A Novel Key Management Mechanism for Dynamic Hierarchical Access Control Based on Linear Polynomials

Vanga Odelu[1], Ashok Kumar Das[2], and Adrijit Goswami[3]

[1] Department of Mathematics
Rajiv Gandhi University of Knowledge Technologies, Hyderabad 500 032, India
odelu.vanga@gmail.com
[2] Center for Security, Theory and Algorithmic Research
International Institute of Information Technology, Hyderabad 500 032, India
iitkgp.akdas@gmail.com, ashok.das@iiit.ac.in
[3] Department of Mathematics
Indian Institute of Technology, Kharagpur 721 302, India
goswami@maths.iitkgp.ernet.in

Abstract. Several key management schemes for dynamic access control in a user hierarchy are proposed in the literature based on elliptic curve cryptosystem (ECC) and polynomial interpolation. Since the elliptic curve scalar multiplication and construction of interpolating polynomials are time-consuming operations, most of the proposed schemes require high storage and computational complexity. Further, most of the proposed schemes are vulnerable to different attacks including the man-in-the-middle attacks. In this paper, we propose a novel key management scheme for hierarchical access control based on linear polynomials only. We show that our scheme is secure against different attacks including the man-in-the-middle attack, which are required for an idle access control scheme. Moreover, the computational cost and the storage space are significantly reduced in our scheme while compared to the recently proposed related schemes.

1 Introduction

In a user hierarchy, the users and their own information items are divided into a group of disjoint security classes. Each user is then assigned to a security class. Let SC be a set of such N disjoint security classes, say $SC = \{SC_1, SC_2, \ldots, SC_N\}$ which forms a partially ordered set (poset, in short) with a binary relation "$\leq$". In a poset $\langle SC, \leq \rangle$, if SC_i and SC_j be two security classes with the relationship $SC_j \leq SC_i$, then the security level of SC_i is higher than or equal to that for SC_j. We call SC_i as predecessor of SC_j, and SC_j as successor of SC_i. We denote such a relationship by $(SC_i, SC_j) \in R_{i,j}$, which means that $SC_j \leq SC_i$. Hierarchical access control is an important research area in computer science, which has numerous applications including schools, military,

S.M. Thampi et al. (Eds.): SNDS 2012, CCIS 335, pp. 1–10, 2012.

governments, corporations, database management systems, computer network systems, e-medicine systems, etc.

In a hierarchical access control, a trusted central authority (CA) distributes keys to each security class in the hierarchy such that any predecessor of a successor class can easily derive its successor's secret key. Using that derived secret key, the predecessor class can decrypt the information encrypted by its successor. However, the reverse is not true in such access control, that is, no successor class of any predecessor will be able to derive the secret keys of its predecessors. Consider a simple example of a poset in a user hierarchy in Fig. 1. In this figure, we have the following relationships: $SC_2 \leq SC_1$, $SC_3 \leq SC_1$, $SC_4 \leq SC_1$, $SC_5 \leq SC_1$, $SC_6 \leq SC_1$, $SC_7 \leq SC_1$; $SC_5 \leq SC_2$; $SC_5 \leq SC_3$, $SC_6 \leq SC_3$; $SC_7 \leq SC_4$.

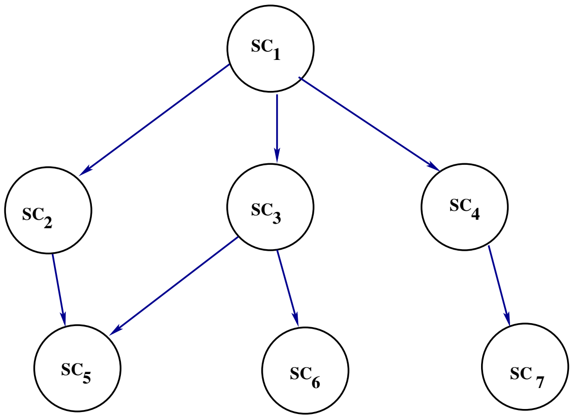

Fig. 1. An example of a poset in a user hierarchy

1.1 Related Work

Akl and Talor [2] first introduced the cryptographic key assignment scheme in an arbitrary poset hierarchy. Since then several different solutions to solve access control problem have been proposed in the literature. Chung et al. [5] proposed an efficient key management scheme for solving dynamic access control problem in a user hierarchy based on polynomial interpolation and elliptic curve cryptography (ECC). However, Das et al. in [6] showed that when a new security class is added into the hierarchy, any external attacker who is not a user in any security class can easily derive the secret key of a security class using the root finding algorithm. In order to withstand this security flaw found in Chung et al.'s scheme, they proposed an improved dynamic access control solution. Jeng-Wang's scheme [7] is based on ECC and it requires to regenerate keys for all the security classes when a security class is inserted into or removed from the existing hierarchy. Lin and Hsu [8] later showed that Jeng-Wang's scheme is insecure against a compromised attack in which the secret key of some security classes can be compromised by an attacker if some public information are modified. In order to remedy this security flaw, Lin and Hsu [8] proposed a key management scheme for dynamic hierarchical access control based on polynomial interpolation and ECC. However, their scheme requires high storage and computational complexity. Wu and Chen [13] proposed a key management scheme to solve dynamic access control problems in a user hierarchy based on hybrid cryptosystem in e-medicine system. Though their scheme improves computational efficiency over Nikooghadam et al.'s scheme [11], it still suffers from large storage space for public parameters in public domain and computational inefficiency due to costly elliptic curve point multiplication operations. Recently, Nikooghadam and Zakerolhosseini [10] showed that Wu-Chen's scheme is vulnerable to the man-in-the-middle attack. In order to remedy this security weakness in Wu-Chen's scheme, they further proposed a secure access control scheme using mobile agent, which is again based on ECC. However, their scheme requires huge computational cost for providing verification of public information in the public domain as their scheme uses ECC digital signature for verifying the public information by the security classes. Atallah et al. proposed a dynamic efficient access control scheme [3], [4] based on one-way hash functions. However, as pointed out in [9], their scheme is not suitable for a deep tree hierarchy or in a situation where a tree contains complex relationships.

1.2 Motivation

Symmetric key cryptosystem is more efficient than public key cryptosystem. Though several key management schemes for dynamic access control in a user hierarchy are proposed in the literature, most schemes are based on elliptic curve cryptosystem (ECC) and polynomial interpolation. Due to time-consuming operations of elliptic curve scalar multiplication and construction of interpolating polynomials, most of the proposed schemes require high storage and computational complexity. Moreover, majority of such schemes are vulnerable to different attacks including active attack called the man-in-the-middle attack. In this paper, we aim to propose a novel key management scheme for hierarchical access control based on linear polynomials. Our scheme does not require any polynomial interpolation and ECC operations. We make use of symmetric key cryptosystem along with efficient hash function so that our scheme will require minimum storage and computational complexity. We further show that our idle access control scheme is secure against different attacks including the man-in-the-middle attack.

1.3 Organization of the Paper

The rest of the paper is organized as follows. In Section 2, we describe our proposed scheme. In Section 3, we discuss dynamic access control problems of our scheme. Security analysis of our scheme is provided in Section 4. We compare the performance of our scheme with other related schemes in Section 5. Finally, we conclude the paper in Section 6.

2 Our Proposed Scheme

We assume that there are N security classes in the hierarchy which form a set $SC = \{SC_1, SC_2, \ldots, SC_N\}$. We use the following notations for describing our scheme. $H(\cdot)$ is a secure one-way hash function (for example, SHA-1 hash function [12]), Ω a symmetric key cryptosystem (for example, AES symmetric-key block cipher [1]), $E_k(\cdot)/D_k(\cdot)$ the symmetric-key encryption/decryption using key k, ID_{CA} the identity of CA, and $||$ the bit concatenation operator. Our scheme consists the following three phases, namely the relationship building phase, key generation phase, and key derivation phase.

2.1 Relationship Building Phase

CA builds the hierarchical structure for controlling access according to the given relationships among the security classes in the hierarchy. Assume that $SC_i \in SC$ and $SC_j \in SC$ be two security classes such that $SC_j \leq SC_i$, that is, SC_i has a higher security clearance than that for SC_j. We say that a legitimate relationship $(SC_i, SC_j) \in R_{i,j}$ between SC_i and SC_j exists if SC_i can access SC_j.

2.2 Key Generation Phase

CA executes the following steps in order to complete this phase:

Step 1. CA chooses a secure hash function $H(\cdot)$, a finite field $GF(m)$ with m is either odd prime or prime power, and a symmetric key cryptosystem Ω.

Step 2. CA randomly selects its own secret key k_{CA}. CA then selects randomly the secret key sk_i and sub-secret key d_i for each security class SC_i $(1 \leq i \leq N)$ in the hierarchy.

Step 3. For each security class SC_i, CA computes the signature $Sign_i$ on sk_i as $Sign_i = H(ID_{CA}||sk_i)$ for the purpose of signature verification of the secret key sk_i. CA then publicly declares them.

Step 4. For each SC_i such that $(SC_i, SC_j) \in R_{i,j}$, CA constructs the linear polynomials $f_{i,j}(x) = (x - H(ID_{CA}||Sign_j||d_i)) + sk_j \pmod{m}$, and declares them publicly.

Step 5. Finally, CA sends d_i to SC_i via a secure channel.

At the end of this phase, CA encrypts d_i of SC_i as $S_i = E_{k_{CA}}(d_i)$, computes its signature Sd_i as $Sd_i = H(ID_{CA}||d_i)$ for the signature verification of d_i and stores the pair (S_i, Sd_i) in the public domain. CA then deletes all the secret keys sk_i and d_i. Note that whenever CA wants to update the secret keys sk_i's, CA first obtains d_i's from public parameters S_i's by decrypting them with its secret key k_{CA} and then verifies signatures by calculating the hash values as $Sd_i' = H(ID_{CA}||d_i)$, and checks if $Sd_i' = Sd_i$. If it matches, CA confirms that derived secret key d_i is legitimate.

2.3 Key Derivation Phase

If the security class SC_i wants to derive the secret key sk_j of its successor SC_j with $(SC_i, SC_j) \in R_{i,j}$, SC_i needs to proceed the following steps:

Step 1. SC_i first computes the hash value $H(ID_{CA}||Sign_j||d_i)$ using its own sub-secret key d_i, signature $Sign_j$ and ID_{CA} publicly available in the public domain.

Step 2. SC_i obtains secret key sk_j of SC_j's (including SC_i) as $sk_j = f_{i,j}(H(ID_{CA} ||Sign_j||d_i))$. CA then verifies signature of sk_j as follows. CA computes $Sign_j^{'} = H(ID_{CA}||sk_j)$ and checks if $Sign_j^{'} = Sign_j$. If it holds, SC_i assures that the derived secret key sk_j is correct.

3 Solution to Dynamic Key Management

The solution to dynamic access problem in user hierarchy for our scheme such as adding a new security class into hierarchy, deleting an existing security class from the hierarchy, modifying the relationships among the security classes and updating secret keys are given below.

3.1 Adding a New Security Class

Suppose a security class SC_l with $SC_j \leq SC_l \leq SC_i$ be added into the hierarchy. CA needs the following steps to manage the accessibility of SC_l:

Step 1. CA randomly needs to select the secret key sk_l and the sub-secret key d_l for SC_l.

Step 2. For SC_l, CA needs to compute the signature $Sign_l$ on sk_l as $Sign_l =$ H(ID_{CA} $||sk_l$) for signature verification of sk_l and publicly declares it.

Step 3. For each SC_i such that $(SC_i, SC_l) \in R_{i,l}$ in the hierarchy, CA will construct the linear polynomials $f_{i,l}(x) = (x - H(ID_{CA}||Sign_l||d_i)) + sk_l \pmod m$, and declares them publicly.

Step 4. For each SC_j such that $(SC_l, SC_j) \in R_{l,j}$, CA will construct the linear polynomials $f_{l,j}(x) = (x - H(ID_{CA}||Sign_j||d_l)) + sk_j \pmod m$, and declares them publicly.

Step 5. CA finally sends d_l to SC_l via a secure channel.

At the end of this phase, CA encrypts d_l of SC_l as $S_l = E_{k_{CA}}(d_l)$, computes signature Sd_l as $Sd_l = H(ID_{CA}||d_l)$ for signature verification of d_l and stores the pair (S_l, Sd_l) in the public domain, and then deletes secret keys sk_l and d_l for security reasons.

3.2 Deleting an Existing Security Class

Suppose the security class SC_l with $SC_j \leq SC_l \leq SC_i$ be removed from the hierarchy. CA needs the following steps to remove SC_l so that the forward security is preserved.

Step 1. CA needs to remove all parameters corresponding to SC_l.

Step 2. After that CA renews secret keys sk_j's of successors SC_j's of SC_l as sk_j^*, and signatures $Sign_j$'s as $Sign_j^* = H(ID_{CA}||sk_j^*)$ and replaces $Sign_j$ with $Sign_j^*$ in the public domain.

Step 3. For each SC_i such that $SC_j \leq SC_i$ ($\neq SC_l$) in the hierarchy, CA constructs the linear polynomials $f_{i,j}^*(x) = (x - H(ID_{CA}||Sign_j^*||d_i)) + sk_j^* \pmod{m}$ and declares them publicly.

3.3 Creating a New Relationship

Assume that $SC_j \leq SC_i$ represents a new relationship between two immediate security classes SC_j and SC_i. Further, assume $SC_i \leq SC_l$ and $SC_y \leq SC_j$ (SC_y is not successor of SC_l before creating relationship). CA needs to compute linear polynomials $f_{l,y}(x) = (x - H(ID_{CA}||Sign_y||d_l)) + sk_y \pmod{m}$ and publicly declares them.

3.4 Revoking an Existing Relationship

Suppose the relationship between two immediate security classes SC_j and SC_i with $SC_j \leq SC_i$ be deleted from the hierarchy. Let $SC_j \leq SC_l$ ($\neq SC_i$) and $SC_y \leq SC_j$. CA then removes all parameters corresponding to the keys sk_y (including sk_j). CA also renews secret keys sk_y as sk_y^* and updates signatures $Sign_y$ as $Sign_y^* = H(ID_{CA}||sk_y^*)$ in the public domain. Finally, CA constructs public polynomials $f_{l,y}^*(x) = (x - H(ID_{CA}||Sign_y^*||d_l)) + sk_y^* \pmod{m}$.

3.5 Changing Secret Keys

Suppose we want to change the secret key sk_j of SC_j, where $SC_j \leq SC_i$. CA needs to renew the secret key sk_j as sk_j^* and update the signature $Sign_j$ as $Sign_j^* = H(ID_{CA}||sk_j^*)$, compute the corresponding polynomials $f_{i,j}^*(x) = (x - H(ID_{CA}||Sign_j^*||d_i)) + sk_j^* \pmod{m}$ and declare them publicly.

4 Security Analysis

In this section, we show that our scheme is secure against the following attacks.

4.1 Contrary Attack

Suppose $SC_j \leq SC_i$ and the successor class SC_j tries to derive the secret key sk_i of its predecessor class SC_i from the available public parameters $f_{i,i}(x) = (x - H(ID_{CA}||Sign_i||d_i)) + sk_i \pmod{m}$ and $f_{i,j}$'s. However, without knowledge of the sub-secret key d_i of SC_i, SC_j cannot compute $H(ID_{CA}||Sign_i||d_i)$ and as a result, the secret key sk_i. One important observation is that the pairs $(Sign_j, d_i)$ used in the construction of linear polynomials are distinct for two different polynomials. Even from the public parameter $S_i = E_{k_{CA}}(d_i)$, SC_j or any other user (except CA) cannot retrieve d_i without knowing CA's private key k_{CA}. Therefore, our scheme is secure against this attack.

4.2 Exterior Collecting Attack

This potential attack is from an external adversary. The question is that whether an external intruder can derive the secret key from lower level security classes through the accessible public parameters? However, to compute the secret key of a security class is computationally infeasible due to collision-resistant property of the one-way hash function $H(\cdot)$. Thus, no external intruder can retrieve the secret key of any security class. Our scheme is thus secure against such an attack.

4.3 Collaborative Attack

In this attack, several users in a hierarchy try to collaborate to launch an attack in order to compute their predecessor's secret key. Let SC_j and SC_l be two immediate successor classes of a predecessor class SC_i and they try to hack the secret key sk_i of SC_i. First, they can exchange secret keys with each other and derive the sub-secret key d_i of SC_i in order to derive the secret key sk_i of SC_i through the public linear polynomials $f_{i,j}(x) = (x - H(ID_{CA}||Sign_j||d_i)) + sk_j \pmod m$ and $f_{i,l}(x) = (x - H(ID_{CA}||Sign_l||d_i)) + sk_l \pmod m$. However, d_i is masked with one-way hash function $H(\cdot)$, and thus, determination of d_i is a computational infeasible problem due to hash function properties. Hence, no successor class can obtain the secret key of a predecessor class by collaborating each other and then our method is secure under this attack.

4.4 Equation Attack

Suppose a security class SC_j has common predecessors SC_i and SC_l, where SC_i does not have an accessibility relationship with SC_l. Let SC_i try to access the secret key sk_l of SC_l through the public linear polynomials $f_{l,j}(x) = (x - H(ID_{CA}||Sign_j||d_l)) + sk_j \pmod m$ and $f_{l,l}(x) = (x - H(ID_{CA}||Sign_l||d_l)) + sk_l \pmod m$. SC_i can compute $H(ID_{CA}||Sign_j||d_l)$ from $f_{l,j}(x)$ by using the derived secret key sk_j of SC_j, but SC_i cannot compute the sk_l from $f_{l,l}(x)$, since the hash values $H(ID_{CA}||Sign_j||d_l)$ and $H(ID_{CA}||Sign_l||d_l)$ are different. Therefore, the polynomials corresponding to one security class cannot be solvable by other security classes. As a result, our scheme is also secure against this attack.

4.5 Forward Security of Successors While Changing $SC_j \leq SC_k \leq SC_i$ to $SC_j \leq SC_i$

Assume that the relationship $SC_j \leq SC_k \leq SC_i$ is modified to another relationship $SC_j \leq SC_i$ after removing the security class SC_k from an existing hierarchy. Then CA not only deletes the accessibility relationship $SC_j \leq SC_k$, it also updates the accessibility-link relationship between SC_i and SC_j. CA further renews the secret keys sk_j's of SC_j's and the corresponding linear polynomials as $f^*_{i,j} = (x - H(ID_{CA}||Sign^*_j||d_i)) + sk^*_j \pmod m$. Since the hash values $H(ID_{CA}||Sign^*_j||d_i)$ can be computed only by the security class SC_i, the security class SC_k cannot hack the updated key sk^*_j of SC_j later. Therefore, the authority of SC_k over SC_j is terminated, and our scheme preserves the forward security property.

4.6 Man-in-the-Middle Attack

As in [10], we refer the "man-in-the-middle" attack as the masquerade attack. Suppose an attacker wants to be represented as an authorized central authority. Though the public domain is write-protected, we assume that the attacker can update somehow the information in the public domain. Let the attacker change the public linear polynomials $f_{i,j}(x)$'s in the public domain. The derivation of the secret key sk_j of a security class SC_j becomes a computationally infeasible problem since the sub-secret key d_j is only known to SC_j. As a result, the attacker does not have any ability to change properly the signatures $Sign_j = H(ID_{CA}||sk_j)$ and $Sd_j = H(ID_{CA}||d_j)$ in the public domain. Hence, our scheme protects against such an potential attack.

5 Performance Comparison with Other Schemes

Let T_{MUL}, T_{ADD} and T_{INV} denote the time complexity of executing modular multiplication, modular addition and modular inversion in $GF(2^{163})$, respectively. We denote $T_{EC_{MUL}}$ and $T_{EC_{ADD}}$ for time complexity of executing a point multiplication and a point addition in elliptic curve over $GF(2^{163})$. T_{SHA1} denotes the time complexity of hashing 512-bit message block using hash function, SHA-1 and T_{AES} for the time complexity of encrypting/decrypting 128-bit message block using AES with a 128-bit key.

From the analysis provided in Table 1 [13], it is noted that T_{INV}, $T_{EC_{MUL}}$, $T_{EC_{ADD}}$, T_{SHA1} and T_{AES} require approximately 3, 1200, 5, 0.36 and 0.15 field multiplications in $GF(2^{163})$, respectively, whereas T_{ADD} is negligible.

Table 1. Time complexity of various operations in terms of T_{MUL}

$T_{INV} \approx 3T_{MUL}$	$T_{EC_{MUL}} \approx 1,200T_{MUL}$
$T_{EC_{ADD}} \approx 5T_{MUL}$	$T_{SHA1} \approx 0.36T_{MUL}$
$T_{AES} \approx 0.15T_{MUL}$	T_{ADD} is negligible

We consider a hierarchy with N security classes SC_1, SC_2, ..., SC_N. Each security class SC_i has v_i predecessors. Comparison of storage complexity among various schemes is shown in Table 2. In our scheme, each key length is 128-bit since we have used AES algorithm. We see that the storage space of our scheme is reduced significantly compared with other schemes. In Table 3, we have compared the computational complexity and rough estimation in terms of field multiplications of our scheme with other schemes. In our scheme, key generation phase requires $NT_{SHA1} + \sum_{i=1}^{N}(v_i + 1)(T_{ADD} + T_{SHA1})$ and $N(T_{SHA1} + TAES)$ operations for computing signature, constructing linear polynomials and the pairs (S_i, Sd_i), whereas key derivation phase requires $\sum_{i=1}^{N}(v_i + 1)(T_{ADD} + T_{SHA1})$ operations. Thus, the total computational cost for our scheme is $\sum_{i=1}^{N}(v_i + 1)(2T_{ADD} + 3T_{SHA1}) + 3NT_{SHA1} + NT_{AES}$. It is also clear to observe the

Table 2. Comparison of storage space among various schemes

Schemes	CA's private domain	SC_i's private domain	Public domain
[7]	$163(2N+1)$	163	$163(\sum_{i=1}^{N}(v_i+1)+6N+2)$
[5]	$163(2N+1)$	163	$163(\sum_{i=1}^{N}(v_i+1)+6N+2)$
[11]	$163N$	163	$163(2\sum_{i=1}^{N}(v_i+1)+2N)$
[13]	$128+163$	163	$128(\sum_{i=1}^{N}(v_i+1)+N)+$ $163(2N+2)$
[10]	$163(N+1)$	163	$163(\sum_{i=1}^{N}v_i+(5N+2))$
[8]	163	163	$163(\sum_{i=1}^{N}v_i+3N+4)$
Ours	128	128	$128(\sum_{i=1}^{N}(v_i+1)+3N+1)$

Table 3. Comparison of computational costs among different schemes for key generation and key derivation phases

Scheme	Time complexity	Rough estimation
[7]	$\sum_{i=1}^{N}2(v_i^2+v_i).T_{MUL}+2N.T_{EC_{ADD}}$ $+(4N+2\sum_{i=1}^{N}(v_i+1)).T_{EC_{MUL}}$ $+2\sum_{i=1}^{N}(v_i+1).T_{SHA1}$	$(\sum_{i=1}^{N}(2v_i^2+2,402v_i)$ $+7,210N).T_{MUL}$
[5]	$\sum_{i=1}^{N}2(v_i^2+v_i).T_{MUL}+2N.T_{EC_{ADD}}$ $+(3N+2\sum_{i=1}^{N}(v_i+1)).T_{EC_{MUL}}+$ $2\sum_{i=1}^{N}(v_i+1).T_{SHA1}$	$(\sum_{i=1}^{N}(2v_i^2+2,402v_i)$ $+6,010N).T_{MUL}$
[11]	$N.T_{INV}+(N+2\sum_{i=1}^{N}(v_i+1))$ $.T_{EC_{MUL}}+(N+\sum_{i=1}^{N}(v_i+1)).T_{SHA1}$	$(\sum_{i=1}^{N}2,400v_i+3,603N).T_{MUL}$
[13]	$(2N+1).T_{EC_{MUL}}+$ $2(N+\sum_{i=1}^{N}(v_i+1)).T_{AES}+2N.T_{SHA1}$	$(\sum_{i=1}^{N}0.3v_i+2,401N$ $+1,200).T_{MUL}$
[10]	$(2\sum_{i=1}^{N}v_i).T_{XOR}+(N+\sum_{i=1}^{N}v_i)$ $.T_{ADD}+(2N+\sum_{i=1}^{N}v_i).T_{MUL}$ $+((2N+1)+4\sum_{i=1}^{N}v_i).T_{EC_{MUL}}$ $+(N+2\sum_{i=1}^{N}v_i).T_{SHA1}$	$(\sum_{i=1}^{N}4800.72v_i+2402.36N+$ $1200).T_{MUL}$
[8]	$N(3T_{EC_{MUL}}+2T_{MUL}+3T_{SHA1}+$ $T_{INV}+\sum_{i=1}^{N}v_i(T_{MUL}+2T_{SHA1}))+$ $v_iT_{MUL}+T_{SHA1}$	$(N\sum_{i=1}^{N}1.72v_i+v_i+$ $3606.08N+0.72)T_{MUL}$
Ours	$\sum_{i=1}^{N}(v_i+1)(2T_{ADD}+3T_{SHA1})+$ $3NT_{SHA1}+NT_{AES}$	$(\sum_{i=1}^{N}1.08v_i+1.95N)T_{MUL}$

the computational complexity of our scheme is reduced significantly compared to other schemes proposed recently. Further, our scheme, [8] and [10] are secure against possible attacks as compared to other schemes [5], [7], [11], [13]. However, [8] and [10] require very high storage and computational overheads compared to our scheme. Moreover, dynamic access control problems in our scheme are solved efficiently as compared to other schemes. Considering security and low storage and computational complexity, our scheme is significantly better than all other schemes [5], [7], [8], [10], [11], [13].

6 Conclusion

In this paper, we have proposed a novel efficient key management method to solve dynamic access control problems in a user hierarchy. We have utilized the linear polynomials along with symmetric-key cryptosystem to achieve the required goals for an idle access control scheme with low computational cost and small storage space. Further, our scheme is also secure against known attacks including the man-in-the-middle attack. Hence, our approach is more effective than previously proposed methods for practical applications.

References

1. Advanced Encryption Standard: FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce (November 2001), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
2. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems (TOCS) 1(3), 239–248 (1983)
3. Atallah, M., Blanton, M., Fazio, N., Frikken, K.: Dynamic and Efficient Key Management for Access Hierarchies. ACM Trans. Inf. Syst. Secur. 12(3), Article 18, 198–208 (2009)
4. Atallah, M., Frikken, K., Blanton, M.: Dynamic and efficient key management for access hierarchies. In: ACM Conference on Computer and Communications Security (CCS 2005), pp. 190–202 (2005)
5. Chung, Y.F., Lee, H.H., Lai, F., Chen, T.S.: Access control in user hierarchy based on elliptic curve cryptosystem. Information Sciences 178(1), 230–243 (2008)
6. Das, A.K., Paul, N.R., Tripathy, L.: Cryptanalysis and improvement of an access control in user hierarchy based on elliptic curve cryptosystem. Information Sciences 209, 80–92 (2012)
7. Jeng, F.G., Wang, C.M.: An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem. Journal of Systems and Software 79(8), 1161–1167 (2006)
8. Lin, Y.L., Hsu, C.L.: Secure key management scheme for dynamic hierarchical access control based on ECC. Journal of Systems and Software 84(4), 679–685 (2011)
9. Lo, J.W., Hwang, M.S., Liu, C.H.: An efficient key assignment scheme for access control in a large leaf class hierarchy. Information Sciences 181(4), 917–925 (2011)
10. Nikooghadam, M., Zakerolhosseini, A.: Secure Communication of Medical Information Using Mobile Agents. Journal of Medical Systems (2012), doi:10.1007/s10916-012-9857-8
11. Nikooghadam, M., Zakerolhosseini, A., Moghaddam, M.E.: Efficient utilization of elliptic curve cryptosystem for hierarchical access control. Journal of Systems and Software 83(10), 1917–1929 (2010)
12. Secure Hash Standard: FIPS PUB 180-1, National Institute of Standards and Technology (NIST), U.S. Department of Commerce (April 1995)
13. Wu, S., Chen, K.: An Efficient Key-Management Scheme for Hierarchical Access Control in E-Medicine System. Journal of Medical Systems (2011), doi:10.1007/s10916-011-9700-7