

Accepted Manuscript

Title: Efficient Utilization of Elliptic Curve Cryptosystem for Hierarchical Access Control

Authors: Morteza Nikooghadam, Ali Zakerolhosseini, Mohsen Ebrahimi Moghaddam



PII: S0164-1212(10)00135-4
DOI: doi:10.1016/j.jss.2010.05.072
Reference: JSS 8492

To appear in:

Received date: 18-3-2009
Revised date: 11-4-2010
Accepted date: 6-5-2010

Please cite this article as: Nikooghadam, M., Zakerolhosseini, A., Moghaddam, M.E., Efficient Utilization of Elliptic Curve Cryptosystem for Hierarchical Access Control, *The Journal of Systems and Software* (2008), doi:10.1016/j.jss.2010.05.072

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Efficient Utilization of Elliptic Curve Cryptosystem for Hierarchical Access Control

Morteza Nikooghadam, Ali Zakerolhosseini and
Mohsen Ebrahimi Moghaddam

Electrical and Computer Engineering Department,
Shahid Beheshti University; G.C, Tehran, Iran

Abstract

The Elliptic Curve Cryptosystem (ECC) has recently received significant attention by researchers due to its high performance, low computational cost, and small key size. In this paper, an efficient key management and derivation scheme based on ECC is proposed to solve dynamic access problems in a user hierarchy. Compared to previous ECC based works, the proposed method does not require constructing interpolate polynomials, therefore, the computational complexity of key generation and key derivation is significantly reduced. At the same time, time complexity of adding/deleting security classes, modifying their relationships, and changing of secret keys is decreased in the proposed method.

Keywords: Key management; Elliptic curve cryptosystems; Access control; User hierarchy; Security

1. Introduction

Generally, for distributed sharing environments such as computer networks and internet, users can be classified into a set of security classes $SC = \{SC_1, SC_2, \dots, SC_N\}$ depending on their security clearance. These security classes usually may be structured as a hierarchical tree.

In a secure distributed system with different security levels, the encrypted data are broadcasted into the network without concern of unallowable access from unintended recipients, because these recipients would be unable to decrypt the data. In such systems, an authorized member of the organization at certain level of the hierarchy can obtain the keys of his/her successors. Consequently, the member is able to access the encrypted data by means of

those keys. This form of access control is widely applied to different fields such as computer networks [10, 21], database management systems [13], and distributed operating systems [11, 18, 29, 12].

Assume the security classes are partially ordered by a binary relation “ \leq ”. In this partial order, $SC_j \leq SC_i$ denotes that users in SC_i are allowed to access the data items owned by users in SC_j where $i, j \in N$ while access in the opposite direction is prohibited. Fig. 1 shows a user hierarchy instance. In this figure, the relation $SC_3 \rightarrow SC_6$ denotes that SC_3 has a security clearance higher than that of SC_6 .

Generally, the relation $SC_j \leq SC_i$ means that SC_i is the predecessor of SC_j and SC_j is the successor of SC_i . Furthermore, if $SC_j \leq SC_i$ and any other security class SC_x , $x \in N$ does not exist such that $SC_j \leq SC_x \leq SC_i$, then SC_i is called an immediate predecessor of SC_j , and SC_j an immediate successor of SC_i . In a hierarchy, a certain secret key SK_i is assigned to each user in security clearance SC_i by a Certification Authority (CA). Therefore, each user has to use a symmetric cryptosystem to encrypt any data ‘ m ’ using its secret key SK_i before distributing along a network. Thus, only users in possession of SK_i are able to retrieve ‘ m ’. The main problem in this situation is that only the key SK_j is used for encrypting or decrypting the confidential data ‘ m ’ for security class SC_j . Therefore, when security classes SC_i (such that $SC_j \leq SC_i$) would like to retrieve data encrypted by SK_j , the security class SC_i should have the SK_j . The easiest way to achieve this is to hold all the keys of its successors. For example, as shown in Fig. 1, if the user in SC_1 needs to retrieve each data that encrypted under its successor secret keys, he/she has to save a set of secret keys $\{SK_2, SK_3, SK_4, SK_5, SK_6, SK_7\}$ in addition to its secret key ‘ SK_1 ’. Therefore, when the hierarchy is large and complex, the users with higher clearance are required to hold a large number of secret keys. Thus, it is difficult to manage keys for all users and to keep the system secure; also, this key retention is memory consuming and makes this approach inefficient. Therefore, an efficient mechanism is that each user only holds one specific secret key and uses this key and some public parameters to derive secret keys of all successors in order to retrieve the authorized data. Consequently, it is not possible to derive the secret key through the related secret or public parameters without authority.

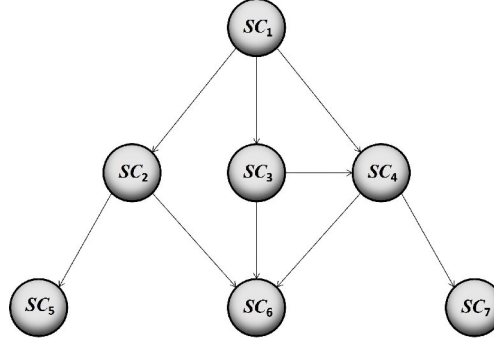


Fig. 1. A hierarchically-based structure.

Many key assignment schemes have already been proposed in the literature to provide solutions for the hierarchical access control problem [3, 4, 5]. Akl and Taylor proposed a top down solution for the partially-ordered multilevel key distribution problem using a cryptographic approach [25, 26]. The advantage of this scheme is that the key generation and key derivation algorithms are quite simple; however, it has some drawbacks such as public parameters require a large amount of storage when the number of security classes in the hierarchy is large, and adding a new node into the system after establishment is impossible. As another top down method, Mackinnon and Taylor [27] proposed an optimal key assignment algorithm. However, when the number of nodes becomes very large this method becomes impractical. These top down approaches are insecure in dynamic environment whenever a new security class is added into the system as an immediate successor of an existing security class; that is to say, all the issued keys should be re-generated [19].

To achieve dynamic ability, together with keeping the size of public information as small as possible, some other methods have been proposed [5, 2]. However, the main disadvantage of these methods is the iterative key derivation when a user with higher security clearance is going to derive a secret key of his successor which is not an immediate one. There are many other solutions for access control in user hierarchy; for instance, some approaches have been proposed that are based on discrete logarithm problem to generate and derive the secret keys of all classes [16, 4, 15, 30]; or models in [14, 20, 3] that use the integer factorization problem to complete the generation and derivation of key.

Recently, Chung et al. [33] and Jeng and Wang [9] proposed two efficient solutions for dynamic access problem in a user hierarchy based on elliptic curve cryptosystems. Elliptic curve cryptosystems have stronger mathematical base than the integer factorization and discrete logarithm systems [22, 32, 24]. Hence, ECC provides greater

efficiency than either integer factorization systems or discrete logarithm systems in terms of computational overheads, key sizes, and power consumption [28]. Therefore, the performance of the two mentioned methods is quite commendable in terms of security and efficiency in comparison with the previous works. In these methods, the CA determines a public elliptic curve polynomial for each security class. Therefore, each security class in key derivation phase uses the public elliptic curve polynomial of its successors to determine their secret key. By using ECC, performance of these schemes is significantly increased; nevertheless, using the polynomials in the structure of these methods imposes some computational overhead to the schemes.

In this paper, a method based on ECC is proposed that in contrast with the two mentioned schemes, does not require constructing interpolating polynomials. Therefore, not only computational complexity of key generation and key derivation is decreased significantly, also the cost of adding/deleting security classes, modifying their relationships and changing secret keys is reduced simultaneously.

The rest of the paper has been organized as follows: Section 2 briefly reviews Jeng and Wang key management scheme for hierarchical access control based on ECC with the emphasis on the performance efficiency, in Section 3 the proposed key generation and derivation algorithms is presented, Section 4 describes dynamic key management, Section 5 analyzes the resultant efficiency and security from the proposed scheme; and finally, conclusion is presented in Section 6.

2. Overview of Jeng and Wang's scheme

This section presents some background on elliptic curve cryptosystems and briefly describes the ECC scheme of Jeng and Wang.

2.1. Mathematical Background of Elliptic Curve Cryptosystems

In 1985, Koblitz [17] and Miller [31] independently proposed using the group of points on an elliptic curve defined over a finite field in discrete logarithm cryptosystems [23].

Finite fields are usually divided into prime fields or $GF(p)$ fields and binary extension fields or $GF(2^m)$ fields [38]. The $GF(2^m)$ fields as stated in [39, 31], are very attractive for hardware implementation due to their "carry free" arithmetic. Another advantage of $GF(2^m)$ is the availability of different equivalent representations of field elements, e.g., polynomial, normal, or dual bases [39, 40].

Let $GF(2^m)$ be a finite field of 2^m elements, where m is an integer. An elliptic curve over $GF(2^m)$ is defined as (1):

$$y^2 + xy = x^3 + ax^2 + b \quad \text{with } a, b \in GF(2^m), b \neq 0 \quad (1)$$

An elliptic curve over $GF(2^m)$ consists of all points (x, y) ($x, y \in GF(2^m)$) such that it satisfies equation (1) together with the point of infinite O . The addition of two points and doubling a point on an elliptic curve (generally over a set of real numbers) in a geometrical space are illustrated in Figs. 2 and 3, respectively [7].

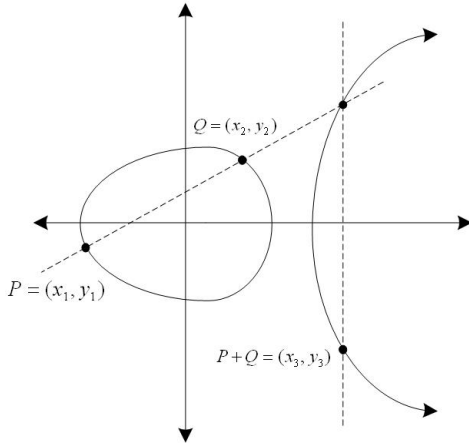


Fig. 2: Addition on elliptic curves

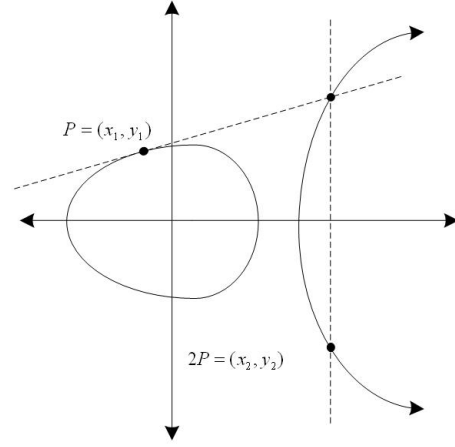


Fig. 3: Doubling a point

Considering an elliptic curve E_q on $GF(2^m)$, the Group Law is supported by following terms [7]:

1. Identity: $P + O = O + P = P$ for all $P = (x, y) \in E_q$.
2. Negativity: Let $P = (x, y) \in E_q$ and $Q = (x, x + y) \in E_q$ therefore $P + Q = O$, that is to say, negative of P is Q .
3. Point addition: If $P = (x_1, y_1) \in E_q$ and $Q = (x_2, y_2) \in E_q$ such that $P \neq \pm Q$, then $P + Q = (x_3, y_3)$ is defined as (2):

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a & \text{with} & & \lambda &= \frac{y_2 + y_1}{x_1 + x_2} \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \end{aligned} \quad (2)$$

4. Point doubling: If $P = (x_1, y_1) \in E_q$ and $P \neq -P$, then $2P = (x_2, y_2)$ is defined as (3):

$$\begin{aligned} x_2 &= \lambda^2 + \lambda + a \\ y_2 &= \lambda(x_1 + x_2) + x_2 + y_1 \end{aligned} \quad \text{with} \quad \lambda = x_1 + \frac{y_1}{x_1} \quad (3)$$

Scalar Multiplication as a fundamental operation in ECC is obtained by performing the elliptic curve addition operation k times [1, 7]:

$$Q = kP = \underbrace{P + P + \dots + P}_k \quad (4)$$

Calculating Q is relatively easy when k and P are given, but it is a hard problem to determine k when Q and P are specified. This problem is called the Elliptic Curve Discrete Logarithm Problem (ECDLP). Thus, the scalar multiplication on elliptic curves over finite fields is considered as a one-way function which is useful in cryptographic applications [37].

2.2. Jeng and Wang's Key Management Scheme Based on ECC

In Jeng and Wang's scheme [9], each security class in the hierarchy is allowed to select its own secret key (SK_i) and a secret parameter (n_i) randomly. Consequently, each class has to generate its corresponding public parameter $P_i = n_i G$ where G is a base point on a specific elliptic curve determined by the CA. Finally, each class sends both of its secret key SK_i and secret parameter n_i to the CA via a secret channel. Therefore, the keys have to be encrypted before being sent to the CA. In this procedure, first the CA selects a secret parameter n_{ca} randomly and makes $P_{ca} = n_{ca} G$ and $f(x, y)$ public where $f(x, y)$ is an algorithm to represent a point on the elliptic curve as a real number. Then, each class forms its secret data as a point (SK_i, n_i) and chooses a random positive integer k_i and produces a pair of points, $\{k_i G, (SK_i, n_i) + k_i P_{ca}\}$; consequently, the class sends this pair of points to the CA. The CA multiplies first point by its own secret parameter n_{ca} and subtracts the result from second point to derive (SK_i, n_i) as follows:

$$(SK_i, n_i) + k_i P_{ca} - n_{ca}(k_i G) = (SK_i, n_i) + k_i(n_{ca} G) - n_{ca}(k_i G) = (SK_i, n_i) \quad (5)$$

In other words, each class masks (SK_i, n_i) by adding $k_i P_{ca}$ to it and only the CA can remove the mask. Then, for each security class SC_i , $1 \leq i \leq n$, the CA constructs a polynomial $H_i(x)$ and declares these polynomials publicly. Each polynomial ($H_i(x)$) is demonstrated as the following equation:

$$H_i(x) = \prod_t (x - f(n_i P_t)) + SK_i \quad \text{for all } SC_i \leq SC_t \quad (6)$$

Now, assume the user with security class SC_i is going to access the encrypted data held by a user in one of his successor classes SC_j . Therefore, he/she has to get the public parameters $H_j(x)$ and P_j of SC_j and then compute $H_j(f(n_i P_j))$ to obtain SK_j .

There is a little difference between the proposed method in [33] and the Jeng and Wang's scheme. The difference is that in [9], each security class selects its own secret key and sends the secret key to the CA via a secure way, but in [33], the CA selects all the secret parameters and sends them to the corresponding security class in the same secure way. Nonetheless, the proposed scheme in [33] uses the same public polynomials in key generation and derivation phase. As mentioned in the previous sections, requirement of the storage and the computational overhead for constructing the interpolating polynomials are tremendous. Based on [6], the cost of constructing an interpolating polynomial of degree m is exactly m addition, $2m^2 + 2$ subtraction, $2m^2 + m - 1$ multiplication and $m + 1$ division. Also, computational complexity of constructing each interpolating polynomial by applying Fast Fourier Transform is $O(m(\log m)^2)$. Therefore, it is obvious that any new key management scheme for hierarchical access control based on ECC that does not require these polynomials in its structure, has much higher performance.

3. Proposed Scheme

In the proposed method, a CA is considered that builds a hierarchical structure for access control according to the relationships between the security classes. Also, it is supposed that there are N security classes which form a set $SC = \{SC_1, SC_2, \dots, SC_N\}$. In addition, SC_i is considered as a security class with higher clearance than SC_j , therefore SC_j is accessible by SC_i , and this relationship is represented as $SC_j \leq SC_i$.

The proposed scheme consists of two phases: key generation and derivation. The details of these phases have been described in the following sub-sections:

3.1. Key Generation Phase

This phase includes the following steps:

Step 1: The CA determines a field size q which defines the underlying finite field F_q , where either $q = p$ in case that p is an odd prime, or $q = 2^m$ that q is a prime power.

Step 2: The CA specifies an appropriate elliptic curve by selecting two parameters a and b of elliptic curve equation E over F_q : $y^2 + xy = x^3 + ax^2 + b$. Then, the CA determines the base point G that is a finite point on elliptic curve having the largest order n such that $nG = O$. The values of E_q , G and n are declared publicly by the CA.

Step 3: The CA selects a one way hash function $f(x)$ to transform a point on elliptic curve E_q into a number v that $v \in F_q$ and makes $f(x)$ public.

Step 4: Each security class SC_i selects a random integer d_i from the interval $[1, n-1]$ as its secret parameter, then computes the point $P_i = d_i G$ as its public parameter and declares it publicly.

Step 5: The CA selects a random integer k_i from the interval $[1, n-1]$ for each security class SC_i and computes $Z_i = k_i G$. Therefore, the secret key for each security class is $SK_i = f(Z_i)$.

Step 6: For each SC_j , $1 \leq j \leq N$

For all security classes SC_i that satisfies $SC_j \leq SC_i$

The CA determines the point $M_{i,j} = k_j(P_i)$ and declares it publicly.

3.2. Key derivation phase

For the relationship $SC_j \leq SC_i$, the security class SC_i determines secret keys of all its successors (SK_j) and its secret key (SK_i) as follows:

Step 1: Determines d_i^{-1} then compute $Z_j = k_j G = d_i^{-1} M_{i,j}$; note that d_i^{-1} indicates the inversion in the finite field, which is one of the required operations in elliptic curve digital signature algorithm [1]. An efficient procedure for execution this operation has been offered in [1].

Step 2: Determines $SK_j = f(Z_j)$.

Example. Fig.1 illustrates a simple hierarchy including some security classes with certain relationship. As shown in this figure, the user set has seven security classes denoted as $SC = \{SC_1, SC_2, SC_3, SC_4, SC_5, SC_6, SC_7\}$. The CA

determines the public parameters $M_{i,j}$ and declares them publicly. The set of required public parameters for deriving the secret key of each security class by her/his predecessors or owner of the secret key is shown as follows:

$$SC_1: \{M_{1,1} = k_1(P_1)\}$$

$$SC_2: \{M_{1,2} = k_2(P_1), M_{2,2} = k_2(P_2)\}$$

$$SC_3: \{M_{1,3} = k_3(P_1), M_{3,3} = k_3(P_3)\}$$

$$SC_4: \{M_{1,4} = k_4(P_1), M_{3,4} = k_4(P_3), M_{4,4} = k_4(P_4)\}$$

$$SC_5: \{M_{1,5} = k_5(P_1), M_{2,5} = k_5(P_2), M_{5,5} = k_5(P_5)\}$$

$$SC_6: \{M_{1,6} = k_6(P_1), M_{2,6} = k_6(P_2), M_{3,6} = k_6(P_3), M_{4,6} = k_6(P_4), M_{6,6} = k_6(P_6)\}$$

$$SC_7: \{M_{1,7} = k_7(P_1), M_{3,7} = k_7(P_3), M_{4,7} = k_7(P_4), M_{7,7} = k_7(P_7)\}$$

Therefore, each security class can derive its secret key and secret keys of his successors, as follows:

$$SK_1 = f(Z_1) = f(d_1^{-1}M_{1,1}), \text{ since there is no predecessor for } SC_1, \text{ no one can derive } SK_1 \text{ except } SC_1$$

$$SK_2 = f(Z_2) = f(d_1^{-1}M_{1,2}) = f(d_2^{-1}M_{2,2}), \text{ it means that only } SC_2 \text{ and its predecessor, } SC_1, \text{ can derive } SK_2.$$

$$SK_3 = f(Z_3) = f(d_1^{-1}M_{1,3}) = f(d_3^{-1}M_{3,3})$$

$$SK_4 = f(Z_4) = f(d_1^{-1}M_{1,4}) = f(d_3^{-1}M_{3,4}) = f(d_4^{-1}M_{4,4})$$

$$SK_5 = f(Z_5) = f(d_1^{-1}M_{1,5}) = f(d_2^{-1}M_{2,5}) = f(d_5^{-1}M_{5,5})$$

$$SK_6 = f(Z_6) = f(d_1^{-1}M_{1,6}) = f(d_2^{-1}M_{2,6}) = f(d_3^{-1}M_{3,6}) = f(d_4^{-1}M_{4,6}) = f(d_6^{-1}M_{6,6})$$

$$SK_7 = f(Z_7) = f(d_1^{-1}M_{1,7}) = f(d_4^{-1}M_{4,7}) = f(d_7^{-1}M_{7,7})$$

In order to clarify the above example, the various phases of the proposed scheme are summarized in Fig. 5, 6 and 7, respectively. Fig. 4 indicates the definition of symbols in the following figures.






Private domain whose its contents are kept secret by their owner	
Public domain whose data anyone can access	
Proper computation is performed	
Security classes	
Secret parameter is selected	

Fig. 4: Symbols that used in the following figures.

As shown in Fig. 5, initially, each security class and the CA select the required secret parameters and compute the corresponding public parameters. All the selected secret parameters are kept secret in the related private domain. The public parameters P_i are recorded in the public domain and are accessible by all the users. Afterward, the CA selects a secret parameter k_i and computes resultant parameter $Z_i = k_i G$ which corresponds to each security class SC_i . Then, the CA determines the secret key SK_i for each security class SC_i . It is worth mentioning that all the stated parameters are kept secret by the CA and maintained in its private domain.

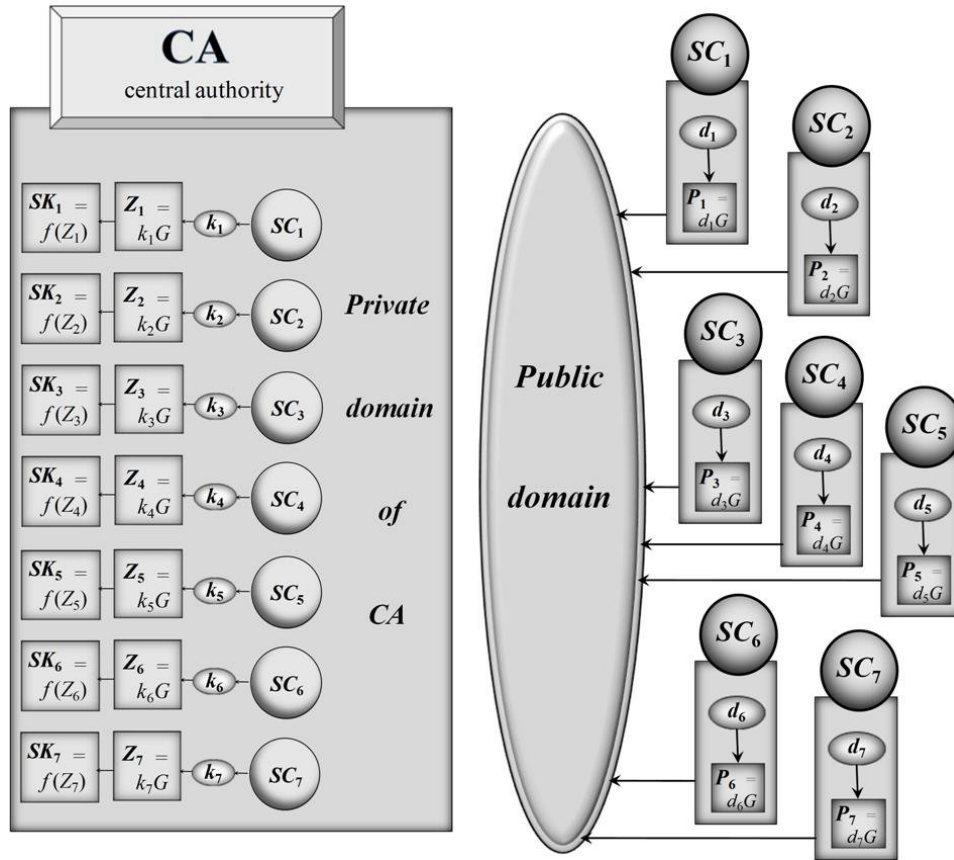


Fig. 5: public and private domains of security classes and the CA in the proposed method for example 1 (Fig 1)

In the proposed scheme, derivation process is performed directly instead of iteratively. For example, if security class SC_1 in Fig. 1 requires deriving the secret key SK_6 , iterative process for going through intermediate nodes is not required. In the other words, it is not required for SC_1 to derive secret keys of nodes SC_2 , SC_3 , or SC_4 and then

derives secret key of SC_6 . That is to say, key derivation process is quite independent of intermediate nodes when a predecessor is going to reach secret key of his/her successors that is not immediate ones. Fig. 6 shows the next phase of the proposed scheme for the mentioned example.

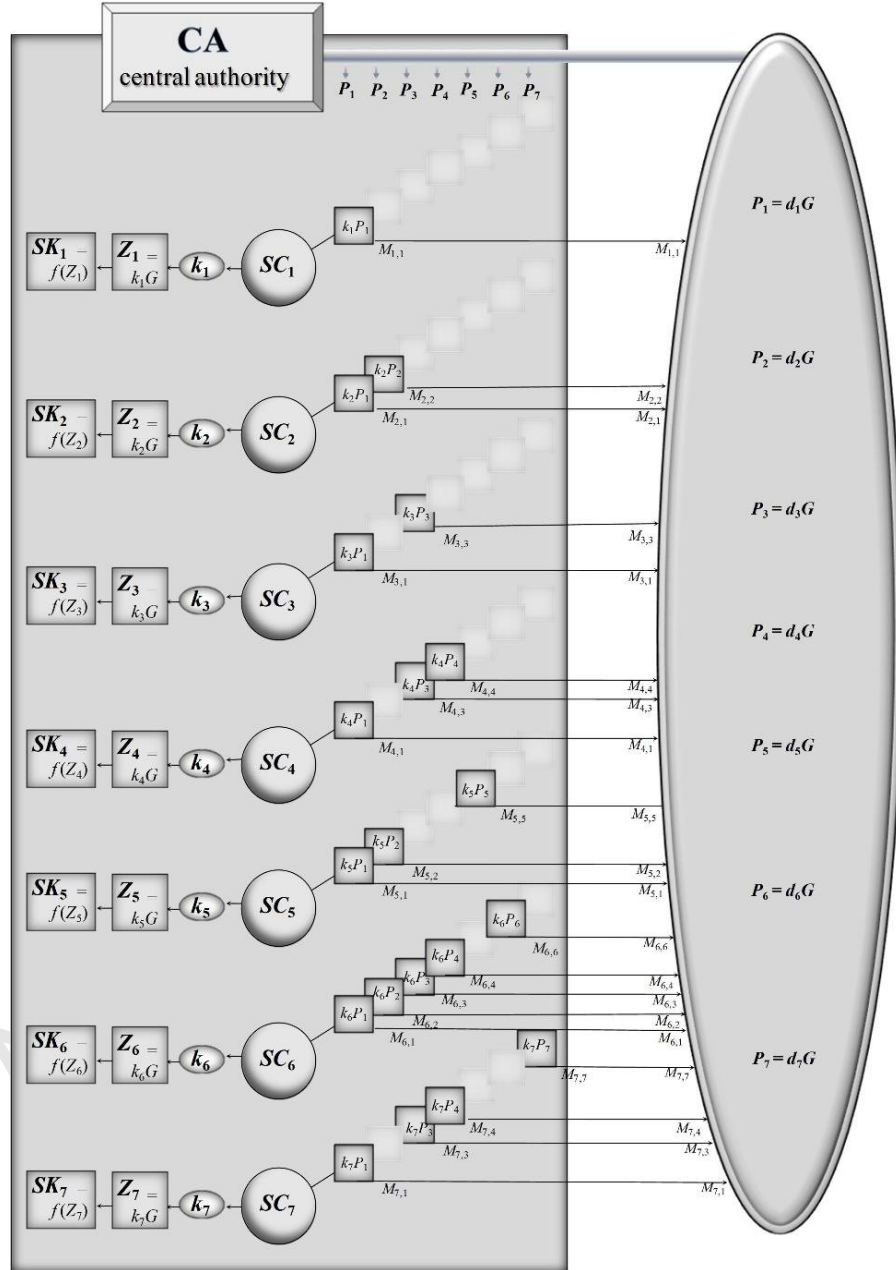


Fig. 6: Generation of public parameters by the CA, based on authority of each security class and presenting them in the public domain when proposed method applied on Fig.1.

As illustrated in Fig. 6, the CA takes the values of P_1 to P_7 from the public domain and uses them together with values of k_1 to k_7 inside its private domain to compute some public parameters based on the predecessors of each node. For instance, regarding to Fig. 1, the security classes that have authority to access secret key of SK_7 are SC_1 , SC_3 , SC_4 and SC_7 therefore, the CA computes proper public parameters $M_{1,7}$, $M_{3,7}$, $M_{4,7}$ and $M_{7,7}$, respectively by using corresponding public parameters P_1 , P_3 , P_4 and P_7 , and the secret parameter k_7 that is assigned to SC_7 . These parameters are declared publicly by transmitting them to the public domain. Similarly, based on the relationship in Fig. 1, other required parameters $M_{i,j}$ are computed simultaneously and transmitted to the public domain in order to use by security classes to derive authorized secret keys at the next phase. Finally, it is desired that by utilizing public parameters in public domain, each security class SC_i having its secret parameter d_i is able to derive its secret key and secret key of his/her successors directly. Fig. 7 shows required operations that each node should perform in order to obtain all secret keys.

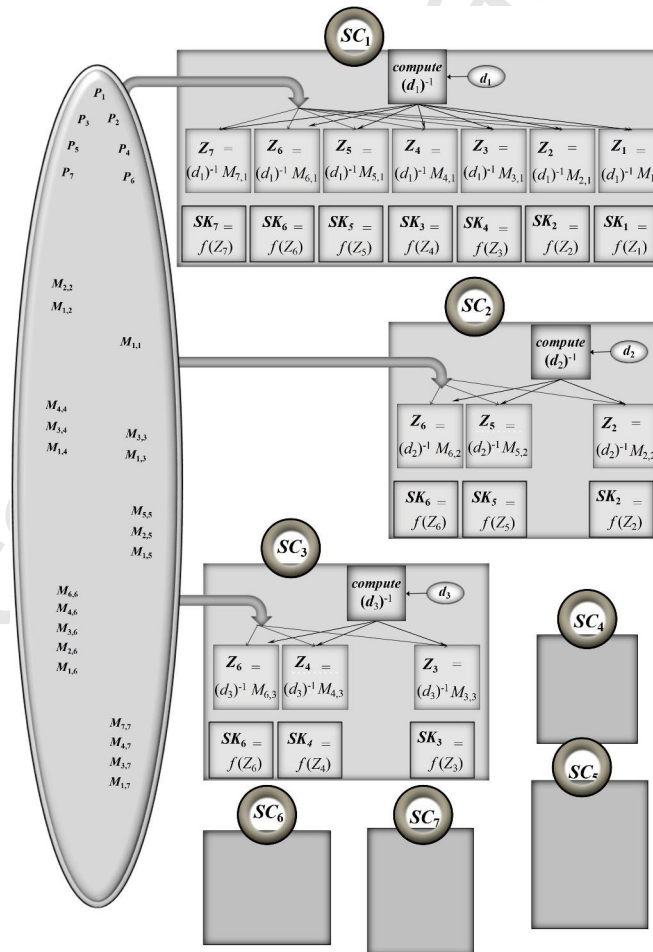


Fig. 7: Each of security classes can compute all the secret keys that have authority to access them in fig.1.

For instance, regarding to Fig. 1, the security class SC_2 has authority to access the secret keys SK_6 , SK_5 and SK_2 , therefore, the security class SC_2 takes public parameters $M_{2,6}$, $M_{2,5}$ and $M_{2,2}$ from public domain and computes $(d_2)^{-1}$ to use in next required computations in private domain of SC_2 . It is considerable that the security class SC_2 keeps secret the value of $(d_2)^{-1}$ same as d_2 . As illustrated in Fig. 7, using the mentioned parameters, the security class SC_2 can compute secret keys SK_6 , SK_5 , and SK_2 . Similarly, based on the relationships in Fig. 1, other nodes can compute authorized secret keys.

Evaluation of the proposed scheme using example: Considering the hierarchy in Fig. 1, the CA should only save the value of secret parameters k_1 to k_7 , therefore, this makes the computation of Z_1 to Z_7 and SK_1 to SK_7 possible whenever it is required. Required storage space for public parameters is large and it is assumed that these parameters are maintained in an area generally named “public domain”. Since any one is authorized to access its content and no securitization strategy is required, there is no concern about their maintenance. Nevertheless, increase of security classes especially in the underside of hierarchy leads to multiple rising of these public parameters and increasing the required storage space. Also, physically distribution of these parameters affects the performance and is an important factor in implementation.

Each of security classes, SC_1 to SC_7 , should save only the value of its secret parameter, d_1 to d_7 , respectively. Therefore computation of all authorized secret keys will be possible using the maintained secret parameter d_i and required public parameters $M_{i,j}$ that may be taken from public domain. In this approach, the main concern of each security class is hiding only one secret parameter from others. Fig. 8 shows all parameters that have to be kept secret by the CA and security classes. With respect to other parameters in public domain, only a plain storage space is required without any security policies.

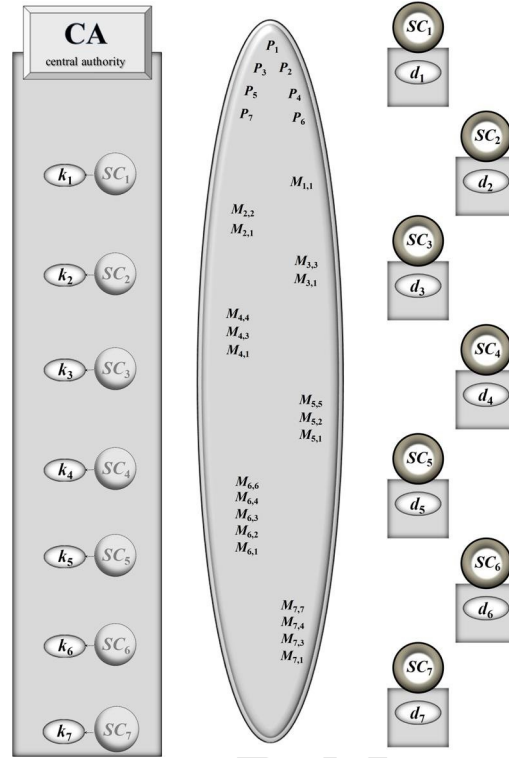


Fig. 8: Required storage space of the proposed scheme when applied on the stated example.

Let a simple primitive scheme that all secret keys SK_i are determined by the CA and declared to authorize security classes by an arbitrary secure way. Assume each security class has to maintain all the secret keys of its successors. Fig. 9 illustrates required storage space of such scheme for the stated example. In contrast with the proposed scheme, each security class has to maintain large number of parameters secret. Clearly, increase of security classes especially at the bottom of the hierarchy leads to increase in maintained parameters by each of security classes.

Although the proposed scheme needs similar number of data as public parameters in public domain, but against scheme in Fig. 9, confidential policies are not required for their privacy. Another main drawback of this simple scheme is problems of dynamic key management such as adding a new security class, removing an existing security class, creating a new relationship, revoking an existing relationship, and changing secret keys. The other considerable cost in the scheme shown in Fig. 9 is applying a secure low cost method to transmit secret keys between security classes and the CA. Clearly, any employed solution increases the cost of illustrated scheme in Fig. 9 in comparison with proposed scheme.

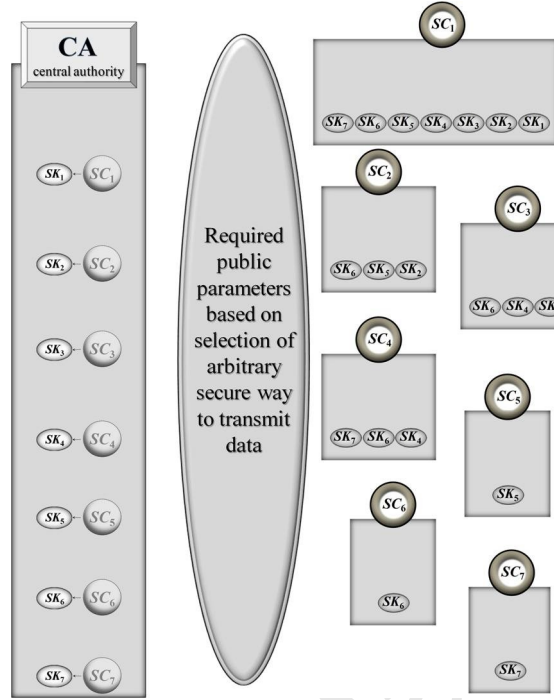


Fig. 9: Required private storage space when a simple primitive scheme is applied on the stated example.

4. Solution to Key Management of Dynamic Access Problems

Now, we consider the problems of dynamic key management such as adding a new security class, removing an existing security class, creating a new relationship, revoking an existing relationship, and changing secret keys.

4.1. Adding New Security Classes

Suppose a new security class SC_x is added into an existing hierarchy such that $SC_j \leq SC_x \leq SC_i$. Accessing priority of SC_x in the hierarchy is managed by following procedure:

Step 1: the security class SC_x selects a random integer d_x from the interval $[1, n-1]$ as its secret parameter, then the

point $P_x = d_x G$ is computed as its public parameter and declared publicly.

Step 2: The CA selects a random integer k_x from interval $[1, n-1]$. Therefore, the secret key of the security class SC_x

is $SK_x = f(Z_x) = f(k_x G)$.

Step 3: For all security classes SC_i that satisfies $SC_x \leq SC_i$

The CA determines the point $M_{i,x} = k_x(P_i)$ and declares it publicly.

Step 4: For all security classes SC_j that satisfies $SC_j \leq SC_x$

The CA determines the point $M_{x,j} = k_j(P_x)$ and declares it publicly.

Example. As it is shown in Fig. 10, suppose a new security class SC_8 is added into user hierarchy of Fig. 1, such that $SC_2 \leq SC_8 \leq SC_1$. For SC_8 , the CA selects a random integer k_8 from the interval $[1, n-1]$. Therefore, the secret key of the security class SC_8 is $SK_8 = f(Z_8)$, and the CA determines $M_{1,8} = k_8(P_1)$, $M_{8,2} = k_2(P_8)$, $M_{8,5} = k_5(P_8)$, and $M_{8,6} = k_6(P_8)$, because SC_8 is assigned as a successor to SC_1 and as a predecessor to SC_2 . Thus, SK_8 can be derived by SC_1 and the set of secret keys $\{SK_2, SK_5, SK_6\}$ can be derived by SC_8 .

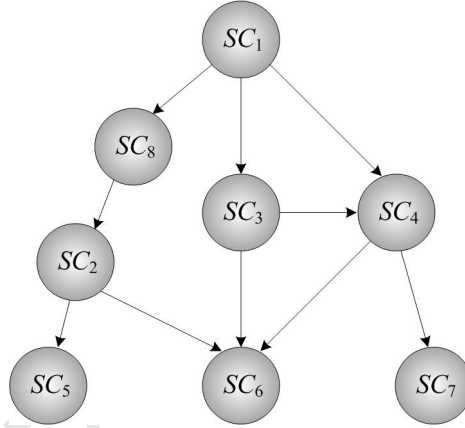


Fig. 10. The structure of Fig.1 after inserting SC_8 .

4.2. Removing the Existing Security Classes

Now, consider the case when a security class SC_x is removed from the hierarchy such that the relationship $SC_j \leq SC_x \leq SC_i$ breaks up. Therefore, the security class SC_x should have no privilege to access all the information that originally was authorized to. Thus, to control the forward security of all security classes SC_j that satisfies $SC_j < SC_k$, the CA has to renew all secret keys SK_j as SK_j^* , as follows:

Step 1: For all security classes SC_j that satisfies $SC_j \leq SC_x$

The CA reselects a random integer k_j^* from the interval $[1, n-1]$. Therefore, the new secret key of the security class SC_j is $SK_j^* = f(Z_j)$.

For all security classes SC_i that satisfies $SC_j \leq SC_i$

The CA determines the point $M_{i,j} = k_j^*(P_i)$ and declares it publicly.

Example. Suppose that SC_2 is removed from the user hierarchy of Fig. 10 and the resulted hierarchy is shown in Fig. 14. In this case, the CA has to remove all parameters related to SC_2 and revokes the accessibility of SC_2 . Thus, to control forward security of SC_5 and SC_6 , the CA reselects two random integer k_5^* and k_6^* from the interval $[1, n-1]$. Therefore, the new secret keys of the security classes SC_5 and SC_6 are $SK_5^* = f(k_5^*G)$ and $SK_6^* = f(k_6^*G)$, respectively. Then, the CA determines $M_{8,5} = k_5^*(P_8)$ and $M_{8,6} = k_6^*(P_8)$ and declares them publicly.

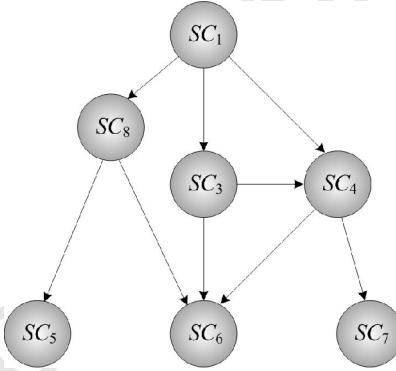


Fig. 11. The structure of figure 13 after deleting SC_2 .

4.3. Creating new relationships

Suppose $SC_l \leq SC_k$ shows a new relationship between two immediate security class such as SK_l and SK_k . To create this relationship the following procedure should be used:

Step 1: The CA determines the point $M_{k,l} = k_l(P_k)$ and declares it publicly.

Step 2: For each security class SC_i that satisfies $SC_l \leq SC_i$

If SC_i is not a predecessor of SC_l until the relationship is created such that $SC_j \leq SC_l \leq SC_k \leq SC_i$

CA determines the point $M_{i,l} = k_l(P_i)$ and declares it publicly.

Step 3: For each security class SC_j that satisfies $SC_j \leq SC_l$

For each security classe SC_i that satisfies $SC_l \leq SC_i$

If SC_i is not a predecessor of SC_j until the relationship is created such that $SC_j \leq SC_l \leq SC_k \leq SC_i$

The CA determines the point $M_{i,j} = k_j(d_l G)$ and declares it publicly.

Example. Suppose a new relationship between SK_8 and SK_3 as $SC_3 \leq SC_8$ is created in the user hierarchy of Fig. 11 and the resulted hierarchy is shown in Fig. 12. Therefore, the CA only determines $M_{8,3} = k_3(P_8)$, $M_{8,4} = k_4(P_8)$, $M_{8,7} = k_7(P_8)$, and $M_{8,6} = k_6(P_8)$ and declares them publicly.

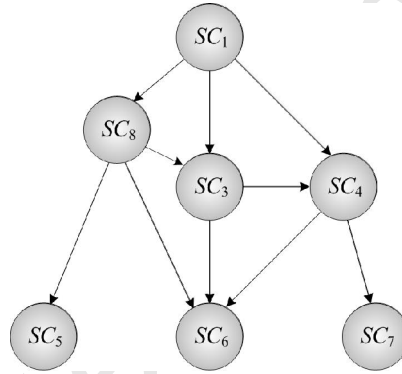


Fig. 12. The structure of Fig. 11 after creating $SC_3 \leq SC_8$.

4.4. Revoking Existing Relationships

Consider the case of revoking an existing relationship between SK_l and SK_k as $SC_l \leq SC_k$. The security class SC_k should have no privilege to access all the information that originally was authorized when the relationship is revoked from the hierarchy. Thus, to control the forward security of all security classes SC_j that satisfies $SC_j \leq SC_k$, the CA has to renew all secret keys SK_j as SK_j^* as follows:

Step 1: The CA selects a random integer k_l^* from the interval $[1, n-1]$. Therefore, the new secret key of the security

class SC_l is calculated as $SK_l^* = f(k_l^* G)$.

Step 2: For each the security class SC_i that satisfies $SC_l \leq SC_i$ after revoking the relationship

The CA determines the point $M_{i,l} = k_l^*(P_i)$ and declares it publicly.

Step 3: for each security class SC_j that satisfies $SC_j \leq SC_k$

If the relationship $SC_j \leq SC_k$ breaks up after revoking the relationship $SC_l \leq SC_k$

The CA selects a random integer k_j^* from the interval $[1, n-1]$. Therefore, the new secret key of the security class SC_j is calculated as $SK_j^* = f(k_j^*G)$.

For each security class SC_i that satisfies $SC_j \leq SC_i$

The CA determines the point $M_{i,j} = k_j^*(P_i)$ and declares it publicly.

Example. Suppose the relationship between SK_8 and SK_3 in Fig. 12 is revoked and the resulted hierarchy is shown in Fig. 13. Thus, to control forward security of SC_4 and SC_7 , the CA selects two random integer k_4^* and k_7^* from the interval $[1, n-1]$. Therefore, the new secret key of the security classes SC_4 and SC_7 are $SK_4^* = f(k_4^*G)$ and $SK_7^* = f(k_7^*G)$, respectively. Then, the CA determines $M_{1,4} = k_4^*(P_1)$ and $M_{1,7} = k_7^*(P_1)$ and declares them publicly.

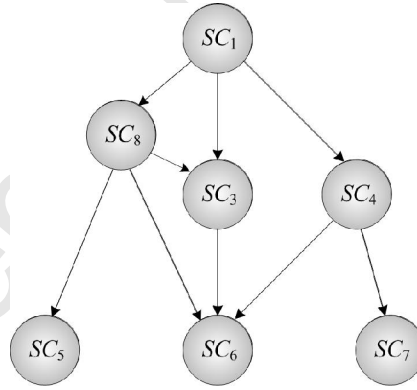


Fig. 13. The structure of Fig. 12 after revoking $SC_3 \leq SC_8$.

4.5. Changing secret keys

Changing a secret key is sometimes essential in practical implementation and as well as previous dynamical properties, it is achieved by the proposed scheme efficiently. Changing a secret key SK_x to SK_x^* is performed as follows:

Step 1: The CA selects a random integer k_x^* from the interval $[1, n-1]$. Therefore, the new secret key of the security

class SC_x is calculated as $SK_x^* = f(k_x^*G)$.

Step 2: For each security class SC_i that satisfies $SC_x \leq SC_i$

The CA determines the point $M_{i,x} = k_x^*(P_i)$ and declares it publicly.

5. Security and Performance

5.1. Security

In the proposed scheme, the difficulties associated with attacks are based on the solution of the Elliptic Curve Discrete Logarithm Problem (ECDLP), and the security resulted from such problems is still sufficient enough under the reasonable computational complexity [34]. In the following, some considerable attacks and resistance of the proposed scheme are investigated in details.

5.1.1. Masquerade attack

There is an extremely significant attack whenever a security class wants to compute authorized secret keys. Regarding to Fig. 14, a malicious user can masquerade as a the CA and publish some planned public parameters $M_{i,j}$ in public domain. Assume some security classes use these public parameters to compute secret keys SK_i . If this security class uses SK_i as a proper symmetric key and publishes encrypted confidential data into public domain, then the malicious user having proper secret key SK_i can decrypt and access these confidential data.

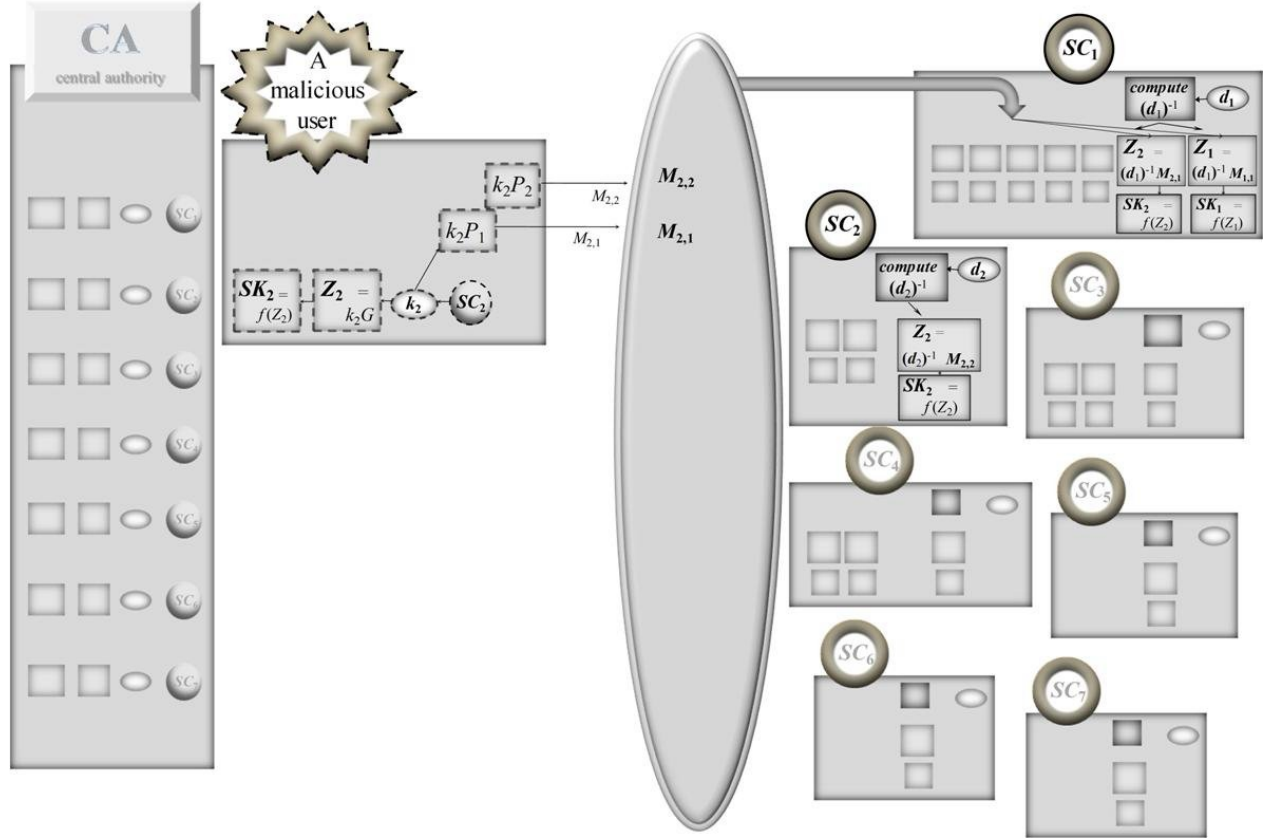


Fig. 14: The masquerade attack

An authentication mechanism such as the proposed scheme in [35] can be applied to avoid this attack; however some considerable overheads are imposed. Suppose a constant and unique private key such as b has been selected by generally named "CA" in a certain hierarchy. Moreover, assume resultant public key $Q = bG$ has been registered in a reliable center that is trustworthy for all users. In the following, employment of digital signature proposed in [35] is described step by step.

a) The CA prepares special information corresponding to each public parameter $M_{i,j}$, like a digital signature as follows:

1. Selecting a random integer such as a .
2. Computing the point $F = aG = (x_0, y_0)$ on agreed elliptic curve and assigning $r = x_0 \bmod n$. If $r = 0$ then go to step 1.
3. Converting public parameter $M_{i,j}$ into an integer e using a secure one way hash function as follows:

$$e = \text{Hash}(M_{i,j}).$$

4. Computing $s = (dre + a) \bmod n$.
 5. Publishing certain couple information (s, F) along with public parameter $M_{i,j}$.
- b) Each security class verifies validity of public parameter $M_{i,j}$ before use to compute assigned Secret key as follows:
1. Converting received $M_{i,j}$ to e by means of same hash function.
 2. Computing $v = sG$,
 3. Computing $u = erQ + F$; consider that r is x -coordinate of the received point F and Q is reliable public key of the CA.
 4. If $v = u$ then signature is valid; otherwise it is rejected.

In Fig. 15, executed operations by the CA to generate a signature for $M_{1,1}$, and required operations to verify the signature by SC_1 have been demonstrated. Similarly, all the illustrated actions have to be repeated for each public parameter $M_{i,j}$. Security of employed signature scheme has been investigated in [35].

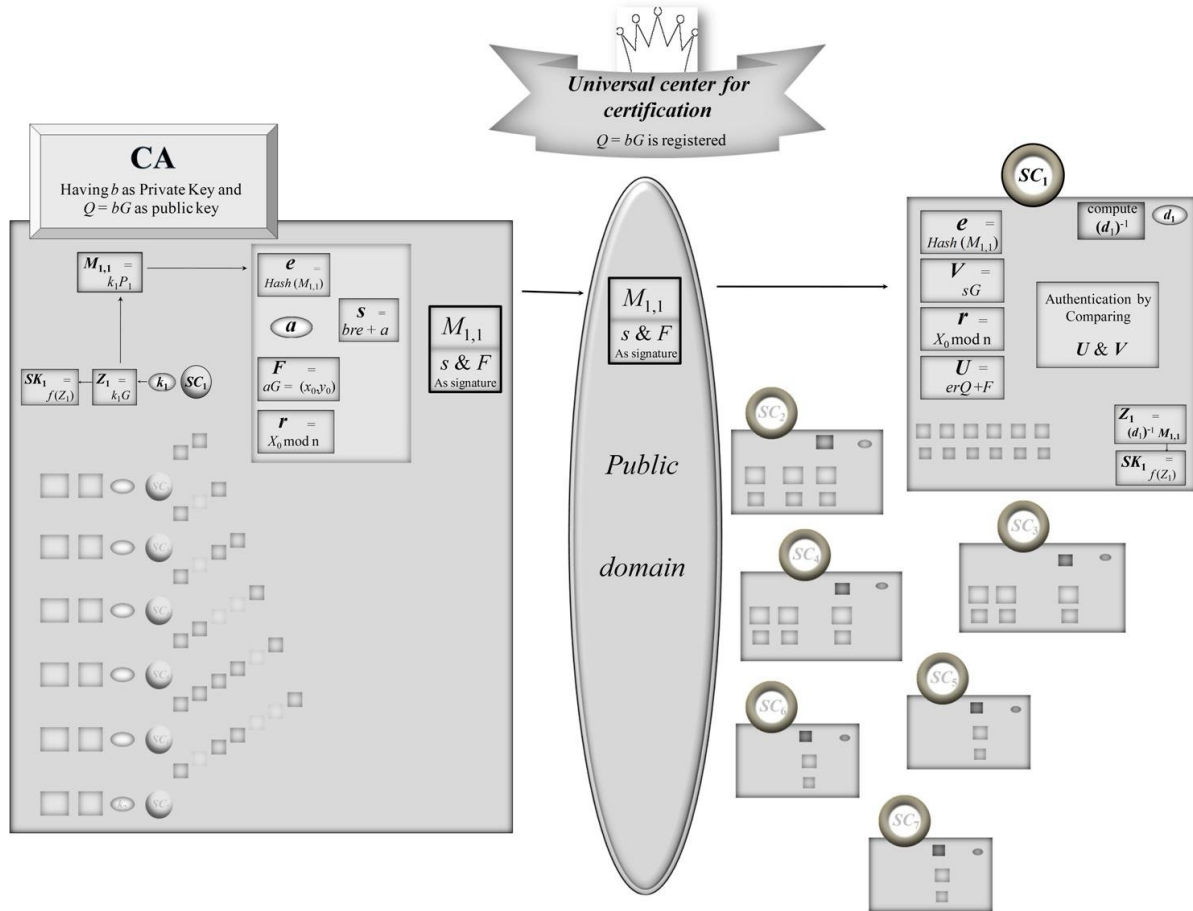


Fig. 15. Required phases for authenticating public parameters in the proposed scheme

Overhead of employing such elliptic curve based authentication policy on required storage space and computational complexity is investigated in the “performance” section of this paper.

5.1.2. Contrary attack

Assume a successor in the hierarchy tries to obtain secret key of its predecessors. This user by applying its own secret parameter, d_i , can simply compute secret key of itself and its successors. Inversely, this user is forced to solve ECDLP if decide to compute secret key of others. Regarding to resistance of ECDLP [8], [36], the proposed scheme will be reliable against this attack.

5.1.3. Collaborative attack

Assume two classes SC_2 and SC_3 in Fig. 1 are going to collaborate to obtain the secret key of SC_1 . Nevertheless they have access only to P_1 as the public parameter of SC_1 . As mentioned in Section 2, obtaining d_1 from d_1G , is the problem of solving discrete logarithm over an elliptic curve which is a computational infeasible problem [8].

Some other attacks, such as “equation attack” that is investigated in [33], are not applicable to the proposed scheme due to elimination of the polynomials in its structure.

5.2. Performance

As mentioned in the previous sections, since two proposed methods in [33] and [9] use the elliptic curve cryptosystem which has a low computational cost and small key size, these schemes are superior in both security and efficiency to other schemes. Therefore, we shall compare our proposed scheme to these ones for evaluation of performance.

Suppose there are N security classes and each of them has v_i predecessors. In both [33] and [9], for ensuring secure transmission of secret and public parameters, $2N$ points on the elliptic curve are declared publicly. Assume the length of x and y -coordinate of each elliptic curve point in an acceptable security level is 163 bits [23]. Thus,

$163 \times 2 \times 2N$ bits must be declared publicly for the transmission. Also, N points are declared publicly as public parameters of security classes in [9] or as base points in [33]. Furthermore, in these methods, all polynomials coefficients must be declared publicly. Let the finite field F_q be $q = 2^{163}$, since all of the coefficients are defined over F_q , the total storage space in both [9] and [33] is $(163 \times 2 \times 3N) + 163 \sum_{i=1}^N (v_i + 1)$ bits. In the proposed scheme, $N + \sum_{i=1}^N (v_i + 1)$ points on the elliptic curve are declared publicly. Thus, the total storage space in the proposed scheme is $163 \times 2N + 163 \times 2 \sum_{i=1}^N (v_i + 1)$ bits.

Table 2 defines the various notations used in this section. On computational complexity, the proposed method in [9] spends $3T_{EC_MUL}$ and $2T_{EC_ADD}$ to send secret key SK_i and secret parameter n_i to the CA. Also, each security class SC_i spends a T_{EC_MUL} to estimate its public parameter from its secret parameter. For security class SC_i , $1 \leq i \leq N$, each of key generation and key derivation phases spends $(v_i + 1)T_{EC_MUL}$. Furthermore, as mentioned in the previous section, based on [6] the cost of constructing an interpolating polynomial of degree v_i is exactly v_i additions, $2v_i^2 + 2$ subtractions, $2v_i^2 + v_i - 1$ multiplications and $v_i + 1$ divisions. Therefore, even if the cost of division is assumed to be equal to multiplication, in a rough estimation, minimum cost of constructing an interpolating polynomial is $2(v_i^2 + v_i)T_{MUL}$ by neglecting T_{ADD} and T_{SUB} . Thus, the total computation cost is:

$$\sum_{i=1}^N 2(v_i^2 + v_i)T_{MUL} + 2NT_{EC_ADD} + \left(4N + 2 \sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL} \quad (7)$$

The proposed methods in [33] spend $3T_{EC_MUL}$ and $2T_{EC_ADD}$ to send the secret key SK_i and the sub-secret key s_i to corresponding security class SC_i . For security class SC_i , $1 \leq i \leq N$, each of key generation and key derivation phases spends $(v_i + 1)T_{EC_MUL}$. Furthermore, as mentioned in the above, minimum cost of constructing an interpolating polynomial is $2(v_i^2 + v_i)T_{MUL}$, thus, the total computation amount is:

$$\sum_{i=1}^N 2(v_i^2 + v_i)T_{MUL} + 2NT_{EC_ADD} + \left(3N + 2 \sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL} \quad (8)$$

In the proposed method, computing inversion of all secret parameters d_i requires NT_{inv} . Determination of public parameters from secret parameters requires NT_{EC_MUL} . Key generation phase spends $\left(\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL}$ for computing all values of $M_{i,j}$. Also deriving all values of the secret keys by authorized security classes spends $\left(\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL}$. Thus, the total computation amount is:

$$NT_{INV} + \left(N + 2\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL} \quad (9)$$

In Table 3, the conversion of various operation units to the time complexity for executing the modular multiplication is given based on the reference [23]. The time complexity of the proposed scheme and two mentioned methods in terms of modular multiplication operation is illustrated in Table 4. Based on Table 3, all the estimated times have been exhibited in terms of the required execution time for modular multiplication.

Table 2: Definition of some notations used in performance evaluation of the proposed scheme

Notation	Definition
T_{MUL}	Time complexity for the execution of a modular multiplication
T_{SUB}	Time complexity for the execution of a modular subtraction
T_{ADD}	Time complexity for the execution of a modular addition
T_{EC_MUL}	Time complexity for the execution of a multiplication in an elliptic curve point
T_{EC_ADD}	Time complexity for the execution of an addition of two points in an elliptic curve
T_{inv}	Time complexity for the execution of a modular inversion

Table 3: Unit conversion of various operations in terms of T_{MUL}

Time complexity of an arithmetic unit	Time complexity in terms of modular multiplication
T_{EC_MUL}	$1200 T_{MUL}$
T_{EC_ADD}	$5 T_{MUL}$
T_{ADD}, T_{SUB}	Negligible
T_{INV}	$3T_{MUL}$

Table 4: Required time complexity in unit of T_{MUL}

	Required storage space	Required computational cost	
		Time complexity	Rough estimation
[9]	$163\left(\sum_{i=1}^N (v_i + 1) + 6N\right)$	$\sum_{i=1}^N 2(v_i^2 + v_i)T_{MUL} + 2NT_{EC_ADD} + \left(4N + 2\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL}$	$\left[\sum_{i=1}^N (2v_i^2 + 2402v_i) + 4810N + 2400\right]T_{MUL}$
[33]	$163\left(\sum_{i=1}^N (v_i + 1) + 6N\right)$	$\sum_{i=1}^N 2(v_i^2 + v_i)T_{MUL} + 2NT_{EC_ADD} + \left(3N + 2\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL}$	$\left[\sum_{i=1}^N (2v_i^2 + 2402v_i) + 3610N + 2400\right]T_{MUL}$
Proposed scheme	$163\left(2\sum_{i=1}^N (v_i + 1) + 2N\right)$	$NT_{INV} + \left(N + 2\sum_{i=1}^N (v_i + 1)\right)T_{EC_MUL}$	$\left[\left(\sum_{i=1}^N 2400v_i\right) + 1203N + 2400\right]T_{MUL}$

From the result of comparison, we can see that in terms of storage space and especially computation amount, the performance of the proposed scheme far exceeds than other two methods. Furthermore, it is apparent that cost of any alteration such as insertion and removal of classes, updating of their relationships and changing of secret keys is less than other schemes.

None of presented schemes in [9] and [33] considers the “masquerade attack” as stated in our paper. Employment of any authentication policy imposes considerable cost to access control scheme. Therefore, for a fair comparison, authentication cost is not shown in Table 3.

Considering all assumptions about the size of hierarchy and used symbols in this section, additional storage space and computational cost of authentication policy in the proposed scheme is summarized in Table 5.

Table 5: Required storage space and computation costs for authentication of public parameters.

Required storage space	Required computational cost	
	For CA	For sum of security classes
$163\left(3\sum_{i=1}^N (v_i + 1)\right)$	$\left(\sum_{i=1}^N (v_i + 1)\right)[hash + T_{EC_mul} + 2T_{Mul} + T_{add}]$	$\left(\sum_{i=1}^N (v_i + 1)\right)[hash + 2T_{EC_mul} + T_{mul} + T_{EC_add}]$

Another applicable procedure in the proposed scheme is point compression methods. Regarding the required storage space for public domain in Fig. 8, some points in elliptic curve are published. There are some methods to reduce size of required storage space in public domain by nearly half. In these techniques, only one of x or y -coordinate for each point is maintained in the public domain. In addition to the selected coordinate, another bit has to be saved that its value is determined depending on the compression method. A practical compression method has

been described in [8] in detail. Although employment of point compression method decreases required storage space in the public domain, it imposes heavy computational cost to the scheme while its improvement includes none of private domains. That is to say, employment of compression method is not required when we use a public domain as “public key directory”.

6. Conclusion

In this paper, a novel key management method based on elliptic curve cryptosystem was proposed to solve dynamic access problems in a user hierarchy. The time complexity and the required storage space of the proposed scheme was compared with the previous works and the results indicated that the proposed method had less time complexity compared to other related works even recently ECC based presented schemes. The Proposed scheme is efficient in the key-derivation process especially, since any user can derive any of his successors key directly, instead of iteratively.

References

- [1] ANSI, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). American National Standard, X9.62-1998
- [2] C.C. Chang, R.J. Hwang, T.C. Wu, Cryptographic key assignment scheme for access control in a hierarchy, *Information Systems* 17 (3) (1992), pp. 243–247.
- [3] C.C. Chang, I.C. Lin, A new solution for assigning cryptographic keys to control access in mobile agent environments, *Wireless Communications and Mobile Computing* 6 (1) (2006), pp. 137–146.
- [4] C.C. Chang, I.C. Lin, H.M. Tsai, H.H. Wang, A key assignment scheme for controlling access in partially ordered user hierarchies, in: *Proceedings of the 18th IEEE International Conference on Advanced Information Networking and Applications (AINA2004)*, Fukuoka, Japan, vol. 2, March 2004, pp. 376–379.
- [5] C.C. Chang, D.J. Buehrer, Access control in a hierarchy using a one-way trapdoor function, *Computers and Mathematics with Applications* 26 (5) (1993), pp. 71–76.
- [6] D.E. Knuth, 3rd ed., *The Art of Computer Programming*, vol.2: Semi numerical Algorithms, Addison-Wesley, Reading, MA, (1998).

- [7] D. Hankerson, A.J. Menezes, S. Vanstone, Guide to elliptic curve cryptography, Springer-Verlag New York (2004)
- [8] D. Johnson, A. Menezes, S. Vanstone, The elliptic curve digital signature algorithm (ECDSA), *International Journal of Information Security* 1 (1) (2001), pp. 36–63.
- [9] F.G. Jeng, C.M. Wang, An efficient key-management scheme for hierarchical access control based on elliptic curve cryptosystem, *The Journal of Systems and Software*, 79 (2006), pp. 1161–1167
- [10] G. Lopez, O. Canovas, A.F. Gomez, J.D. Jimenez, R. Marin, A network access control approach based on the AAA architecture and authorization attributes, *Journal of Network and Computer Applications* 30 (2007), pp. 900–919.
- [11] H.R. Hassen, A. Bouabdallah, H. Bettahar, Y. Challal, Key management for role hierarchy in distributed systems, *Journal of Network and Computer Applications* 30 (2007), pp. 920–936.
- [12] I.C. Lin, H.H. Ou, M.S. Hwang, Efficient access control and key management schemes for mobile agents, *Computer Standards & Interfaces* 26 (2004), pp. 423–433
- [13] J. Biskup, D.W. Embley, and J.H. Lochner, Reducing inference control to access control for normalized database schemas, *Information Processing Letters*, article in press, Accepted 17 September 2007.
- [14] J.H. Yeh, a secure time-bound hierarchical key assignment scheme based on RSA public key cryptosystem, *information processing letters* 105 (2008), pp. 117–120.
- [15] J. Wu, R. Wei, An access control scheme for partially ordered set hierarchy with provable security, in: *Proceedings of SAC 2005, LNCS 3897*, (2006), pp. 221–232.
- [16] K.P. Wu, S.J. Ruan, C.K. Tseng, F.P. Lai, Hierarchical access control using the secure filter, *IEICE Transactions on Information & Systems* E84-D (6) (2001), pp. 700–707.
- [17] K. Kobitz, Elliptic curve cryptosystems. *Mathematics of Computation* 48 (177) (1987), pp. 203–209.
- [18] K.H. Huang, Y.F. Chung, C.H. Liu, F. Lai, T.S. Chen, Efficient migration for mobile computing in distributed networks, *Computer Standards & Interfaces*, article in press, accepted 10 October 2007
- [19] L. Harn and H.Y. Lin, A cryptographic key generation scheme for multilevel data security, *Computer Security* 9 (1990), pp. 539–546.
- [20] M.S. Hwang, W.P. Yang, Controlling access in large partially-ordered hierarchies using cryptographic keys, *Journal of Systems and Software* 67 (2) (2003), pp. 99–107.

- [21] N. Saxena, G. Tsudik, J.H. Yi, Threshold cryptography in P2P and MANETs: The case of access control, *Computer Networks* 51 (2007), pp. 3632–3649.
- [22] NIST, Recommendation on Key management, DRAFT Special Publication (2003), pp. 800-857, <http://csrc.nist.gov/CryptoToolkit/kms/guideline-1-jan03.pdf>.
- [23] N. Kobitz, A. Menezes, S.A. Vanstone, The state of elliptic curve cryptography, *Designs, Codes and Cryptography* 19 (2–3) (2000), pp. 173–193.
- [24] R.L. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* 21 (2) (1978), pp. 120–126.
- [25] S.G. Akl and P.D. Taylor, Cryptographic solution to a multilevel security problem. in: Chaum D, Rivest RL, Sherman AT, editors. *Advances in cryptology*. (1982), pp. 237-249.
- [26] S.G. Akl and P.D. Taylor, Cryptographic solution to a problem of access control in a hierarchy, *ACM Transaction on Computer Systems* 1(3) (1983), pp. 239-248.
- [27] S.T. Mackinnon, P.D. Taylor, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transaction on Computer systems* 34 (9) (1985), pp. 797-802.
- [28] S.A. Vanstone, Elliptic curve cryptosystem-the answer to strong, fast public-key cryptography for securing constrained environments, *Information Security Technical Report* 12 (2) (1997), pp. 78–87.
- [29] V. Roth and M. Jalali, access control and key management for mobile agents, *Computer & Graphics*, 22 (4) (1998), pp. 457-461
- [30] V.R.L. Shen, T.S. Chen, A novel key management scheme based on discrete logarithms and polynomial interpolations, *Computers and Security* 21 (2) (2002), pp. 164–171.
- [31] V.S. Miller, Use of elliptic curves in cryptography. In: *Advances in Cryptology, CRYPTO_85, Lecture Notes in Computer Science*, vol. 218, (1986), pp. 417–426.
- [32] W. Stallings, *Cryptography and Network Security: Principles and Practice*, fourth ed., Prentice Hall, 2005.
- [33] Y.F. Chung, H.H. Lee, F. Lai, T.S. Chen, Access control in user hierarchy based on elliptic curve Cryptosystem, *Information Sciences*, 178 (2008), pp. 230–243
- [34] Y.F. Chung, K.H. Huang, F. Lai, T.S. Chen, ID-based digital signature scheme on the elliptic curve cryptosystem, *Computer Standards & Interfaces*, 29 (2007), pp 601–604

- [35] Nikooghadam, M., Bonyadi, M. R., Malekian, E., & Zakerolhosseini, A. A protocol for digital signature based on the elliptic curve discrete logarithm problem. *Journal of Applied Sciences*, 8 (10) (2008), pp. 1919-1925
- [36] NIST, Special Publication 800-56A, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, March, 2006
- [37] P.P. Deepthi, P.S. Sathidevi, New stream ciphers based on elliptic curve point multiplication, *Computer Communications* 32 (2009), pp. 25–33
- [38] A.P. Fournaris, O. Koufopavlou, “Versatile Multiplier Architectures in $GF(2^k)$ fields using the Montgomery Multiplication Algorithm”, *INTEGRATION, the VLSI journal*, vol. 41 Issue 3 may 2008, paged 371-384.
- [39] I.E. Shparlinski, *Finite Fields: Theory and Computation*. Kluwer Academic Publishers, May 1999. [4] L. Song and K.K. Parhi, “Efficient finite field serial/parallel multiplication,” *In Proc. of the 10th IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP '96)*, pp. 72–82, IEEE Comput. Society Press, Aug. 1996.
- [40] M. Nikooghadam, E. Malekian, A. Zakerolhosseini, “A Versatile Reconfigurable Bit-serial Multiplier Architecture in Finite Fields $GF(2^m)$ ”, *Communications in Computer and Information Science*, vol. 6, (2009) pp. 227-234, Springer.