n = 2$. As a result, even if DK_j is known to the adversary at level C_i, it is also difficult to compute the secret key SK_j of the class C_j because of the fact that it is computationally infeasible to compute SK_j due to the discrete logarithms and factorization problems. Further, finding roots of a polynomial over a large prime field by the adversary at level C_i may feasible due to results based on [2, 4, 10]. In our scheme, SK_i is encrypted using the encryption key $h(k||i||SK_j^2)$, where the computation of DK_j is computationally hard to the adversary at level C_i because of the fact that SK_j is not known to the adversary. As a result, even if DK_j is known to the adversary at level C_i, it is computationally hard to compute SK_j of the class C_j using root finding algorithms by the adversary at level C_i, which is already discussed previously in Subsection 3.5. The adversary can also try to compute the secret encryption key $h(k||i||SK_j^2)$. Therefore, the adversary has to compute DK_j and then the adversary has to solve the plaintext-ciphertext pair attacks against the symmetric cryptosystem, which is again difficult problem for insufficient number of plaintext-ciphertext pairs because in practical situations, the number of security classes is not more in order to derive the encryption key from plaintext-ciphertext pairs. Even if the encryption key is known to the adversary, it is also difficult to compute the secret key SK_j from $h(k||i||SK_j^2)$ because of the fact that it is computationally infeasible to invert the secure one-way hash function [29]. Since there are no efficient algorithms available so far for solving discrete logarithm problems, integer factorization problems and inversion of one-way hash functions, we conclude that our scheme is secure against such type of attack.

Collaborative attack: Let us check whether the decryption key of the upper level class can be derived by two or more lower security level classes. Let us consider C_j, C_k and C_l be the successors of C_i. Assume that C_j, C_k and C_l compromise their secret keys SK_j, SK_k and SK_l. We assume that C_x, C_y and C_z are the immediate predecessors of C_j, C_k and C_l respectively, where $C_x \leq C_i$, $C_y \leq C_i$ and $C_z \leq C_i$. We investigate whether SK_i can be calculated by C_j, C_k and C_l using their secret keys and public parameters. The equations known to them are as follows:

$$\begin{aligned} SK_j &= D_{h(x||j||SK_i^2)}(NIP_{i,x}(j||DK_i)), \\ SK_k &= D_{h(y||k||SK_i^2)}(NIP_{i,y}(k||DK_i)), \\ SK_l &= D_{h(z||l||SK_i^2)}(NIP_{i,z}(l||DK_i)), \end{aligned}$$

where $DK_i = g^{SK_i^3 \bmod R} \bmod P$. From these above equations, the derivation of SK_i is based on the difficulty of computing the discrete logarithms over $GF(P)$ and the factoring a large composite integer R as in *contrary attack*. Hence, it is computationally hard to compute secret key of a class for the collaboration of two or more lower security level classes. As a result, our scheme is secure against this kind of attack.

Interior collecting attack: Let us consider the subordinate class C_j which be accessible by m predecessors, say, $C_i, C_{i+1}, \ldots$, and C_{i+m-1}. Again, assume that the immediate predecessors of C_j be $\{C_k, C_{k+1}, \ldots, C_{k+m-1}\}$, where $C_{k+s} \leq C_{i+s}$ for all $s \in \{0, 1, \ldots, m-1\}$. Let us verify whether a user of C_j being an adversary can derive the secret key of one of its predecessors $C_i, C_{i+1}, \ldots$, and C_{i+m-1}. Assume that the following equations are known to the attacker.

$$\begin{aligned} SK_j &= D_{h(k||j||SK_i^2)}(NIP_{i,k}(j||DK_i)), \\ SK_j &= D_{h(k+1||j||SK_{i+1}^2)}(NIP_{i+1,k+1}(j||DK_{i+1})), \\ &\vdots \\ SK_j &= D_{h(k+m-1||j||SK_{i+m-1}^2)} \\ &\qquad (NIP_{i+m-1,k+m-1}(j||DK_{i+m-1})). \end{aligned}$$

It is also computationally hard as in *contrary attack* to compute the secret key of one of the classes $\{C_i, C_{i+1}, \ldots, C_{i+m-1}\}$ by the adversary. Hence, our scheme is secure against this attack.

Exterior attack: Assume that an intruder enters from outside the system, i.e., he/she is not an user of any class of the hierarchy. He/she being an adversary may try to compute the secret key SK_i of a class C_i using only the public parameters. The security of our scheme resists the unauthorized intruder. Because, even if DK_i and $h(j||k||SK_i^2)$ are known to the adversary, it computationally hard to compute SK_i, where k and j are the identities of the classes C_k and C_j respectively, and C_k is the immediate successor of C_j with $C_k \leq C_j \leq C_i$.

Sibling attack: Let us consider C_j and C_k be the siblings with same immediate predecessor C_i. Let us investigate whether C_j can compute SK_k of the class C_k or vice versa. Let a user of C_j being an adversary want to compute SK_k. C_j already knows the following equation:

$$SK_j = D_{h(i||j||SK_i^2)}(NIP_{i,\,i}(j||DK_i)).$$

If C_j wants to compute SK_k $(= D_{h(i||k||SK_i^2)}(NIP_{i,\,i}(j||DK_i)))$ using its secret key SK_j and all public parameters, C_j needs to compute SK_i first, which is computationally hard as in *contrary attack*. As a result, it is computationally hard to compute SK_k by the adversary without deriving SK_i. Hence, our scheme is secure against this attack.

Interior root finding attack: In this attack, a security class being an adversary has to compute the roots of a polynomial over a prime field $GF(Q)$, which is feasible due to [2, 4, 10]. Then, the adversary can try to compute the secret key of a class which is not a successor of the class. For an example, in Figure 1, C_2 can compute the secret keys SK_4, SK_5 and SK_7 of the classes C_4, C_5 and C_7 respectively. Then, C_2 can try to compute the secret key of any one of the classes $\{C_0, C_1, C_3, C_6\}$. However, Hus et al. [8] show that C_2 can compute the secret key SK_3 of the class C_3 in the Shen and Chen's scheme [30] for the same hierarchical structure as in Figure 1 after computing the secret key SK_4 of the class C_4 and then applying the root finding algorithm supplying SK_4 and the identity 3 of the class C_3 (more details can be found in [8]). Further, using the secret key SK_3, C_2 can also compute the secret key SK_6 of the class C_6 . Now, let us consider our scheme. Consider that C_i and C_j have a common successor C_k. Beside that common successor, let C_i and C_j have other successors. Let us check whether C_i can compute the secret key of any other successor of C_j which is not a successor of C_i, or whether C_j can compute the secret key of any successor of C_i which is not a successor of C_j. If it is true, these violate the hierarchy requirement. However, such type of attack is not possible in our scheme because of the fact that successors' secret keys are encrypted by the secret key of its predecessor to construct the Newton's interpolating polynomials corresponding to that predecessor. Following example shows that our scheme is secure against the attack in [8]. In Figure 1, C_1 and C_2 have a common successor C_4. C_1 has also another successor C_3, and C_2 has another successor C_5 and so on. Let us investigate whether C_2 being an adversary can compute the secret key SK_3 of C_3. As C_4 is a successor of C_2, C_2 can compute the secret key SK_4 of the class C_4. But, it is computationally hard for the adversary C_2 to compute the secret key SK_3 of the class C_3 from the public parameters and the secret key SK_4 of the class C_4 without knowing the secret key SK_1 of the class C_1 from the following equations

$$SK_3 = D_{h(1||3||SK_1^2)}(NIP_{1,1}(3||DK_1)),$$
$$SK_4 = D_{h(1||4||SK_1^2)}(NIP_{1,1}(4||DK_1)).$$

As a result, it is computationally hard for C_2 being an adversary to compute the secret key SK_3 of C_3. Thus, our scheme is secure against this attack, whereas such attack can be mounted on Shen and Chen's scheme (see in [8]).

Exterior root finding attack: In this attack, an adversary who is not a user in any class in a hierarchy can derive secret key of a class by root finding algorithm over a large prime field. Such type of attacks is shown in more details in [31]. All successors' secret keys of a class C_i are embedded in its public polynomial, say, $f_i(x)$, where C_i can compute the secret keys of its all successors. When CA adds or deletes some immediate successors from C_i, CA updates the public polynomial as $f_i'(x)$. But, for those successors, which remain as successors of C_i, their secrets are still computed by C_i using $f_i'(x)$. As a result, the adversary can try to compute x-coordinates of points which are used to construct the public polynomials by solving the equation $f_i(x) - f_i'(x) = 0$. Then, the adversary can try to compute the secret key of the successors of C_i (more details can be found in [31]). But, in our scheme, the adversary can compute the x-coordinates from the equation $NIP_{i,j}(x) - NIP'_{i,j}(x) = 0$ corresponding to the class C_i, where j is the identity of C_j with $C_j \leq C_i$. That is adversary can get $k||g^{SK_i^3 \bmod R} \bmod P$, where k is the identity of an immediate successor of C_j. From this value, it is computationally infeasible to compute SK_i. As a result, it is computationally hard to derive the secret key SK_k of the class C_k, which is an immediate successor of the class C_j, and $C_k \leq C_j \leq C_i$. Since SK_k is encrypted by the encryption key $h(j||k||SK_i^2)$, which is composed by the secret key SK_i of the class C_i, our scheme is secure against such type of attack. But, such type of attack can be possible for the Shen and Chen's scheme (see in [31]).

6 Efficiency of our Scheme

Storage space requirement: In our scheme, the secret parameter is SK_i for each class C_i, where $SK_i \in \{0,1\}^L$. Therefore, the storage requirement for storing the secret parameter is L bits. Let us consider C_i has k number of relations among all successors of C_i and the class C_i itself. Then, from the key generation procedure, CA publishes k number of public parameters (i.e., all coefficients of the Newton's interpolating polynomials) corresponding to the class C_i, where each public parameter lies between 1 and Q. Therefore, the storage requirement for storing

Table 1: Functional comparisons

Items ⇒ Schemes ⇓	*Public storage for a class with n successors,* and n' relations among these n successors and the class itself	*Secret storage for a class with n successors*	*key derivation complexity*
Akl-Taylor	$\Omega(n^3 \log_2 n)$ bits	$\lceil log_2 N \rceil$ bits	Exponentiation
Harn-Lin	$\Omega(n^3 \log_2 n)$ bits	$\lceil log_2 N \rceil$ bits	Exponentiation
Shen-Chen	$O(n\lceil \log_2 P \rceil)$ bits	$\Omega(n\lceil log_2 P \rceil)$ bits	2 Exponentiation + Interpolations
Our scheme	$O(n'\lceil \log_2 Q \rceil)$ bits	L bits	3 Multiplications + Exponentiation + Hash + Decryption + Interpolations

the public parameters is $k\lceil log_2 Q \rceil$ bits corresponding to the class C_i. In the Shen and Chen's scheme, $3\lceil \log_2 P \rceil + r\lceil \log_2 P'' \rceil$ bits are required to store the secret parameters for each class C_i, where r is the number of successors of C_i, P'' is a prime slightly larger than P. Since $L < 3\lceil \log_2 P \rceil + k\lceil \log_2 P'' \rceil$ and $L < \lceil \log_2 P \rceil$ because $L = 256$ and $\lceil \log_2 P \rceil \geq 512$ as P can be at least 512-bit for security on discrete logarithm problems, our scheme requires less amount of space to store secret parameters compared to the Shen and Chen's scheme.

Time requirement for deriving a key: Let $n+1$ be the number of successors of a class C_j, and C_i be a successor of C_j. In worst case, there is $n+1$ number of successors which may be the immediate successors of C_j, and as a result, the degree of the Newton's interpolating polynomial is n for the class C_j. Moreover, the evaluation of a n degree polynomial needs n number of modular multiplications and n modular additions. Thus, the time required to evaluate a polynomial of degree n at a point is $O(n \log_2^2 Q)$ in terms of bit operations, where the notation O (big oh) denotes upper bound. Further, the time required to compute the derivation key is $O(\log_2^3 P)$ bit operations because it is exponentiation operation on large modulus P. As a result, in our scheme, it takes $O(n \log_2^2 Q + \log_2^3 P)$ computational time in terms of bit operations to derive a secret key of lower security level class by an upper security level class after neglecting the computational time taken for multiplication, hashing and decrypting operations because of the fact that these operations take less computational time compared to the exponentiation operations on large modulus.

7 Comparison

In this section, we compare our scheme with the previously published schemes. Ω represents the lower bound.

Table 1 shows that the space requirement to store public parameters and secret parameters, and time taken to derive a key for different schemes. Let us assume that P (a large prime) and N (product of two large primes) be in the range between 1024-2048 bits for decent security and are of the same size, and $L = 256$. However, in the Shen and Chen's scheme, when hierarchy becomes quite large, the users in a higher security level classes need to store a large number of secret parameters. As a result, a large numfer of secret parameters becomes inconvenient to administer and hazardous to keep them secure. But, in our scheme, the size of secret parameter is always L bits, which does not depend on the size of the hierarchy. As a result, in our scheme, the size of secret parameter is much less than the Shen and Chen's scheme even if the hierarchy becomes large. Further, we observe from this table that our scheme requires three modular multiplication, one hashing, one modular exponentiation, computation of one interpolating polynomial, and one symmetric decryption operations. We know that cryptographic hashing and symmetric encryption/decryption are much more efficient than modular exponentiation for a large exponent compared to the computational point of view, whereas two modular exponentiation and computation of one interpolating polynomial are needed in the Shen and Chen's scheme. Since there is one more modulo exponentiation is needed in the Shen and Chen's scheme compared to our scheme to derive a secret key of a class, our scheme is more efficient than the Shen and Chen's scheme. Furthermore, sometimes the computation of interpolating polynomial in our scheme is less than that of the Shen and Chen's scheme. In the Shen and Chen's scheme, the Newton's interpolating polynomial for a class C_i consists of points corresponding all successors of C_i. There is only one interpolating polynomial corresponding to the class and the degree of the polynomial depends on the number of suc-

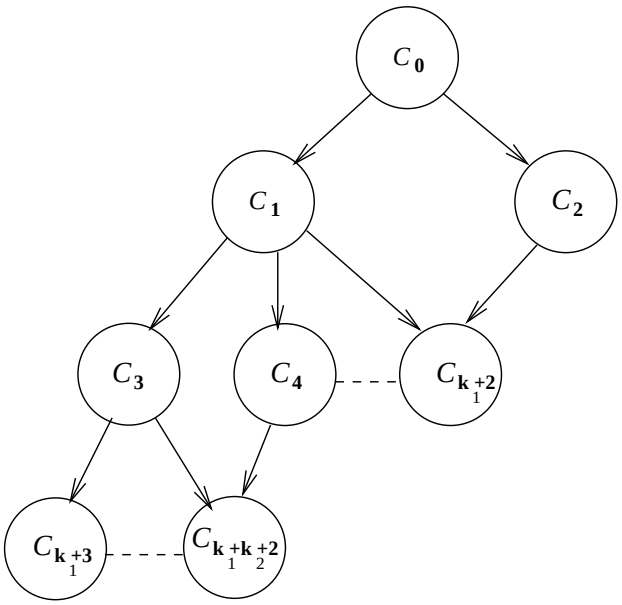

Figure 3: An example of poset ordered hierarchical structure

cessors of that class. If a class has n number of successors, the degree of polynomial is $n-1$ corresponding to that class. On the other hand, in our scheme, the number of the Newton's interpolating polynomial may be more than one corresponding to a class, which depends upon the number of non-leaf successors of that class plus one. For an example, in Figure 1, the number of the Newton's interpolation polynomials for the class C_0 is 6, because the number of non-leaf successors of C_0 is 5 plus 1. Further, the degree of the Newton's interpolating polynomial in our scheme is less then or equal to the degree that of the Shen and Chen's scheme corresponding to a class for computing the secret key of a successor of that class, which can be shown by the following example.

Example 3. *In Figure 3, C_0 has two immediate successors C_1 and C_2. C_1 has k_1 number of immediate successors, say, $C_3, C_4, \ldots, C_{k_1+2}$. Furthermore, C_3 has k_2 number of immediate successors, say, $C_{k_1+3}, C_{k_1+4}, \ldots, C_{k_1+k_2+2}$, and C_4 has an immediate successor $C_{k_1+k_2+2}$. Now, let C_1 want to compute the secret key of the class C_{k_1+3}. In the Shen and Chen's scheme, the total number of successors of C_1 is k_1+k_2. Therefore, the degree of the Newton's interpolation polynomial corresponding to the class C_1 is k_1+k_2-1. As a result, (k_1+k_2-1) modular multiplications and (k_1+k_2-1) modular additions are required to compute the secret key of C_{k_1+3} by C_1. But, in our scheme, to derive the secret key of the class C_{k_1+3}, C_1 needs the Newton's interpolation polynomial $NIP_{1,3}(x)$ which is of degree k_2-1. Thus, k_2-1 modular multiplications and k_2-1 modular additions are required for our scheme. Hence, for deriving the secret key of the class C_{k_1+3}, the degree of $NIP_{1,3}(x)$ in our scheme is less than the degree of the Newton's interpolating polynomial in Shen and Chen's scheme corresponding to the class C_1. Due to less number of modular multiplications and additions, our scheme requires less computational time for interpolation than that of the Shen and Chen's scheme. If we consider the class C_3 in Figure 3, the degree of the Newton's interpolating polynomial is k_2, which is same both in our scheme, and Shen and Chen's scheme. Hence, the degree of the Newton's interpolating polynomial in our scheme is less then or equal to the degree that of the Shen and Chen's scheme corresponding to a class for computing the secret key of a successor of that class. Further, when a user in a class wants to compute the secret key of its successor, he/she first chooses the appropriate Newton's interpolating polynomial so that degree of the polynomial is less.*

Hence, our scheme is more efficient than the Shen and Chen's scheme. Further, when hierarchy becomes quite large, Akl-Taylor's, and Harn-Lin's schemes are not applicable because of the fact that the size of public parameters will increase dramatically. Moreover, in Akl-Taylor's, and Harn-Lin's schemes, the key assignment technique encounters great difficulties in re-updating key. Finally, it is difficult to provide the user with a convenient way to change his/her secret key for the security considerations for these schemes. However, our scheme eliminates these difficulties.

8 Conclusion

In this paper, we have proposed a scheme for solving the multilevel key generation technique in poset ordered hierarchies. The security of our proposed scheme is based on the difficulties of simultaneously solving the strong collision resistant of secure one way hash functions, the discrete logarithms and the factoring a composite number, i. e. a mixture of multiple cryptographic difficulty problems, to enhance the security of hierarchical access control. Furthermore, our scheme is applicable to a large-scale hierarchical model. By comparing with the Shen and Chen's scheme, our proposed scheme needs less computational time to derive a key and provides better security. This scheme also supports the dynamic key management techniques. Hence, the proposed scheme is more efficient, flexible and secure.

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. 2, pp. 239-248, 1983.

[2] M. Ben-Or, "Probabilistic algorithms in finite fields," *22nd IEEE Anual Symposiam on Foundations of Computer Science (FOCS'81)*, pp. 394-398, 1981.

[3] C. C. Chang, R. J. Hwang, and T. C. Wu, "Cryptographic key assignment scheme for access control in a hierarchy," *Information Systems*, vol. 17, no. 3, pp. 243-247, 1992.

[4] H. Cohen, *A Course In computational Algebraic Number Theory*, Springer-Verleg, 1991.

[5] D. E. Denning, S. G. Akl, M. Morgenstern, P. G. Neumann, R. R. Schell, and M. Heckman, "Views for multilevel database security," *Proceeding of the IEEE Symposium on Security and Privacy*, pp. 156-172, Oakland, 1986.

[6] L. J. Fraim, "SCOMP: A solution to the multilevel security problem," *IEEE Computer*, vol. 16, no.7, pp. 26-34, 1983.

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

[8] C. L. Hus and T. S. Wu, "Crypanalysis and inprovements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy," *Computers and Security*, vol. 22, no. 5, pp. 453-456, 2003.

[9] M. S. Hwang, "An asymmetric cryptographic key assignment scheme for access control in totally-ordered hierarchies," *International Journal Computer Mathematics*, vol. 73, pp. 463-468, 2000.

[10] E. Keltofen and V. Shoup, "Subquadratic-time factoring of polynomials over finite fields," *Mathematics of Computations*, vol. 67, no. 223, pp. 1179-1197, 1998.

[11] B. LaMacchia, and A. M. Odlyzko, "Computation of discrete logarithms in finite fields," *advanced in Cryptology (CRYPTO'90)*, pp. 616-618, 1991.

[12] A. K. Lenstra and M. S. Manasse, "Factoring by electronic mail," *advanced in Cryptology (EUROCRYPT'89)*, pp. 355-371, 1990.

[13] H. T. Liaw and C. L. Lei, "An Optimal algorithm to assign cryptographic keys in a tree structure for access control," *BIT*, vol. 33, pp. 46-56, 1993.

[14] H. T. Liaw, S. J. Wang, and C. L. Lei, "An dynamic cryptographic key assignment scheme in a tree structure," *Computers and Mathematics with Applications*, vol. 25, no. 6, pp. 109-114, 1993.

[15] 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. 19, no. 4, pp. 457-462, 2003.

[16] W. P. Lu and M. K. Sundareshan, "Enhanced protocols for hierarchical encryption key management for secure communication in internet environments," *IEEE Transactions on Communications*, vol. 40, no. 4, pp. 658-660, 1992.

[17] 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.

[18] K. S. McCurley, "The discrete logarithm problem," *Proceedings of Symposia in Applied Mathematical Society*, vol. 42, pp. 49-74, 1990.

[19] J. McHugh, and A. P. Moore, "A security policy and formal top level specification for a multi-level secure local area network," *Proceeding of the IEEE Symposium on Security and Privacy*, pp. 34-39, 1986.

[20] National Institute of Standards and Technology, *Advanced Encryption Standard*, Federal Information Processing Standard (FIPS) 197, Nov. 26, 2001.

[21] National Institute of Standards and Technology, *Secure Hash Standard*, Federal Information Processing Standard (FIPS) 180-2, Aug. 2002.

[22] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographics significance," *Cryptology (EUROCRYPT'89)*, pp. 224-314, 1990.

[23] C. Pomerance, "Analysis and comparison of some integer factoring algorithms," *Computational Methods in Number Theory*, vol. 154, pp. 89-139, 1982.

[24] C. Pomerance, "Factoring," in *Proceedings of Symposia in Applied Mathematics*, vol. 42, pp. 27-48, 1990.

[25] M. O. Rabin, *Digitalized Signatures And Public-Key Functions As Intractable As Factorization*, Technical Report MIT/LCS/TR- 212, Laboratory for Computer Science, Massaachusetts Institute of Technology, Cambridge, Mass, 1979.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 637-647, 1978.

[27] R. S. Sandhu, "Cryptographic implimentation of a tree hierarchy for access control," *Information Processing Letters*, vol. 27, pp. 95-98, 1988.

[28] J. B. Scarborough, *Numerical Mathematical Analysis*, Oxford and IBH publishing Co. Pvt. Ltd., 1966.

[29] B. Schneier, *Applied Cryptography*, Second edition, John Wiley and Sons, New York, 1996.

[30] V. R. L. Shen and T. S. Chen, "A novel key management scheme based on discrete logarithms and polynomial interpolations," *Computers and Security*, vol. 21, no. 2, pp. 167-171, 2002.

[31] S-Y. Wang and C. S. Laih, "Crypanalysis of two key assignment schemes based on polynomial interpolations," *Computers and Security*, vol. 24, pp. 134-138, 2005.

[32] T. C. Wu and C. C. Chang, "Cryptographic key assignment scheme for hierarchical access control," *International Journal of Computer Systems Science and Engineering*, vol. 16, no. 1, pp. 25-28, 2001.

Debasis Giri received his M. Sc. degree in Mathematics from the Indian Institute of Technology, Kharagpur 721 302, India in 1998. He also received the M. Tech. degree in Computer Science and Data Processing from the Indian Institute of Technology, Kharagpur 721 302, India, in 2001. He is currently pursuing his Ph. D. degree in the Department of Mathematics from the Indian Institute of Technology, Kharagpur 721 302, India. Before joining the Ph. D. program, he worked as a lecturer in the Department of Computer Science and Engineering of Haldia Institute of Technology, West Bengal, India from March, 2001 to January, 2004. His current research interests include cryptography, network security, information security and e-commerce security.

Parmeshwary Dayal Srivastava received his M. Sc. degree in Mathematics from Kanpur University, Kanpur (U. P.), India in 1975 and Ph.D. in Mathematics from Indian institute of Technology, Kanpur (U. P.), India in 1980. Dr. Srivastava joined as Faculty in the department of Mathematics, I. I. T. Kharagpur (India) in May, 1980. During his 26 years of teaching, he taught various courses of pure & Applied Mathematics such as Real Analysis, Complex Analysis, Algebra, Measure theory, Numerical Analysis etc. to UG & PG students at IIT, Kharagpur. He has published more than 37 papers in a journal of International repute. He is referee of Indian Journal of Pure & Appl. Maths. (India); Demonstratio Mathematica (Warsa, Poland); Soochow J. Mathematics (China); Tamkang J. Mathematics (China); Bull. National Metallurgical Lab. (CSIR) Jamshedpur (India); ISTAM, IIT Kharagpur (India); J. Natural Sciences & Mathematics (Pakistan); Journal of Orissa Mathematical Society (India) and reviewer for Mathematical Review. Professor Srivastava is the life member of Indian Mathematical Society, Allahabad (India) & Indian Academy of Social Science, Allahabad (India). Presently, Dr. Srivastava is Professor of Mathematics at I. I. T. Kharagpur (India). His current research interests are Functional Analysis and Cryptography & Network Security.