Journal of Computer Science 10 (9): 1650-1665, 2014
ISSN: 1549-3636
© 2014 Science Publications
doi:10.3844/jcssp.2014.1650.1665 Published Online 10 (9) 2014 (http://www.thescipub.com/jcs.toc)

HIERARCHICAL ACCESS CONTROL IN DYNAMIC PEER GROUPS USING SYMMETRIC POLYNOMIAL AND TREE BASED GROUP ELLIPTIC CURVE DIFFIE HELLMAN SCHEME

[1]Nafeesa Begum Jeddy, [2]Kumar Krishnan and [3]Sumathy Vembu

[1]Department of Computer Science, Government College of Engineering, Bargur, Tamilnadu, India
[2]Department of Computer Science and Engineering, Government College of Technology, Coimbatore, India
[3]Department of Electronics and Communication Engineering, Govt. College of Tech., Coimbatore, India

Received 2014-02-21; Revised 2014-03-18; Accepted 2014-04-15

ABSTRACT

Hierarchical Access Control in group communication is an active area of research which is difficult to achieve it. Its primary objective is to allow users of a higher authority group to access information or resource held by lower group users and preventing the lower group users to access information held by higher class users. Large collection of collaborative applications in organizations inherently has hierarchical structures for functioning, where providing security by efficient group key management is a big challenging issue. While preserving centralized methods for hierarchical access control, it is difficult to achieve efficiency as a single membership change will result in lot of changes which are difficult to maintain. So, using distributed key agreement techniques is more appropriate for this scenario. This study explore on novel group key agreement approach, which combines both the symmetric polynomial scheme and Tree Based Group elliptic Curve key exchange. Also, it yields a secure protocol suite that is good in fault-tolerant and simple. The efficiency of SP-TGECDH is better than many other schemes. Using TGECDH makes the scheme suitable small Low powered devices.

Keywords: Collaborative, Dynamic Peer Groups, Hierarchical Access Control, Symmetric Polynomial, Tree Based Group Elliptic Curve Key Exchange

1. INTRODUCTION

Fault-tolerant, scalable and reliable communication services have become critical in modern computing. An important and popular trend is to convert traditional centralized services (e.g., file sharing, authentication, web and mail) into distributed services spread across multiple systems and networks (Kim *et al*., 2004). Many of these newly distributed and other inherently collaborative applications (e.g., conferencing, white-boards and shared instruments) need secure communication. However, experience shows that security mechanisms for collaborative and dynamic peer groups tend to be both expensive and unexpectedly complex. In that regard, dynamic peer groups are very different from non-collaborative, centrally managed, one-to-many (or few-to-many) broadcast groups such as those encountered in Internet multicast.

Dynamic Peer Groups (DPGs) are common in many layers of the network protocol stack and many application areas of recent computing. Examples of DPGs include replicated servers, audio conferencing, video conferencing and other applications supporting collaborative work. Comparing large multicast groups, DPGs seem to be relatively small in size, in the order of hundred larger groups are harder to control on a peer basis and are often organized in a hierarchy. DPGs typically assume a many-to-many (or, equivalently, any-to-any) communication pattern rather than one-to-many pattern common of larger hierarchical groups. Despite

Corresponding Author: Nafeesa Begum Jeddy, Department of Computer Science, Government College of Engineering, Bargur, Tamilnadu, India

their relatively small number, group members in a DPG could be spread throughout the Internet and should be able to deal with arbitrary partitions due to network failures, congestion and hostile attacks. In essence, a group can be split into a number of disconnected partitions each of which must persist and function as an independent peer group (Kim *et al*., 2004).

Security requirements in collaborative DPGs present several interesting research challenges. In this study, we focus on secure and efficient group key management. The goal of group key management is to set up and maintain a shared secret key among the group members (Zhong, 2002). It serves as a foundation for other DPG security services.

There are many applications in organizations that share data in a carefully managed fashion by using access control mechanisms. The common method used for enforcing access control is by encrypting the data and managing the encryption keys. Access control can be Discretionary Access Control, Role-Based Access Control, Mandatory Access Control and Hierarchical Access Control. Discretionary Access Control restricts access to objects based solely on the identity of users who are trying to access them.

Mandatory Access Control mechanisms assign a security level to all information, assign a security clearance to each user and ensure that all users only have access to that data for which they have a clearance. It has better security than Discretionary Access Control. In Role Based Access Control a user has access to an object based on the assigned role. Roles are defined based on job functions. Permissions are defined based on job authority and responsibilities within a job function. Based on the permissions, operations on an object are invocated. The object is concerned with the user's role and not the user. Users can change roles frequently and hence once roles are fixed, access can be given to roles and objects of the respective roles get permission.

The realistic assumption is that the structure of any organization (**Fig. 1**) is a hierarchy of security classes lead to Hierarchical Access Control. Hierarchical Access Control is very difficult to achieve in secure group communication due to highly dynamic nature of members (Kuang *et al*., 2011; Aparna and Amberker, 2009). In a hierarchical access control system, users are partitioned into a number of classes called security classes which are organized in a hierarchy. Hierarchies arise in systems where some users have higher privileges than others and a security class inherits the privileges of its descendant classes.

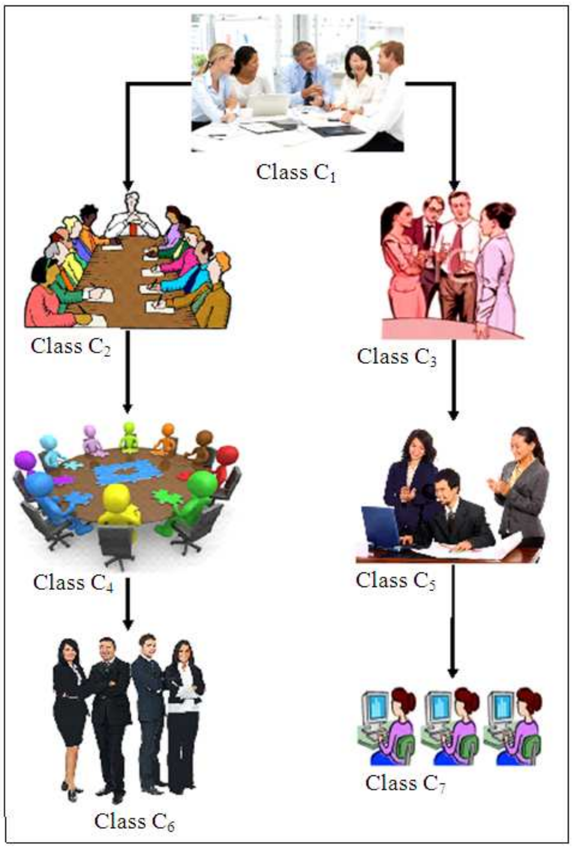

Fig. 1. A sample hierarchy in organizations

As shown in **Fig. 1**, there are seven classes C_1, C_2, C_3, C_4, C_5, C_6 and C_7. Each class has any number of users. There are two hierarchies in the above example and they are $C_1>C_2>C_4>C_6$ and $C_1>C_3>C_5>C_7$. Users of an increasing class should be able to access the resources held by the users of the descendant class. Hierarchical Access Control Problem is defined as the procedure by which members in a group can communicate with each other in a secure manner so that the information or resource that is being shared is known to the members of that group and all members who are destined as ancestors to the group. In such circumstances, a group key is set up among all the participating members and this key is used to encrypt all the messages destined to the group. This key has to be relayed to all the members who are ancestors so that the communication or resource is accessed by the ancestors.

An exceptional protocol should proficiently manage the group key when members join and leave in a descendant group. Hence user dynamics has to be taken care with at most attention to issues of forward and backward secrecy. In rare cases new classes may need to created and added to the hierarchy and classes may also be deleted. Hence, class dynamics has to be addressed. The desirable properties of an ideal Hierarchical Access Control are Key Establishment for a shared group key (Hwang and Satchell, 1999), Enforcement of Forward and Backward Secrecy, Busty operation with simultaneous multiple user join or leave, Efficiency with a minimum amount of computation and communication, Key Establishment for Hierarchical Access Control, User Dynamics and Class Dynamics (Lin *et al*., 2003).

2. SYSTEM OVERVIEW

The goal of this research is to propose a communication and computation efficient Hierarchical Access control protocol for Dynamic Peer Groups using Symmetric Polynomial Scheme (Das *et al*., 2005; Begum *et al*., 2010a) and Tree Based Group Elliptic Curve Diffie Hellman Scheme (Wang *et al*., 2006). In huge and highly dynamic networks, it is very difficult to have hierarchical access control. It is commendable to use a dual layer encryption protocol to protect the resources from adversaries and also to provide access to resources for the ancestor users. In the proposed scheme the dual encryption uses the key formed by TGECDH for communication within the dynamic peer groups and for communication among the dynamic peer groups which involves ancestor classes the symmetric polynomial scheme (Zou and Bai, 2008) is used. Use of TGECDH scheme within the classes involves its suitability for use in low power small devices which are abundantly used nowadays. The use of two levels of keys reduces the large enormous cost of computation and communication in re-keying. Hence, in Hierarchical Access control protocol for Dynamic Peer Groups all the member nodes joins their respective classes.

As shown in **Fig. 2** the Hierarchical Access Control Problem in an Organization can be logically represented as a set of classes (Hwang and Yang, 2003). Assume that in every class, every user can receive a message broadcasted from other members. The key formation is based on efficient Tree Based Group Elliptic Curve Diffie Hellman protocol. Here, it is not required to create a common key for the global group but for every class keys must be created independently and outer keys using the symmetric polynomial scheme are formed with the help of Central Authority. If a user in a class transmits a message, the message is encrypted by TGECDH key and all the members including the class controller decrypt it. The key is transmitted to the Gateway Node.

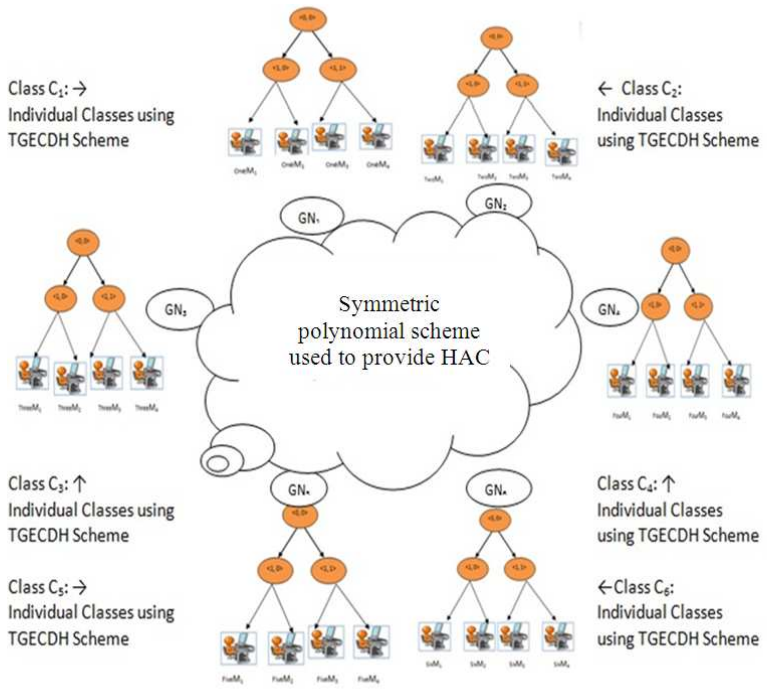

Fig. 2. A sample hierarchy in organizations

Also, the Gateway Node encrypts the message again using its respective symmetric polynomial key and broadcasts it. All the Gateway Nodes receive this. The Gateway Nodes of the ancestor classes derive the key using polynomial approach.

They decrypt the message and encrypt it again with the class key obtained using TGECDH and transmits to all users of its class. Thus all users of the ancestor class are able to access the resource of their descendant classes.

This Hierarchical Access control protocol for Dynamic Peer Groups using Symmetric Polynomial Scheme and Tree Based Group Elliptic Curve Diffie Hellman Scheme is suitable for the inherent characteristics of networks which have independent, mobile and unreliable links. The main reasons can be summarized as follows:

- The protocol is based on contributory group key agreement in the lower layer, so it does not require pre-existing infrastructure except a special node called the Gateway Node for performing a TGECDH Group key agreement. It is a hybrid scheme where a central authority needs to have communication with the gateway Nodes and once the polynomial is sent to them. There is no more interference except in very rare cases during a hierarchy change
- The Gateway Nodes of each class receive two sets of keys. One from the class controllers of their respective classes and another from the Central Authority which may be powerful nodes in the infrastructure with the capability of encrypting and decrypting and multicasting to their group members
- The protocol is efficient especially for large, dynamic group. Because re-keying of one class does not influence other groups which avoids the problem of “1-affects-n” (i.e., a single membership change in the group affects all the other group members) and it can provide better performance
- Networks are always composed of unreliable links

This Scheme provides reliable communication for networks.

The proposed Hierarchical Access control protocol for Dynamic Peer Groups using Symmetric Polynomial Scheme and Tree Based Group Elliptic Curve Diffie Hellman Scheme mainly targets at security, scalability and efficiency (Wang *et al*., 2006).

A Class key CK is used for communication between members in the same class. The Class key is re-keyed whenever there is a membership change, joins or leaves and member failure. The Outer keys are rarely changed only during the change in the hierarchical Structure.

3. KEY ESTABLISHMENT AMONG THE DYNAMIC PEER GROUPS USING SYMMETRIC POLYNOMIALS

In mathematics, specifically in commutative algebra, the elementary symmetric polynomials are one type of basic building block for symmetric polynomials, in the sense that any symmetric polynomial P can be expressed as a polynomial in elementary symmetric polynomials: P can be given by an expression involving only additions and multiplication of constants and elementary symmetric polynomials. The symmetric polynomials have an excellent property that when the order of the parameters is changed also, the value of the polynomial does not change. This property is used in hierarchical access control by the ancestor classes to derive the keys of the descendant classes.

The underlying principle of the upper Tier scheme is secret sharing. Unlike other schemes which are computationally secure. The proposed scheme is unconditionally secure. A central authority chooses a large positive integer P as the system modulus, P need not be a prime and a threshold t. CA randomly generates a symmetric polynomial in m variables with co-efficient from Z_p in which the degree of any variables is at most t as Equation 1:

$$p(x_1, x_2, \ldots, x_m) = \sum_{i_1}^{t} \sum_{i_2}^{t} \ldots \sum_{i_m}^{t} a_{i_1, i_2, \ldots, i_m} x_1^{i_1} i_2^{i_2} \ldots x_m^{im} \pmod p \qquad (1)$$

where, a_i is randomly generated coefficients by the Central Authority. Every class in the hierarchy has a polynomial function which is derived from $P(x_1, x_2, \ldots, x_m)$ and the polynomial function is transmitted to each class securely by the Central Authority.

To derive proper keys in the hierarchy, the CA generates some publicly known numbers:

- N random numbers s_i associated with C_i for i = 1, 2,... n
- (m-1) additional random numbers $s_j^{'}$ for j = 1, 2, ... , m-1. (s_i and $s_j^{'}$ belong to Z_p)

For each security class C_i with an ancestor set $S_i = \{C_{i1}, C_{i2}, \ldots, C_{im}\}$ where i_j is an ordinal number such that $1 \leq i_j \neq i \leq n$, security class C_i is given a polynomial function, g_i derived by the CA as Equation 2:

$$\begin{aligned} & g_i(x_{mi+2}, x_{mi+3}, \ldots, X_m) \\ & = p(s_i, s_{i1}, +s_{i2}, \ldots, s_{im}, x_{mi+2}, x_{mi+3}, \ldots, s_m) \end{aligned} \qquad (2)$$

3.1. Working of the Symmetric Polynomial Scheme

The following case study is taken into consideration to demonstrate the working of the Symmetric Polynomial.

The following steps are followed to show Hierarchical Access Control using Symmetric Polynomials:

- Diagrammatic Representation of the Hierarchy is given above
- Key Calculation by the respective Classes. The keys are named as K1, K_2 to mean that Class C_1 calculates is key K_1, Class C_2 calculates is key K_2
- Key Derivation of a Descendant Class by the Ancestor Classes. The keys are named as $AK_{1,2}$, $AK_{1,3}$ to mean that Ancestor Class C_1 derives the key of its descendant Class C_2 ($AK_{1,2}$), Ancestor Class C_1 derives the key of its descendant Class C_3 ($AK_{1,3}$)
- Key Derivation by a Class who is not an ancestor of classes. The key is called as $NK_{2,1}$, $NK_{3,1}$, $NK_{3,6}$ to mean that a non-ancestor class C_2 derives the key of C_1 ($NK_{1,2}$), non-ancestor class C_3 derives the key of C_1 ($NK_{1,3}$) non-ancestor class C_3 derives the key of C_6 ($NK_{3,6}$)
- It is shown that K_2 = $AK_{1,2}$, K_2 = $AK_{3,2}$, i.e., the symmetric polynomials are evaluated with the same parameters but with different permutations
- It is shown that $K1 \neq NK_{21}$, $K_1 \neq NK_{31}$, $K_6 \neq NK_{36}$ i.e., the symmetric polynomials are evaluated with different parameters which results in the wrong value for the key
- There is no restriction on the number of users in each class. The Key Calculation, Key Derivation is taken care by one user of each class designated as the class controller
- In all the case studies the following symmetric Polynomial is used Equation 3

$$p(x_1, x_2, \ldots, x_m) = \sum_{i_1=0}^{t}\sum_{i_2=0}^{t}\cdots\sum_{i_m=0}^{t} a_{i_1,i_2,\ldots,i_m} x_1^{i_1} i_2^{i_2} \ldots x_m^{im} \bmod p \qquad (3)$$

- The value of m is chosen as $m \geq \max\{m_1, m_2, m_3, \ldots, m_i\}+1$, A greater value of m allows to add more classes. Here m value is chosen as 6
- To calculate its key each class applies uses the symmetric polynomial with m parameters. The m parameters are chosen as follows. $K_i = s_i, s_{i1}, s_{i2}, \ldots, s_{imi}, s'_1, s'_2, \ldots, s'_{m-mi-1})$
- Key Derivation: In key derivation, a term Sj/i that is used to identify the hierarchy is used:

$$S_{j/I} = S_j / (S_i U\{C_i\}) = \{C_{(j/i)1}, C_{(j/i)2}, \ldots, C_{(j/i)rj}\} \qquad (4)$$

- 12) Consider a security class C_i which is ancestor to security class C_j and key K_j can be calculated by C_i as:

$$\begin{aligned} K_j &= g_i(S_j, s_{(j/i)1}, s_{(j/i)2}, \ldots, s_{(j/i)rj}, s'_1, s'_2, \ldots, s'_{m-mi-2-rj}) \\ &= p(s_i, s_j, s_i, s_{i1}, s_{i2}, \ldots, s_{imi}, s_{(j/i)1}, \\ &\quad s_{(j/i)2}, \ldots, s_{(j/i)rj}, s'_1, s'_2, \ldots, s'_{m-mi-2-rj}) \end{aligned} \qquad (5)$$

As shown in **Table 1** Set of Classes = {C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}), Set of Ancestor Classes = {S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15} ,S_{16}}. There are five ancestors in the Hierarchy. C_1 is the ancestor of C_2, C_3, C_4 and C_5. C_2 is the ancestor of C_6, C_7, C_8 and C_9. C_3 is the ancestor of C10, C_{11} and C_{12}. C_4 is the ancestor of C_{13} and C_{14}. C_5 is the ancestor of C_{15} and C_{16}. Equation 4 and 5 are used for deriving the keys.

Table 1. Set of classes and its ancestor classes

Set S and C	m Value	Set K
S_1 = Ancestor of $C_1 = \{\Phi\}$	$m_1 = 0$	$m-m_i -1= 5$ $K_1 = (s_1, s'_1, s'_2, s'_3, s'_4, s'_5)$
S_2 = Ancestor of $C_2 = \{S_1\}$	$m_2 = 1$	$m-m_i -1 = 4$ $K_2 = (s_2, s_1, s'_1, s'_2, s'_3, s'_4)$
S_3 = Ancestor of $C_3 = \{S_1\}$	$m_3 = 1$	$m-m_i -1 =4$ $K_3 = (s_3, s_1, s'_1, s'_2, s'_3, s'_4)$
S_4 = Ancestor of $C_4 = \{S_1\}$	$m_4 = 1$	$m-m_i -1 = 4$ $K_4 = (s_4, s_1, s'_1, s'_2, s'_3, s'_4)$
S_5= Ancestor of $C_5 = \{S_1\}$	$m_5 = 1$	$m-m_i -1 = 4$ $K_5 = (s_5, s_1, s'_1, s'_2, s'_3, s'_4)$
S_6 = Ancestor of $C_6 = \{S_1, S_2\}$	$m_6 = 2$	$m-m_i -1 = 3$ $K_6 = (s_6, s_1, s_2, s'_1, s'_2, s'_3)$
S_7 = Ancestor of $C_7 = \{S_1, S_2\}$	$m_7 = 2$	$m- m_i -1 = 3$ $K_7 = (s_7, s_1, s_2, s'_1, s'_2, s'_3)$
S_8 = Ancestor of $C_8 = \{S_1, S_2\}$	$m_8 = 2$	$m-m_i -1 =3$ $K_8= (s_8, s_1, s_2, s'_1, s'_2, s'_3)$
S_9 = Ancestor of $C_9 = \{S_1, S_2\}$	$m_9 = 2$	$m-m_i -1 = 3$ $K_9 = (s_9, s_1, s_2, s'_1, s'_2, s'_3)$
S_{10} = Ancestor of $C_{10} = \{S_1. S_3\}$	$m_{10} = 2$	$m- m_i -1 = 3$ $K_{10} = (s_{10}, s_1, s_3, s'_1, s'_2, s'_3)$
S_{11} = Ancestor of $C_{11} = \{S_1, S_3\}$	$m_{11} = 2$	$m- m_i -1 = 3$ $K_{11} = (s_{11}, s_1, s_3, s'_1, s'_2, s'_3)$
S_{12} = Ancestor of $C_{12} = \{S_1, S_3\}$	$m_{12} = 2$	$m-m_i -1 = 3$ $K_{12} = (s_{12}, s_1, s_3, s'_1, s'_2, s'_3)$
S_{13} = Ancestor of $C_{13} = \{S_1, S_4\}$	$m_{13} = 2$	$m-m_i -1 = 3$ $K_{13} = (s_{13}, s_1, s_4, s'_1, s'_2, s'_3)$
S_{14} = Ancestor of $C_{14} = \{S_1, S_4\}$	$m_{14} = 2$	$m-m_i -1 = 3$ $K_{14} = (s_{14}, s_1, s_4, s'_1, s'_2, s'_3)$
S_{15} = Ancestor of $C_{15} = \{S_1, S_5\}$	$m_{15} = 2$	$m- m_i -1= 3$ $K_{15} = (s_{15}, s_1, s_5, s'_1, s'_2, s'_3)$
S_{16} = Ancestor of $C_{16} = \{S_1, S_5\}$	$m_{16} = 2$	$m-m_i -1 = 3$ $K_{16} = (s_{16}, s_1, s_5, s'_1, s'_2, s'_3, s'_4, s'_5)$

Ancestor: 1 = C_1

Key derivation of C_2 by C_1:

$j = 2, i = 1$
$S_j/_i = S_j/S_i \cup C_i$
$= \{C_1\}/\{\Phi \cup C_1\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6 - 0 - 2 - 0 = 4$
$AK_{1,2} = P(s_1, s_2, s_1', s_2', s_3', s_4')$
$K_2 = (s_2, s_1, s_1', s_2', s_3', s_4')$
$AK_{1,2} = K_2$ (same parameters in different permutation)

Key derivation of C_3 by C_1:

$j = 3, i = 1$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1\}/\{\Phi \cup C_1\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}0\text{-}2\text{-}0 = 4$
$AK_{1,3} = P(s_1, s_3, s_1', s_2', s_3', s_4')$
$K_3 = (s_3, s_1, s_1', s_2', s_3', s_4')$
$AK_{1,3} = K_3$ (same parameters in different permutation)

Key derivation of C_4 by C_1:

$j = 4, i = 1$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1\}/\Phi \cup C_1\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}0\text{-}2\text{-}0 = 4$
$AK_{1,4} = P(s_1, s_4, s_1', s_2', s_3', s_4')$
$K_4 = (s_4, s_1, s_1', s_2', s_3', s_4')$
$AK_{1,4} = K_4$ (same parameters in different permutation)

Key derivation of C_5 by C_1:

$j = 5, i = 1$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1\}/\{\Phi \cup C_1\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}0\text{-}2\text{-}0 = 4$
$AK_{1,5} = P(s_1, s_5, s_1', s_2', s_3', s_4')$
$K_5 = (s_5, s_1, s_1', s_2', s_3', s_4')$
$AK_{1,5} = K_5$ (same parameters in different permutation)

Ancestor: 2 = C_2

Key derivation of C_6 by C_2:

$j = 6, i = 2$
$S_j/_i = S_j/S_i \cup C_i$
$= \{C_1, C_2\}/\{C_1 \cup C_2\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}1\text{-}2\text{-}0 = 3$
$AK_{2,6} = P(s_6, s_2, s_1, s_1', s_2', s_3')$
$K_6 = (s_6, s1, s2, s_1', s_2', s_3')$
$AK_{2,6} = K_6$ (same parameters in different permutation)

Key derivation of C_7 by C_2:

$j = 7, i = 2$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1, C_2\}/\{C_1 \cup C_2\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}1\text{-}2\text{-}0 = 3$
$AK_{2,7} = P(s_7, s_2, s_1, s_1', s_2', s_3')$
$K_7 = (s_7, s1, s2, s_1', s_2', s_3')$
$AK_{2,7} = K_7$ (same parameters in different permutation)

Key derivation of C_8 by C_2:

$j = 8, i = 2$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1, C_2\}/\{C_1 \cup C_2\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}1\text{-}2\text{-}0 = 3$
$AK_{2,8} = P(s_8, s_2, s_1, s_1', s_2', s_3')$
$K_8 = (s_8, s_1, s_2, s_1', s_2', s_3')$
$AK_{2,8} = K_8$ (same parameters in different permutation)

Key derivation of C_9 by C_2:

$j = 9, i = 2$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1, C_2\}/\{C_1 \cup C_2\}$
$= 0$
$r_j = 0$
$m\text{-} m_i\text{-}2\text{-}r_j = 6\text{-}1\text{-}2\text{-}0 = 3$
$AK_{2,9} = P(s_9, s_2, s_1, s_1', s_2', s_3')$
$K_9 = (s_9, s_1, s_2, s_1', s_2', s_3')$
$AK_{2,9} = K_9$ (same parameters in different permutation)

Ancestor: 3 = C_3

Key derivation of C_{10} by C_3:

$j = 10, i = 3$
$S_j/_I = S_j/S_i \cup C_i$
$= \{C_1, C3\}/\{C_1 \cup C_3\}$
$= 0$
$r_j = 0$

$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{3,10} = P(s_{10}, s_3, s_1,s_1', s_2', s_3')$
$K_{10} = (s_{10}, s_1, s_3, s_1', s_2', s_3')$
$AK_{3,10} = K_{10}$ (same parameters in different permutation)

Key derivation of C_{11} by C_3:

$j = 11, i = 3$
$S_j /_I = S_j/S_i UC_i$
$= \{C_1, C3\}/\{C_1UC_3\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{3,11} = P(s_{11}, s_3, s_1, s_1', s_2', s_3')$
$K_{11} = (s_{11}, s_1, s_3, s_1', s_2', s_3')$
$AK_{3,11} = K_{11}$ (same parameters in different permutation)

Key derivation of C_{12} by C_3:

$j = 12, i = 3$
$S_j /_I = S_j/S_i UC_i$
$= \{C_1, C_3\}/\{C_1UC_3\}$
$= 0$
$r_j= 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{3,12} = P(s_{12}, s_3, s_1, s_1', s_2', s_3')$
$K_{12} = (s_{12}, s_1, s_3, s_1', s_2', s_3')$
$AK_{3,12} = K_{12}$ (same parameters in different permutation)

Ancestor: 4 = C4

Key derivation of C_{13} by C_4:

$j = 13, i = 4$
$S_j /_i = S_j/S_i UC_i$
$= \{C_1, C_4\}/\{C_1UC_4\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{4,13} = P(s_{13}, s_4, s_1, s_1', s_2', s_3')$
$K_{13} = (s_{13}, s_1, s_4, s_1', s_2', s_3')$
$AK_{4,13} = K_{13}$ (same parameters in different permutation)

Key derivation of C_{14} by C_4:

$j = 14, i = 4$
$S_j /_I = S_j/S_i UC_i$
$= \{C_1, C_4\}/\{C_1UC_4\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{4,14} = P(s_{14}, s_4, s_1, s_1', s_2', s_3')$
$K_{14} = (s_{14}, s_1, s_4, s_1', s_2', s_3')$
$AK_{4,14} = K_{14}$ (same parameters in different permutation)

Ancestor: 5 = C5

Key derivation of C_{15} by C_5:

$j = 15, i = 5$
$S_j /_I = S_j/S_i UC_i$
$= \{C_1, C5\}/\{C_1 UC_5\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{5,15} = P(s_{15}, s_5, s_1, s_1', s_2', s_3')$
$K_{15} = (s_{15}, s_1, s_5, s_1', s_2', s_3')$
$AK_{5,15} = K_{15}$ (same parameters in different permutation)

Key derivation of C_{16} by C_5:

$j = 15, i = 6$
$S_j /_I = S_j/S_i UC_i$
$= \{C_1, C_5\}/\{C_1UC_5\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$AK_{5,16} = P(s_{16}, s_5, s_1, s_1', s_2', s_3')$
$K_{16} = (s_{16}, s_1, s_5, s_1', s_2', s_3')$
$AK_{5,16} = K_{16}$(same parameters in different permutation)

A few Examples for the key derivation by the Non Ancestral Class is shown below

Case: 1

Key derivation of C_1 by C_2:

$j = 1, i = 2$
$S_j /_i = S_j/S_i UC_i$
$= \{\Phi\}/\{C_1UC_2\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$NK_{2,1} = P(s_2, s_1, s_1, s_1', s_2', s_3')$
$K_1 = (s_1, s_1', s_2', s_3', s_4', s_5')$
$NK_{2,1} \neq K_1$ (parameters are not correct hence Class C_2 does not get the correct key of Class C_1)

Case: 2

Key derivation of C_1 by C_3:

$j = 1, i = 3$
$S_j /_i = S_j/S_i UC_i$
$= \{\Phi\}/\{C_1UC_3\}$
$= 0$
$r_j = 0$
$m- m_i-2-r_j = 6-1-2-0 = 3$
$NK_{3,1} = P(s_3, s_1, s_1, s_1', s_2', s_3')$
$K_1 = (s_1, s_1', s_2', s_3', s_4', s_5')$

$NK_{3,1} \neq K_1$ (parameters are not correct hence Class C_3 does not get the correct key of Class C_1)

Case: 2
Key derivation of C_9 by C_4:

$j = 9, i = 4$
$S_j /_i = S_j/S_i \cup C_i$
$= \{C_1, C_2\}/ C_1 \cup C_4\}$
$= \{C_2\}$
$r_j = 1$
$m- m_i-2-r_j = 6-1-2-1$
$= 2$
$NK_{4,9} = P(s_4, s_9, s_1, s_2, s_1', s_2')$
$K_9 = (s_9, s_1, s_2, s_1', s_2', s_3')$
$NK_{4,9} \neq K_9$ (parameters are not correct hence Class C_4 does not get the correct key of Class C_9)

3.2. Support of Network Dynamics

In a large group the user dynamics is very frequent which is taken care by the lower layer protocols explained in subsequent sections. There may be occasions where there is a necessity for having changes in the Security Classes during a change in the hierarchy once the system starts functioning. This is managed by the Symmetric Polynomial Scheme in the Upper Layer as explained below.

3.3. Adding a Security Class

When a new security class C_r is added, we need to verify whether m value satisfies the new node restrictions:

- If $m < \max \{m_1, m_2... m_n,..m_r\}+1$, a new m value will be generated so that $m \geq \max \{m_1, m_2, ..., m_n,..m_r\}+1$. Also, the CA will stimulate a new polynomial functions $P(x_1, x_2,…x_m)$ accordingly. In addition, all polynomial functions of security classes are recomputed and retransmitted securely to individual security class controllers
- If $m \geq \max\{m_1, m_2,..., m_n,… m_r\} +1$, the CA selects a random number s_r for the new security class C_r so that a new polynomial function g_r can be computed and transmitted to security class C_r securely. However, if security class C_r is added as a parent security class of any existing security classes, we need to modify keys of C_r's descendant security classes to prevent security class C_r from obtaining old keys of its descendant

3.4. Deleting a Security Class

When a security class C_r is removed from the hierarchy, we need to resolve whether the security class C_r is a leaf node or a parent node. Here, a leaf node a node without any descendant:

- Security class C_r is a leaf node: The CA can plainly discard the public parameter s_r without changing any other keys
- Security class C_r is a parent node: Once security class C_r is deleted from the hierarchy, we cannot allow it to calculate keys of C_r's descendant security classes using polynomial function g_r. We need to thwart security class C_r from accessing its descendants' resources

3.5. Moving a Security Class

A security class C_r can be moved from one node to another node in the hierarchy. There are four cases:

- Leaf node to another leaf node: The CA simply re-computes new polynomial function g_r according the new hierarchy and securely transmits g_r to C_r
- Leaf node to parent node: The CA re-computes polynomial functions of security class C_r and C_r's new descendant security classes according to the new hierarchy. The CA securely transmits polynomial functions to the affected security classes
- Parent node to leaf node: The CA re-computes polynomial functions of previous descendant security classes of C_r and security class C_r according to the new hierarchy and then, securely transmits these polynomial functions to the affected security classes
- Parent node to parent node: The CA re-computes polynomial functions of previous and present descendant security classes of C_r and security class C_r according to the new hierarchy and then, securely transmits these polynomial functions to the affected security classes

3.6. Merging a Security Class

Two or more security classes can merge together and become one security class C_r. Similarly, the CA needs to find previous and present descendant security classes of the merging security classes. The CA randomly chooses a new number S_r and then, generates polynomial functions for all corresponding security classes.

3.7. Splitting a Security Class

A security class C_r splits into two security classes C_{r1} and C_{r2}. Depending on whether C_r is a parent node or leaf node, the CA has to determine what previous and present descendant security classes are associated with these security classes (C_r, C_{r1} and C_{r2}). The CA then selects two new numbers s_{j1} and s_{j2} and generates polynomial functions for these affected security classes.

3.8. Adding a Link

If two security classes C_r and C_k are linked together, we establish a new direct parent-child relationship between two security classes; say security class C_r is the parent of security class C_k. There are two different cases: (1) Security class C_r was an ancestor of security class C_k through other security classes. The CA does not need to perform anything; and (2) security class C_r is the only parent for security class C_k in the new hierarchy. The CA selects a new number S_k and generates new polynomial functions for security class C_k and its descendants security classes. The CA securely transmits new polynomial functions to these affected security classes.

3.9. Deleting a Link

If two linked security classes C_r and C_k are disconnected, we destroy a direct parent-child relationship between two security classes; say security class C_r will not be the parent of security class C_k in the new hierarchy. Again, there are two different cases: (1) Security class C_r is still an ancestor of security class C_k through other security classes in the new hierarchy. The CA does not need to perform anything; and (2) security class C_r is not an ancestor for security class C_k in the new hierarchy. The CA selects a new number S_k and generates new polynomial functions for security class C_k and its descendants security classes. The CA securely transmits new polynomial functions to these affected security classes.

4. KEY ESTABLISHMENT WITHIN THE DYNAMIC PEER GROUPS

Tree based Group Elliptic Curve Diffie-Hellman (TGECDH) protocol is used for maintaining the Classes (C_1, C_2, C_3, C_4, C_5, C_6 and C_7) individually. The key establishment in the class C_6 is shown. The same procedure is used in all the Classes. Each Member contributes the partial key to compute the class key. In this section, an example of the TGECDH key establishment scheme has been discussed. This example shows how the shared key is obtained by the members and the class key is computed in the group consisting of four SixM1, SixM2, SixM3 and $SixM_4$.

In the class (e.g., C_6), initially two members $SixM_1$& $SixM_2$ are available (**Fig. 3**). If a new member $SixM_3$ wants to join the class (**Fig. 4**), it broadcasts a join request message to class controller.

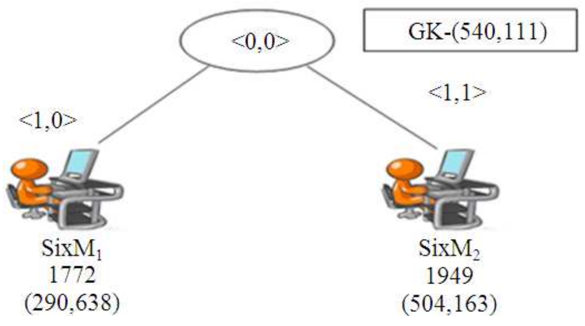

Fig. 3. Member $SixM_1$ and $SixM_2$ join the class

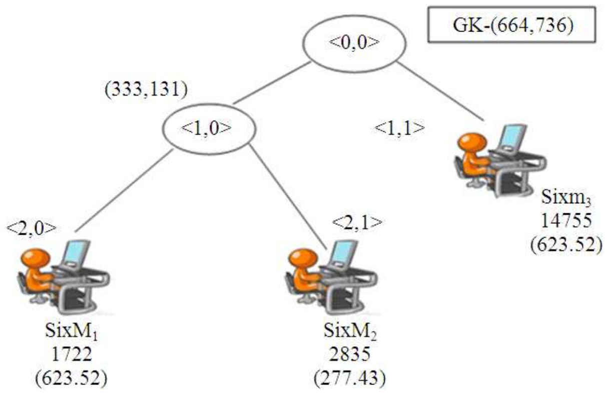

Fig. 4. Member SixM3 join the class

The class controller receives this message and determines the insertion point in the tree. If a member joins in the shallowest rightmost node there, it does not increase the height of the key tree.

If the key tree is fully balanced, the new member joins the root node. The class controller is the rightmost leaf in the sub tree rooted at the insertion node. When a member joins in the class, it creates a new intermediate node and promotes the new intermediate node to be the parent of both the insertion node and the new member node. After updating tree, the class controller proceeds to update its share and passes all public keys tree structure to new member.

The new member acts as the new class controller and computes the new class key. Next, the class controller broadcasts the new tree that contains all public keys. All other members update their trees accordingly and compute the new class key.

If a member wants to leave the class, first it should send the leave request to the class controller to generate the new class key. When the leave request message is received by class controller, it updates its key tree by deleting the leaf node corresponding to leave member. The former sibling of leave member is promoted to parent node. The class controller generates a new private key share, computes all public key pairs on the key-path up to the root and broadcasts the new key tree that contains all public keys. The entire member in the class computes the new class key as:

$$BK_{<l,v>} = K_{<l,v>} * G \quad K_{<l,v>} = r_v * G$$

Where:

$K_{<l,v>}$	=	The private key
$BK_{<l,v>}$	=	The public key
r_v	=	A random number and
G	=	The Generator

The intermediate node with two children does not represent any class member but it represents a sub-class. The intermediate node's private key is treated as the sub-class key. It can be calculated by the following rule where node <l, v>'s two children are <l+1, 2v> and < l+1, 2v+1>. Where, l is the level, v is the vertices index, $K_{<l,v>}$ can be calculated as $X_{co}(K_{<l+1,2v>}*BK_{<l+1,2v+1>})$. This can be solved as:

$$X_{co}\left(K_{<l+1,2v>} * BK_{<l+1,2v+1>}\right) = X_{co}\left(K_{<l+1,2v+1>} * BK_{<l+1,2v>}\right)$$
$$= X_{co}\left(K_{<l+1,2v>} * K_{<l+1,2v+1>} * G\right)$$

where, $X_{co}$is the x-coordinate of the point represented within the parentheses. L is the height (level) of the node and v is the index of the node at level l.

4.1. Initializing the Outer Group

Initially two Gateway Member $SixM_1$ and $SixM_2$ are available in the class. $SixM_1$ and $SixM_2$ are the members of class that is going to exchange their keys. Consider p = 751, Ep (1,188), which is equivalent to the curve $y^2 = x^3+x+188$ and G = (0,376). Member $SixM_1$'s private key is 1772 and its public key is (290, 638). Member $SixM_2$'s private key is 1949 and its public key is (504, 163), $K_{<1,0>}$ is 1772, $BK_{<1,0>}$ is $K_{<1,0>}$*G and $K_{<1,0>}$*G can be calculated as:

$$\begin{aligned} K_{<1,0>}*G &= (1772 \bmod 769)\ G \\ &= 234*(0,376) \\ &= (290, 638) \end{aligned}$$

Similarly,

$$\begin{aligned} K_{<1,1>} &= 1949 \\ BK_{<1,1>} &= K_{<1,1>}*G \\ &= (1949 \bmod 769)\ G \\ &= 411*(0,376) \\ &= (504, 163) \end{aligned}$$

The Class Key is computed as Member $SixM_1$ sends its public key (290,638) to Member $SixM_2$; the Member $SixM_2$ computes its Class key as:

$$\begin{aligned} K_{<0,0>} &= X_{co}(K_{<1,1>}*BK_{<1,0>}) \\ &= X_{co}(1772*(504,163)) \\ &= X_{co}(1772*411 \bmod 769)G \\ &= X_{co}(210,591) = 210 \\ BK_{<0,0>} &= K_{<0,0>}*G \\ &= 210G \\ &= 210*(0,376) \\ &= (540,111) \\ {}_{<1,1>} &= 14755 \\ BK_{<1,1>} &= K_{<1,1>}*G \\ &= (14755 \bmod 769)*G \\ &= 144G = 144*(0,376) \\ &= (623, 52) \end{aligned}$$

Compute the class key:

$$\begin{aligned} K_{<0,0>} &= X_{co}(K_{<1,1>}*BK_{<1,0>}) \\ &= X_{co}(144*(333,131)) \\ &= X_{co}(144*149G) \\ &= X_{co}(337,192) \\ &= 337. \\ BK_{<0,0>} &= K_{<0,0>}*G \\ &= 337G = 337*(0,376) \\ &= (664,736) \end{aligned}$$

$SixM_3$ sends the public key tree values to all users. Now member $SixM_1$ and $SixM_2$ compute their class key:

Member node <2, 0> and <2, 1>

$$\begin{aligned} K_{<0,0>} &= X_{co}(K_{<2,0>} * BK_{<1,1>}) \\ &= X_{co}(149*(623,52)) \\ &= X_{co}(149*144G) = X_{co}(337,192) = 337. \\ BK_{<0,0>} &= K_{<0,0>}*G \\ &= 337G = 337*(0,376) \\ &= (664,736) \end{aligned}$$

4.2. Member SixM4 Joins the Outer Group

When a new Member $SixM_4$ joins the Class, the previous Class controller $SixM_3$ changes its private key value from 14755 to 8751 and passes the public key tree to Member $SixM_4$. New private key is $K'_{<2,2>}$ can be calculated as:

$$\begin{aligned} K'_{<2,2>} &= 8751 \\ BK_{<2,2>} &= K'_{<2,2>}*G \\ &= (8751 \bmod 769)*G \\ &= 292G \\ &= 292*(0,376) \end{aligned}$$

$$\begin{aligned}
&= (7,177)\\
K_{<1,1>} &= X_{co}(K_{<2,2>}*BK_{<2,3>})\\
&= X_{co}(9751*(725,224))\\
&= X_{co}(9751*122G)\\
&= X_{co}(675,243)\\
&= 675.\\
BK_{<1,1>} &= K_{<1,1>} * G\\
&= 675G = 675 * (0,376)\\
&= (127,150)
\end{aligned}$$

Now, $SixM_4$ becomes new Class controller. Then, $SixM_4$ generates the public key (725, 224) from its private key as 48569 and computes the Outer group key as (641,685) shown in **Fig. 5**:

$$\begin{aligned}
K_{<2,3>} &= 48569\\
BK_{<2,3>} &= K_{<2,3>}*G\\
&= (48569 \bmod 769)\ G\\
&= 122G\\
&= 122 * (0,376)\\
&= (725, 224)\\
K_{<1,1>} &= X_{co}(K_{<2,3>}*BK_{<2,2>})\\
&= X_{co}(9751*(725,224))\\
&= X_{co}(9751*122G)\\
&= X_{co}(675,243)\\
&= 675.\\
BK_{<1,1>} &= K_{<1,1>} * G\\
&= 675G\\
&= 675* (0,376)\\
&= (127,150)
\end{aligned}$$

The class key is computed as follows:

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,1>}*BK_{<1,0>})\\
&= X_{co}(675* (333,131))\\
&= X_{co}(149*675G)\\
&= X_{co}(355,103)\\
&= 355\\
BK_{<0,0>} &= K_{<0,0>} * G\\
&= 355G\\
&= 355* (0,376)\\
&= (641,685)
\end{aligned}$$

$SixM_4$ sends public key tree to all members. Now, Member $SixM_1$, $SixM_2$ and $SixM_3$ compute their class key. Member node <2,0> and <2,1>:

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,0>} * BK_{<1,1>})\\
&= X_{co}(149*675G)\\
&= X_{co}(605G)\\
&= X_{co}(355,103)\\
&= 355\\
BK_{<0,0>} &= K_{<0,0>} * G\\
&= 355G\\
&= 355* (0,376)\\
&= (641,685).
\end{aligned}$$

Member node <2,2 >

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,1>}*BK_{<1,0>})\\
&= X_{co}(675* (333,131))\\
&= X_{co}(149*675G)\\
&= X_{co}(355,103)\\
&= 355\\
BK_{<0,0>} &= K_{<0,0>} * G\\
&= 355G\\
&= 355* (0,376)\\
&= (641,685)
\end{aligned}$$

4.3. Leave Operation

In the individual classes either the member or the class controller may leave.

4.4. Member Leave

When Member $SixM_3$ leaves (**Fig. 6**) the class, then the Class Controller $SixM_4$ changes its private key 48569 to 98418 and class key is recalculated as (28,686):

$$\begin{aligned}
K_{<1,1>} &= 98418\\
BK_{<1,1>} &= K_{<1,1>} * G\\
&= (98418 \bmod 769)\ G\\
&= 755G = 755* (0,376)\\
&= (383,702)
\end{aligned}$$

Now member $SixM_4$ will move to level <1,1>. After that, it broadcasts its public key tree to all Members in the Class. Then, the new Class key will be generated by the remaining Members:

Member node <1,1>

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,1>}*BK_{<1,0>})\\
&= X_{co}(755*(333,131))\\
&= X_{co}(755 * 149G)\\
&= X_{co}(579,363)\\
&= 579\\
BK_{<0,0>} &= K_{<0,0>}*G\\
&= 579G = 579* (0,376)\\
&= (428,686)
\end{aligned}$$

Member node <2,0> and <2,1>

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,0>}*BK_{<1,1>})\\
&= X_{co}(149*(383,702))\\
&= X_{co}(149*755G)\\
&= X_{co}(579,363)\\
&= 579\\
BK_{<0,0>} &= K_{<0,0>}*G\\
&= 579G\\
&= 579* (0,376)\\
&= (428,686)
\end{aligned}$$

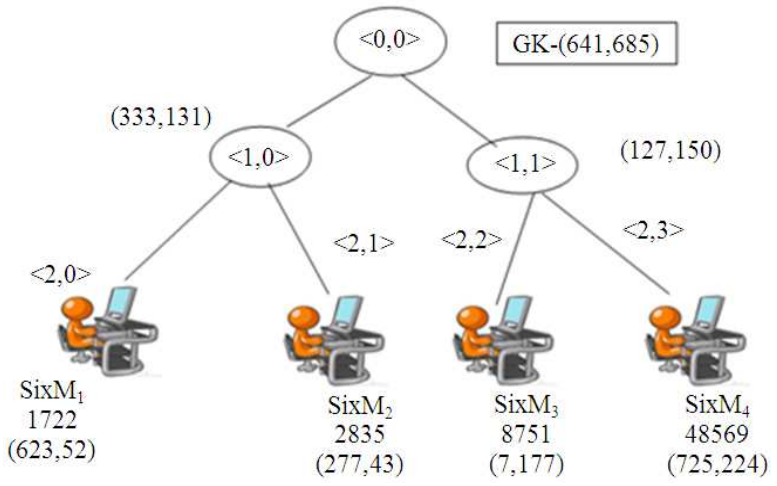

Fig. 5. Member SixM4 joins the class

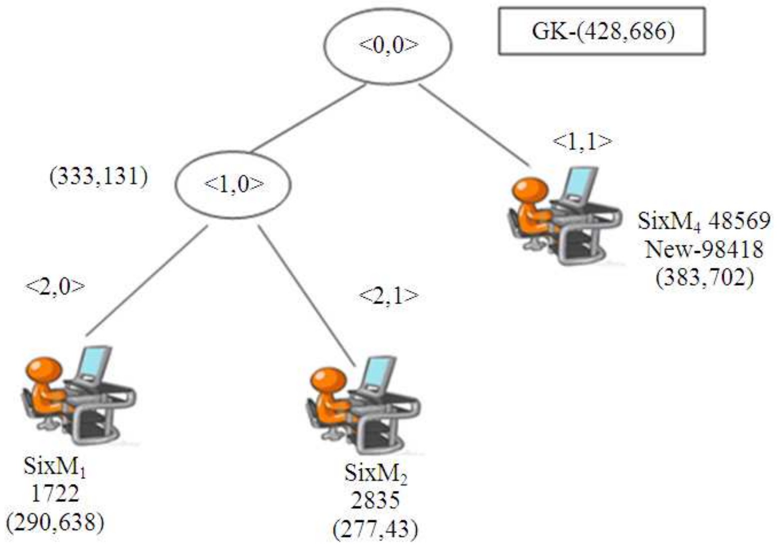

Fig. 6. Member sixM3 leaves the Outer group

4.5. Class Controller Leaves

When Class Controller SixM$_4$ leaves (**Fig. 7**) the Class, then its sibling act as a Class Controller (SixM$_3$) and changes its private key value 8751 to 19478 and recalculates the class key as (681,475):

$$
\begin{aligned}
K_{<1,1>} &= 19478\\
BK_{<1,1>} &= K_{<1,1>}{*}G\\
&= (19478 \bmod 769)\ G\\
&= 253G\\
&= 253{*}\ (0,376)\\
&= (303,673)
\end{aligned}
$$

Compute the Class Key:

$$
\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,1>}{*}BK_{<1,0>})\\
&= X_{co}(253{*}\ (333,\ 131))\\
&= X_{co}(253 * 149G)\\
&= X_{co}(16\ G)\\
&= X_{co}(614,236)\\
&= 614\\
BK_{<0,0>} &= K_{<0,0>} * G\\
&= 614G = 614{*}\ (0,376)\\
&= (681,457)
\end{aligned}
$$

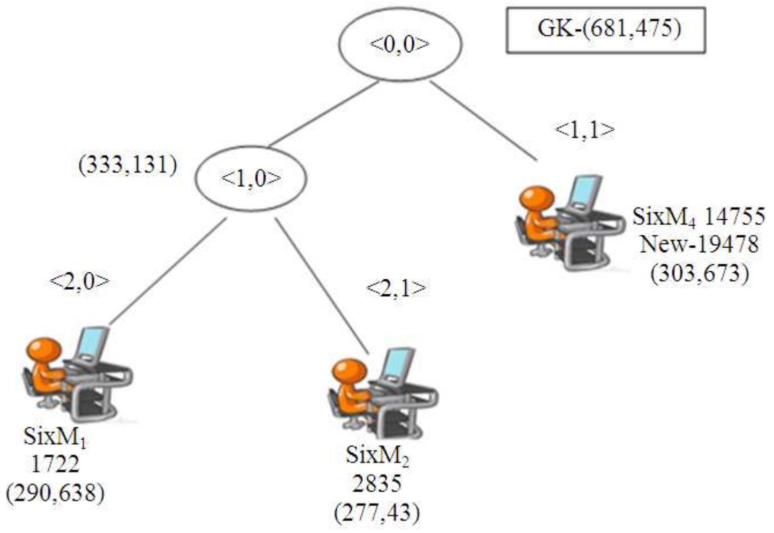

Fig. 7. Class controller leaves the Class

After that, it broadcasts its public key value of the tree to all members in the Class. Then, the new Class key will be generated by the remaining members:

Member <2,0> and <2,1>

$$\begin{aligned}
K_{<0,0>} &= X_{co}(K_{<1,0>} * BK_{<1,1>}) \\
&= X_{co}(149*(303,673)) \\
&= X_{co}(149*253G) \\
&= X_{co}(16\ G) \\
&= X_{co}\ (614,236) \\
&= 614. \\
BK_{<0,0>} &= K_{<0,0>} * G \\
&= 614\ G = 614 * (0,376) \\
&= (681,475)
\end{aligned}$$

The above scheme is used for each and every class and they may use any elliptic curve for the local communication.

5. PERFORMANCE ANALYSIS OF DEP (SP-TGECDH) PROTOCOL

The following Evaluation Criteria is used for analyzing the performance of the proposed scheme:

- Size of the information stored
- Amount of public information
- Efficiency of key derivation by the ancestor classes
- Communication complexity of key updates
- Computational Complexity
- Security against attacks

5.1. Size of Information (Public and Private)

Storage overhead can be considered as the memory capacity required for maintaining the keys, which is directly proportional to the number of members. The total storage required can be calculated as sum of cost incurred for the TGECDH protocol and the Symmetric Polynomial. Suppose there are C classes. Total Storage cost = $\sum TGECDH_i$+Cost for Symmetric Polynomial. Storage Cost (Private Keys +Public Keys) of TGECDH for a single class = $(2\times n_i) + n_i+1$. Where, n_i is the number of members in the class i and TGECDH Cost for c classes is $\Sigma\ ((2\times n_i)+ n_i+1)$ for i = 1 to c:

$$\text{Total Storage Cost} = \Sigma_{i=1,c}\left(\left(2 \text{ x } n_i\right) + n_i + 1\right) + 1 + 2\times m$$

where, m is the number of public parameters. In our case it is 6. The Class Controllers forming a part of Symmetric Polynomial Scheme need to store only one key. The **Table 2** shows the key size for an equivalent security using normal schemes and ECC.

where, K_i is Private Key size in bits PK_u is Public key size in bits. Always public key size is twice that of private key in ECC. The ECC offers a very high security with very less key size and hence is more suitable for implementing the hierarchical access control on devices with low power.

5.2. Efficiency of Key Derivation By The Ancestor Classes

One common operation for HAC is key derivation, which is a node develops the key of its descendant from its own key.

Table 2. Key size for equivalent security

ECC key length public key				
Prime field	Binary field K_i	Public key Pk_u	RSA key length for approximate equivalent security	Private key length for approximate equivalent security
112	113	224	512	56
128	131	256	704	64
160	163	320	1024	80
192	193	384	1536	96
224	233	448	2048	112
256	283	512	3072	128
384	409	768	7680	192
521	571	1042	15360	256

This origin will follow the pathway from the node to the descendant and use the one-way function iteratively. The longer path increases the complexity of the key derivation. The worst case is by O(n). In the case of the proposed scheme the class controllers of the ancestor classes are able to derive the key of the descendant class and use of asymmetric keys that is separate key for the two layers make the scheme adapt to dynamic membership changes very efficiently.

5.3. Communication Cost

Communication Cost depends on Number of Rounds, Number of Messages and Size of a message. Communication costs needed for the group key agreement protocol in terms of number of messages. In each class a join operation requires 2 rounds of message and 3 messages are exchanged to form a key. The leave operation needs only a single message. As the membership changes are local to the respective classes the key for that class alone is changed and the remaining keys of all other classes do not get affected. Dynamic Peer Groups are efficiently managed by this method.

5.4. Computation Cost

The computation cost is the cost involved in the case of calculating the key. In the proposed scheme, only during the changes in hierarchy the key is recalculated by the Central Authority and distributed to the class controllers. In the proposed scheme, asymmetric keys are used between communicating nodes. The class dynamically establishes keys using TGECDH. This offers distinct advantages for key establishment including scalability. ECC is an ideal public key algorithm because it offers the most security per bit than any other public key scheme.

The Total Cost for the proposed scheme is the sum of the cost for Tree Based Group Elliptic Curve Diffie Hellman Scheme and Symmetric Polynomial Scheme. The TGECDH can be calculated as total number of point operations.

Table 3. Execution Time (µs) of field operations in Fp192, F_{p224}, F_{p256}, F_{p384}, F_{p521}

	F_{p192}	F_{p224}	F_{p256}	F_{p384}	F_{p521}
Addition	0.071	0.160	0.083	0.142	0.145
Subtraction	0.088	0.162	0.099	0.137	0.146
Reduction	0.216	0.200	0.200	0.270	0.216
Multiplication	1.500	3.100	3.100	7.800	10.900
Squaring	2.350	2.350	3.150	6.250	8.600
Inversion	150.000	160.000	160.000	310.000	620.000

The following operations take place point Addition P+Q (ADD), point Doubling (DBL), number of field operations: Addition/subtraction (A), Multiplication (M), Squaring (S), Inversion (I). The Common assumptions (from **Table 3**) for high-level estimation is A = 0, S = 0.8 M, I = 60 M, total operations are approximately 62 M (Aparna and Amberker, 2009). The operations are done parallel in the respective classes without affecting other classes. If there are t classes, the symmetric polynomial scheme requires:

$$k * t\,\text{exponentiation},\ t * \binom{k+t-2}{t-2}$$

$$\text{Multiplication}\binom{k+t-2}{t-2}$$

where, k is the threshold. For a Pentium 4 processor at 3 GHz., the number of clock cycles for Addition = 3 clock cycles = (3/3)/1000 = 0.001 µs. Multiplication: 10 clock cycles = (10/3)/1000 = 0.003333333 µs. The Time for bit operations for symmetric polynomial scheme is 0.003 µs. The bit operations have been calculated taking into consideration the key sizes for equivalent security as that for ECC. The timings of prime field operations which are addition, subtraction, modular reduction, multiplication, squaring and inversion are given. These values have been used in calculating the computation time for TGECDH.

Optimum storage cost, communication cost and computation cost makes Hierarchical Access Control in Dynamic Peer Groups using Symmetric Polynomial and Tree Based Group Elliptic Curve Diffie Hellman Scheme ideal for use.

5.5. Hierarchical Access Control Cost

The HAC cost is the number of keys to be transmitted to make the higher class users to see the resources or messages of the lower class users. For a class assume that there are m ancestor classes. The number of users in the Ancestor class are denoted by $H_1, H_2, \ldots, H_m$. Irrespective of the number of users in each class, the proposed scheme uses only one decryption and encryption.

5.6. Security Analysis

The Security of SP-TGECDH is good because of the following properties. 1. The use of symmetric polynomial scheme makes key derivation easier. 2. The Two layer approach allows the secret key to be confined to the respective classes alone and the actual key is never moved on to any other class. The system is developed using java net beans and found to be secure and fast. The system takes care of User level and Class level dynamics (Begum *et al*., 2010b). The large number of parameters prevents a possible guessing e.g., For a sixteen parameter general polynomial 16! (i.e., 10922789888000) combinations are possible. Surges in leave and join operations also can be taken care and the system can be used for any hierarchy. The Security of ECC is due to the discrete logarithms problem over the points on the elliptic curve. Cryptanalysis involves determining x given Q and P where P is a point on the elliptic curve and Q = x P that is P added to itself x times. The best known algorithm to break the elliptic curve points is the pollard-rho algorithm which is a fully exponential algorithm and difficult to solve.

5.7. Performance against Attacks

Attack 1: Contrary Attacks

Assuming that E1 (lower privileged user) needs to crack the secret key of B1 (Higher Privileged User). It is not feasible to decrypt messages as the derivation gives a wrong value.

Attack 2: Interior Collecting Attacks

There is no relation bound between any of the ancestor nodes and so a lower level User cannot decrypt messages by negotiating any one parent.

Attack 3: Exterior Collecting Attack

If an attacker is outside the system, it means no idea about what elliptic curve or generator point is being used is known and hence more difficult to attack.

Attack 4: Collaborative Attacks

We assume that if there is a higher privileged user belonging to class B and there are two descendant classes D and E. Users of D and E cannot perform a collaborative attack as the secret key cannot be derived.

Attack 5: Sibling Attacks

Classes who have same parent also cannot crack the key of a sibling class due to the absence of any related parameters among them.

6. CONCLUSION

In this study the Hierarchical Access Control in Dynamic Peer Groups using Symmetric Polynomial and Tree Based Group Elliptic Curve Diffie Hellman Scheme is proposed and implemented. This can enhance the access control performance by using multiple class keys and in contrast to other existing schemes using only single key, the new proposed scheme exploits asymmetric key, i.e., multiple outer keys and multiple class keys. Compared with other schemes, the new proposed scheme can significantly reduce the key computation cost. Therefore, the number of re-keying messages and the load on computation, communication and memory can be dramatically reduced and communication overheads in the re-keying process can be performed, with acceptable computational overhead.

6.1. Future Work

The future work involves use of this approach for real time applications and to provide wide-ranging analysis on network performance constraints such as latency, bandwidth, utilization and throughput. Different channel properties and different topologies need to be investigated to discover further useful interactions. Also, more studies have to be carryout to identify the best topology combinations to achieve high security at the least expense.

7. REFERENCES

Aparna, R. and B.B. Amberker, 2009. Analysis of key management schemes for secure group communication and their classification. J. Comput. Inf. Technol., 17: 203-214.

Begum, N.J., K. Kumar and V. Sumathy, 2010b. Design and implementation of multilevel access control in synchronized audio to audio steganography using symmetric polynomial scheme. J. Inf. Security., 1: 29-40. DOI: 10.4236/jis.2010.11004

Begum, N.J., K. Kumar and V. Sumathy, 2010a. A novel approach towards multilevel access control for secure group communication using symmetric polynomial based elliptic curve cryptography. Proceedings of the International Conference on Computational Intelligence and Communication Networks, Nov. 26-28, IEEE Xplore Press, Bhopal, pp: 454-59. DOI: 10.1109/CICN.2010.92

Das, M.L., A. Saxena, V.P. Gulati and D.B. Phatak, 2005. Hierarchical key management scheme using polynomial interpolation. SIGOPS Operat. Syst. Rev., 39: 40-47. DOI: 10.1145/1044552.1044556

Kim, Y., A. Perrig and G. Tsudik, 2004. Tree based group key agreement. ACM Trans. Inf. Syst. Secu., 7: 60-96. DOI: 10.1145/984334.984337

Kuang, T.P., H. Ibrahim, N.I. Udzir and F. Sidi, 2011. Security extensible access control markup language policy integration based on role-based access control model in healthcare collaborative environments. Am. J. Econ. Bus. Admin., 3: 101-111. DOI: 10.3844/ajebasp.2011.101.111

Lin, I.C., M.S. Hwang and C.C. Chang, 2003. A new key assignment scheme for enforcing complicated access control policies in hierarchy. Future Generat. Comput. Syst., 19: 457-462. DOI: 10.1016/S0167-739X(02)00200-5

Hwang, M.S. and W.P. Yang, 2003. Controlling access in large partially ordered hierarchies using cryptographic keys. J. Syst. Softw., 67: 99-107. DOI: 10.1016/S0164-1212(02)00091-2

Hwang, S. and S.E. Satchell, 1999. Modelling emerging market risk premia using higher moments. Int. J. Finance Econom., 4: 271-296. DOI: 10.1002/(SICI)1099-1158(199910)4:4<271::AID-IJFE110>3.0.CO;2-M

Wang, Y., B. Ramamurthy and X. Zou, 2006. The performance of elliptic curve based diffie-hellman protocols for secure group communication over adhoc networks. Proceedings of the IEEE International Conference on Communications, Jun. 11-15, IEEE Xplore Press, Istanbul, pp: 2243-2248. DOI: 10.1109/ICC.2006.255104

Zhong, S., 2002. A practical key management scheme for access control in a user hierarchy. Comput. Secu., 21: 750-759. DOI: 10.1016/S0167-4048(02)00815-5.

Zou, X. and L. Bai, 2008. A new class of key management scheme for access control in dynamic hierarchies. Int. J. Comput. Appli., 30: 331-337.