Journal of Computer Science 7 (3): 328-340, 2011
ISSN 1549-3636

A New Scalable and Reliable Cost Effective Key Agreement Protocol for Secure Group Communication

[1]S. Jabeen Begum and [2] Dr. T. Purusothaman
[1]Department of CSE, Velalar College of Engineering and Technology, Erode-12, India
[2]Department of CSE, Government College of Technology, Coimbatore-13, India

Abstract: Problem statement: In a heterogeneous environment, for a secure multicast communication, the group members have to share a secret key which is used to encrypt/decrypt the secret messages among the members. The Secure Group Communication of large scale multicast group in a dynamic environment is more complex than securing one-to-one communication due to the inherent scalability issue of group key management. Since the group members are dynamic in nature such as joining or leaving the group, the key updating is performed among the valid members without interrupting the multicast session so that non group members can't have access to the future renewed keys. **Approach:** The main aim is to develop a scheme which can reduce the cost of computational overhead, number of messages needed during the time of key refreshing and the number of keys stored in servers and members. The cost of establishing the key and renewal is proportionate to the size of the group and subsequently fetches a bottleneck performance in achieving scalability. By using a Cluster Based Hierarchical Key Distribution Protocol, the load of key management can be shared among dummy nodes of a cluster without revealing the group messages to them. **Results:** Especially, the existing model incurs a very less computational and communication overhead during renewal of keys. The proposed scheme yields better scalability because of the fact that the Key computational cost, the keys stored in key server and numbers of rekey-messages needed are very less. **Conclusion:** Our proposed protocol is based on Elliptic curve cryptography algorithm to form secure group key, even with smaller key size, it is capable of providing more security. This protocol can be used both in wired or wireless environments.

Key words: Clustering process, secure group communication, key management, Group Session Key (GSK), Domain Key (DK), complexity analysis

INTRODUCTION

The exponential growth of the Internet for the last few years along with the relative increase in bandwidth of networks has resulted in the development of new services. Although the unicast communication has been dominant so far, the need for multicast communication is mandatory both in the perspective of Internet Service Providers and Distributors. The Key management plays a pivotal role in providing the common security services such as authentication and integrity for a group communication. The secure group communication provides both secure multipoint communication and point-to-point communication. The encryption of the point-to-point messages is made with a key shared by members both ends. The encryption of Multicast messages is made with the help of the group key. The main aim is to elaborate how provable and promising is our proposed secure group key management protocol when combined with the reliable group communication services in obtaining a cost effective computational strategy. For the establishment of group communication, a single common group key is distributed to every member of the group which is highly dynamic in heterogeneous environment and the key is refreshed whenever a member joins or leaves the group. Three main ways of the group key management are Centralized Group Key Management, Distributed Group Key Management and Decentralized Group Key Management.

The Centralized key management is employed for controlling the entire group. Hence, a centralized key management tries to minimize requirements of the storage and computational power for both the client and server. However the problem of single point failure remains existing in this mode of key management. The Protocols mostly used in Centralized Group Key Management are OFT, LKH, ELK and CFKM, GKMP, Keystone.

Corresponding Author: S. Jabeen Begum, Department of CSE, Velalar College of Engineering and Technology, Erode-12

In distributed key management architecture, there is no external Key Distribution Centre to distribute the key. The key generation is performed by the members themselves. The members who want to be independent of third party intervention can do the access control operations and take part in the group key generations. Thus, the security can be enhanced in this method. However it is restricted to only a small group of members in which the collection of the contributions of each and every user is meticulous and time consuming and hence the scalability criterion is not fulfilled. The typical Protocols used in distributed key management are CKA, STR, Octopus and DH−LKH.

In a decentralized architecture, a large group is managed by dividing it among the subgroup managers. It minimizes the problem of focusing the entire task at one particular location. The typical Protocols used in decentralized environment are SMKD and IGKMP.

An Efficient group key management protocol demands a few miscellaneous requirements such as Quality of Service, security and the resources of the group members. The general attributes in Group Key Management are as follow:

Forward Secrecy: It ensures that a member who has left the group should not be able to decrypt the data of his old group.

Backward secrecy: It ensures that a member who has newly joined the group should not be able to decrypt the previous data of the group.

Collusion freedom: It ensures that no fraudulent user can acquire the group key.

Key independence: It is a property of a protocol stating the non compromising nature of the key disclosure.

Minimal trust: It ensures that the Key Management scheme should provide trust only to limited number of entities.

In order to accomplish these aspects, a partial Distributed and Decentralized Architecture is proposed.

Related work: It is generally assumed that the operation of rekeying has to be performed in multicasting, whenever multicasting is used for group transmission (Pour *et al*., 2007; Wong *et al*., 2000). Using a scalable multicast communication, it is not reasonable to consider that transmitting data to the members and rekeying the members under a non-scalable peer to peer communication. If the group has large members, sending them a new key one by one will not be efficient. Although rekeying (Mao *et al*., 2004) a group before joining a new member is trivial, rekeying the group after a member leaves will be far more complicated. The old key cannot be distributed to a new member, because the leaving member has already known the old key. A group key distributor must therefore provide other mechanisms to rekey the group using multicast messages with maintaining the highest level of possible security.

In the centralized system, there is only one entity to control the whole group. The central controller does not have to rely on any auxiliary entity to perform access control and key distribution operations. The central server may undergo the problem of single point failure with only one managing entity. If there is a problem with the controller, then the entire group will be affected.

In Group Key Management Protocol, the KDC (Lee and Shieh, 2004; Al-Talib *et al.*, 2009; David Manz *et al.*,2010) helps the first member to join the group and creates a Group Key Packet (GKP) that consists of a Group Traffic Encryption Key (GTEK) and a Group Key Encryption Key (GKEK). The KDC sends a copy of the GKP whenever a new member wants to join the group. As all members know the GKEK, there is no chance of maintaining the forward secrecy intact when a member leaves the group. Therefore key for entire group has to be renewed.

In Logical Key Hierarchy, the KDC maintains a tree of keys. The nodes of the tree hold key encryption keys. The leaves of the tree correspond to group members and each leaf holds a KEK (Saroit *et al*., 2009) associated with one group. Each member receives and maintains a copy of the KEK associated with its leaf and the KEKs corresponding to each node in the path from its parent node to the root. For a balanced tree, each member stores at most ($\log_2 n$) +1 keys, where ($\log_2 n$) is the height of the tree.

The One-way Function Tree (OFT) scheme (Kim *et al*., 2005; Poovendran and McGrew, 2004; Rafaeli and Hutchison, 2003) is an improvement over the hierarchical binary tree, which reduces the size of the rekeying message from $2(\log_2 n)$ to only ($\log_2 n$). The KEKs held by a node's children are blinded using a one-way function and then mixed together using a mixing function. The result of this mixing function is the KEK held by the node.

One-way Function Chain Tree is a different approach that undergoes the same communication overhead. This scheme uses a pseudo-random-generator (Micciancio and Panjwani, 2008; Rafaeli and Hutchison, 2003) to generate a new KEK rather than using a one-way function and then it is applied only on user removal. This scheme is known as the one-way

function chain tree. The pseudo-random-generator, G(x), doubles the size of its input (x), the output of G(x) is represented as two functions, L(x) and R(x) that are the left and right halves of G (x.) (i.e., G(x) = L(x) R(x))

The Distributed Key Management approach is characterized by having no group controller. The group key can be either generated in a contributory fashion, where all members contribute their own share to computation of the group key, or generated by one member. Although it is fault-tolerant, it may not be safe to leave any member to generate new keys since key generation requires secure mechanisms, such as random number generators, that may not be available to all members. Moreover, in most contributory protocols, processing time and communication requirements increase linearly (Yi, 2005; Sundaram Sudha *et al.*,2009) in term of the number of members.

In Distributed Logical Key Hierarchy, the GC (Kulkarni and Bruhadeshwar, 2010) is completely abolished and the logical key hierarchy is generated among the members, therefore there is no entity that knows all the keys at the same time. This protocol uses the notion of sub trees agreeing on a mutual key. That is, two groups of members namely sub tree L and sub tree R, agree on a mutual encryption key. Assuming that member ml is to be L's leader and member mr is to be R's leader. The Sub tree L has sub tree key kL and the sub tree R has sub tree key kR.

In Diffie–Hellman Logical Key Hierarchy, a logical key hierarchy is used to minimize the number of keys held by group members. The main difference here is that group members generate the keys in the upper levels using the Diffie–Hellman algorithm (Zheng *et al*., 2006; Amir *et al*., 2004) rather than using a one-way function. The key of each node is generated from its two children ($k=\alpha^{k1k2} \bmod p$).

In Conference Key Agreement (CKA) where all group members contribute to generate the group key. The group key can be generated with a combining function: $K = f(h(N_1), h(N_2),\ldots, h(N_n))$, where f is the combining function, h is a one-way function, n is the group size and N_i is the contribution from group member i. The protocol specifies that n - 1members broadcast their contributions (N_i).

In Decentralized Key Management, the large group is split into small subgroups. Different controllers are used to manage each subgroup, minimizing the problem of heaping the work on a single location. In Scalable Multicast Key Distribution, the trees built by the Core Based Tree (CBT) multicast routing protocol are to deliver keys to a multicast group. Any router in the path of a joining member from its location to the primary core can authenticate the member since the router is authenticated with the primary core. Furthermore, there is no solution for breach of forward secrecy other than recreating an entirely a new group without the leaving members. In Intra-Domain Group Key Management scheme, there are a Domain Key Distributor (DKD) and many Area Key Distributors (AKD) (Rafaeli and Hutchison, 2003; Al-Saadoon *et al*., 2009). Each AKD is responsible for his respective area. The group key is generated by the DKD and is propagated to the members through the AKDs The key managers (DKD and AKD) are placed in a multicast group, named All-KD-group .The All- KD-group is used by the DKD to transmit the rekey messages to the AKDs. All areas in the domain use the same group key. Therefore, data packets do not need to be translated when passing from one area to another. Moreover, if an AKD is unavailable, no member in that area is able to access the group communication, since they will not be able to access AKDs from other areas.

A group of nodes is called Cluster where one node acts as Cluster head which is responsible for some specific tasks. Each cluster is formed around a representative called Cluster Head. According to a well defined criterion, Cluster Heads are selected. A cluster is assigned with an identifier that is related to its representative (i.e. its cluster head). Each node in the network carries the cluster identifier to which it belongs. The hierarchy is built based on the capabilities of nodes. To form clusters, a new message called CIA (Cluster Id Announcement) is periodically sent by cluster heads to declare their leaderships and invite other nodes to join their clusters.

In key management algorithms (Prathap and Vasudevan, 2009; Poovendran and McGrew, 2004; Rafaeli and Hutchison, 2003; Zheng *et al*., 2006) when group membership changes, the group controller changes the keys in the key tree and securely broadcasts the new keys to other existing members. The group controller broadcasts all the key updates which are encrypted with shared keys known only to a subset of users in the group. Since all users do not need all the key updates, his mode of key distribution is not efficient. Focusing on the key distribution using these algorithms where each user receives only a small subset of keys that includes all the keys it needs. Towards this end, the forwarding mechanism is modified at the intermediate nodes; an intermediate node forwards a key update message only if it believes that there are descendant users who need this key update. In this approach, an intermediate node performs this check by verifying that any of its descendants know the key with which the key update message is encrypted. The keys known to a user depend on the type of group key management algorithm used.

MATERIALS AND METHODS

Now we will briefly discuss about the Secure Group Communication Protocol (SGCP) and the new designed protocol architecture.

Initialization and updation on clusters: To generate a Cluster Based Hierarchical Tree (CBHT), a certain group of members of common interest has to form a group. The CC forms a group after getting the appropriate count of members, by clustering, along with partition types. The clustering may be any one of the following types based on the application and the mode of environment whether wired/wireless.

Key based clustering: Based on similarities of public key {x1, y1} or private key {x2, y2} of the members, clustering has been done. The public key is constructed with their private keys and the contribution to the group key formation is given.

Position based clustering: By analyzing the exact position of the members, clustering process is done based on their location.

Time based clustering: In order to form a cluster based hierarchical tree based on time, a database which is used to store the time related entities like the member joining time and leaving time, has to be maintained and it helps in forming cluster.

The Group Member database (DBGM) is used to store the Key, Location and Time based entities, which are controlled by the Cluster Controller Head (CCH). By using these entities, a CBHT can be easily generated. After completion of this process, the key can be generated for both the member and cluster head. This process is controlled by Cluster Key Formation (CKF) and Cluster Controller Formation (CCF). Fig. 1 shows the structure of the Cluster Based Hierarchical Tree (CBHT). The Fig. 2 illustrates the Cluster Initialization, Key Formation and Secure Group Communication. The Architecture is well explained below.

Group and member key formation: The Cluster Controller head (CCH) is responsible for generating the group key. Here, the group key is formed using Elliptic Curve Cryptography. A public key is constructed by each member in the cluster with his own private key and it will be sent to the respective Cluster Controllers. An elliptic curve consists of the points satisfying the equation $y^2=x^3+ax+b$. It also has a distinguished point at infinity which is denoted by ∞. The key computational process is done as follows.

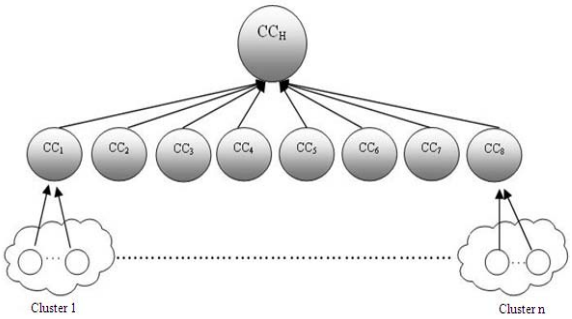

Fig.1: Cluster Based Hierarchical Tree

Elliptic curve key generation: E is considered to be an elliptic curve specified over a finite field Fp. Let p be a point on E (Fp) and is assumed to have a prime order n. The cyclic subgroup E (Fp) that is generated by p is {p} = {p, 2p, 3p……. (n-1)p, ∞}.

The public domain parameters are the prime p and its order n and the equation of the elliptic curve E. A private key d is an integer. It is selected randomly from the interval [1, n-1] and has its corresponding public key is q=dp.

The Key Exchange Protocol (KEP) enables the secured and effective use of keys, considering the members involved in communication. Each member chooses his own random key and multiplies it with global key to form the public key. The result of each member is sent to concerned Cluster Controller where all the public keys are added and multiplied with his own integer private key by the Cluster Controller and the resultant key, the Group key of each Cluster is formed and this will help to do Intra Process Communication.

Then each Cluster Controller will send the Group key of his own Cluster to the Cluster Controller Head to form another key for Inter Process communication. The final group key formed by the Cluster Controllers Head is issued to all the Cluster Controllers and the communication is allowed to take place by encrypting and decrypting the messages secretly among all Cluster Controllers.

Secure group communication: Secure Group Communication (SGC) is the process of transferring the message from one member to another member in a highly secured manner. The SGC performs, joining and leaving operations and maintains the transfer of message between the sender and receiver. The transfer of messages can take place either among nodes under the same cluster called the Intra cluster communication.

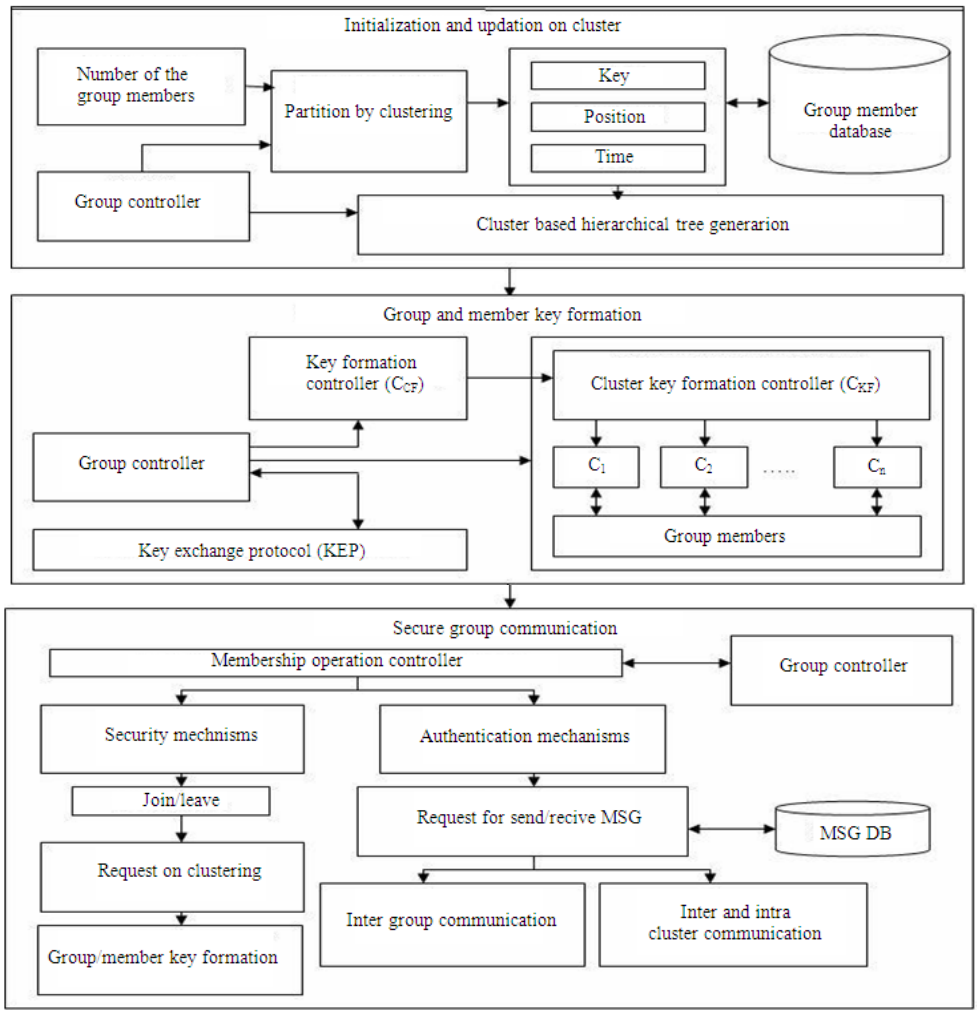

Fig. 2: Protocol architecture

or among different group of clusters called the Inter cluster communication

Due to lack of security aspects in the current scenario of networks, different secrecy policies and authentication mechanisms are to be adopted. They control the join/leave operation in a secured way and check for the user authentication. The DBMSG stores all the group messages and it can be accessed by the proper group member only. The cluster is updated on addition/deletion of a member along with the generation of the authenticated key.

Design and implementation: The design and implementation of Secure Group Communication Protocol is done using Advanced Java. Each phase of the architecture is implemented as a separate algorithm. The output of one algorithm is fed as an input to another algorithm. Each one holds its own security mechanism to secure the message conversation and authentication. Below we will briefly discuss about the Cluster Based Hierarchical Tree generation (CBHT), key formation and message communication through ECC Algorithm.

Cluster based hierarchical tree generation: This phase has an algorithm that generates a cluster based hierarchical tree in an efficient manner. Its main mechanism is to provide tree dynamics for joining / leaving operations. The algorithm for cluster formation is shown below. The CBHT is formed based on the type of applications.

Algorithm 1. Cluster Formation

Input: Cluster Formation Parameters (UserTotal, TypeCluster).

Output: CBHT Generation.

```
1. Digit:=TotalDigit(UserTotal);
2. Digit/=2;
3. assert(UserTotal<=power(2,PowerDigit*Digit));
4. if(true) begin
5.
    CreateDummyNode(GetMemberAddress(dbgm.Poi
nter));
6. end
7. if(TypeCluster==0) begin
8. AssignMember(dbgm.Key);
9. ClusterTree(Sort(dbgm.Member));
10. else if(TypeCluster==1)
11. AssignMember(dbgm.Location);
12. ClusterTree(Sort(dbgm.Member));
13. else
14. AssignMember(dbgm.TimeStamp);
15. ClusterTree(Sort(dbgm.Member));
16. end
17. ClusterUpdation(db.Member,Cluster) begin
18.
    CreateDummyNode(GetMemberAddress(dbgm.Poi
nter));
19. AssignMember(dbgm.Cluster);
ClusterTree(Sort(dbgm.Member));
20. end
```

Key formation: This phase generates the key not only for the group members of the resultant hierarchical tree, but also for the cluster controllers using ECC technique to enable effective key exchange.

Algorithm 2. Key Pair Generation

Input: Key Generation Parameters (a, b, p, dbgm.MemberList).

Output: Pair of Keys [x, y]

```
1. string Eqn="y^2=x^3+ax+b";
2. v1=mod(Value(a, b, x),p);
3. v2=mod(y, p);
4. if(v1==v2) //for any [x, y]points array
5. begin
6. assume n1 as integer //chosen by group member
7. assign(n1,dbgm.Member);
8. arr.x=Computex(global.x,global.y,p,a,b,n1);
9. arr.y=Computey(global.x,global.y,p,a,b,n1);
10. arr.point=Computep(arr.x,arr.y,n1);
11. return arr;
12. end
13. else
14. return null;
```

Elliptic curve encryption/ decryption strategy: A plaintext m is denoted by point M and it is then encrypted by adding it to kQ, where k is an integer selected randomly and Q is the targeted recipient's public key. The sender sends the points C1=kP and C2=M+kQ to the recipient .The recipient uses his/her private key d to compute dC1=d(kP)=k(dP)=kQ and thereafter recovers M= C2 – kQ. An eavesdropper now has to compute kQ. This task of computing kQ from domain parameters is accomplished with the elliptic curve analogue of the Diffie-Hellman problem.

Algorithm 3. Elliptic Curve Encryption

Input: Elliptic Curve Parameters (p, E, P, n). Public key Q, Plain Text m.

Output: Cipher Text (C1,C2)

1. Represent the message m as a point M in E(Fp).
2. Select k€ R [1, n-1].
3. Compute C1=kP.
4. Compute C2=M+kQ.
5. Return(C1,C2)

Algorithm 4. Elliptic Curve Decryption

Input: Elliptic Curve Parameters (p,E,P,n). Private Key d, Cipher Text (C1,C2).

Output: Plain Text m.

1. Compute M= C2-d C1 and extract m from M.
2. Return(m).

Group communication: The generated keys are used to provide security mechanisms for transferring the messages through encryption and decryption methods. The group dynamics enables the updating of the group whenever a joining / leaving operation is performed by the group member. The following pseudo code explains the entire process of SGC.

Algorithm 5. Membership Process Control

```
1. SGC(Operation,dbgm.AccessControl) begin
2. if(Operation.Join==1||Operation.Leave==1)
3. begin
4. DoMembershipProcess(CC);
5. ClusterUpdation(dbgm.Member,Cluster)
6. end
7. elseif(Operation.Send==1||Operation.Receive==1)
8. begin
9. Commn(Crypto(dbmsg.Msg))
10. ClusterCommn(dbmsg.EncMsg);
11. end
12. end
```

RESULTS AND DISCUSSION

The implementation of the CBHKDP is carried out in Windows platform with 64 systems with Advanced Java as front end and Oracle database as backend. A single common group key is formed with the keys obtained from all the 64 terminals which act as servers and each server has 64 terminals as its clients being run simultaneously. The joining/leaving operations of the member are performed on each machine and the key is obtained to have Intra/Inter Cluster communications among all group members.

For instance, an elliptic curve over the finite field F_{23} is assumed. Let a = 1 and b = 0 and the elliptic curve equation be y2 = x3 + x. The points satisfying the equation are: (0,0) (1,5) (1,18) (9,5) (9,18) (11,10) (11,13) (13,5) (13,18) (15,3) (15,20) (16,8) (16,15) (17,10) (17,13) (18,10) (18,13) (19,1) (19,22) (20,4) (20,19) (21,6) (21,17) . The 64 keys from each terminal which are used to generate the group and public keys are obtained with these points that are derived from our implementation. The following Fig. 3 with 8 Cluster Contollers under a Cluster Controller Head illustrates the implementation of our model. Example if there are 8 members under each Cluster Controller(CC), the function of the CC is to calculate the Cluster group key and provide a medium of access for communication between the members of the same Cluster(Intra Communication) and the Cluster Controller Head(CCH) will form a Group Key deriving keys from each Cluster Controllers for Inter Group Communication. Here we have taken the global point as (16, 15).

The CCH in the Fig. 3 is the head of all the Cluster Controllers. Here, (1, 18) is the group key that is computed by the contributions of all the Cluster members.

The main task after the generation of the group key is to establish a secured communication among the group members. For an example, the communication between M1 and M49 is assumed. The member M1 sends the message (say "HELLO") to M49 using ECC. The encrypted form of the message "HELLO" is {(9, 18), (16, 8)}, {(9, 18), (15, 3)}, {(9, 18), (20, 19)}, {(9, 18), (20, 19)}, {(9, 18), (20, 4)} which is obtained as the result of our implementation. Now, the member M49 easily decrypts the message as "HELLO". If three members want to leave the group and two members want to join the group then all the Cluster Controller keys with which the Cluster Controller Head key must be updated.

Performance analysis: The Table 1-5 show the cost effectiveness of proposed model over the other existing models.

Communication cost: The data has been encrypted with the help of the ECC algorithm and then distributed it to the other systems to achieve secure communication over heterogeneous networks. By using the key, the member can encrypt / decrypt the message and also it provides authentication tools for better communication. Table 1 shows Our proposed protocol takes O(1) as cost of communication because only one message is needed to transmit to the Cluster Controller regarding the joining / leaving of the members in the network. Hence our proposed Protocol incurs lesser communication cost than that of the existing protocols.

Computational cost: Whenever group members join/leave the group, the Group Key has to be refreshed to achieve high level of security, forward and backward secrecy. The key updating must be done immediately and sent to all members in the group. The group key is updated due dynamic changes in the group. It is nor prudent to change the group key if the communication takes place during the time of the changes. To solve this problem, the group controller issues a key initially to the newly joined members to take part in the communication temporarily and then it will issue the key later so that all the members can make use of the new group key for further communications. Table 2 and Fig. 5 shows the proposed protocol has only minimum key computational complexity of $\log_a \lceil N/M \rceil$ for the computation of the group key in joining/leaving. An effective ECC Algorithm is utilized for computation along with CBHKDP Mechanism.

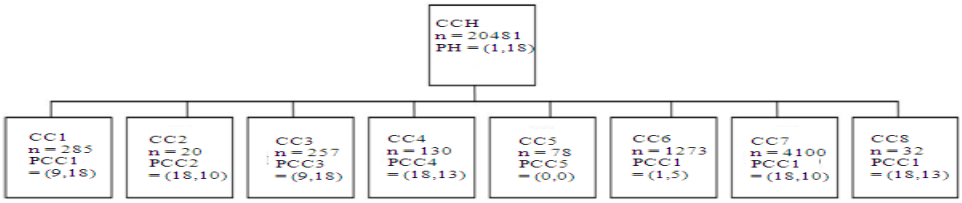

Fig. 3: Example of 8 Cluster Contollers under a Cluster Controller Head

Table 1: Communication Cost

Techniques	Join (Multicast)	Leave (Unicast)	
Simple Application	1	1	n-1
Logical Key Hierarchy (LKH)	$2\log_2 n - 1$	$\log_2 n$	$2\log_2 n$
One-way Function Tree (OFT)	$\log_2 n + 1$	$\log_2 n + 1$	$\log_2 n + 1 + 1$
Key Graph	$\log_2 n$	$\log_2 n + 1$	$\log_2 n + 1$
Logical Key Tree	1	1	1
Proposed Protocol	1	1	1

Table 2: Computation Cost

Protocols	Join	Leave
Simple application	1	n
LKH	$2\log_2 n - 1$	$2\log_2 n$
OFT	$\log_2 n + 1$	$\log_2 n + 1$
Key Graph	$\log_4 n + 1$	$4\log_4 n - 1$
Logical Key Tree	$\log_2 \lceil n/m \rceil$	$\log_2 \lceil n/m \rceil$
Proposed Protocol	$\log_a \lceil N/M \rceil$	$\log_a \lceil N/M \rceil$

Table 3: Number of rekey messages needed

	Number of rekey messages needed	
	Join	Leave
Simple application	1	n
LKH	d+1	2d
OFT	d+1	d+1
OFCT	$\log_d N + 1$	$(d-1)\log_d N$
Our protocol	$\log_a \lceil N/M \rceil + 1$	$\log_a \lceil N/M \rceil - 1$

Table 4: Encryption/decryption overhead

	Key server		Member node	
	Join	Leave	Join	Leave
Simple application	2	n − 1	1	1
LKH	$3\log_2 n$	$2\log_2 n$	$\log_2 n$	$\log_2 n$
OFT	$2\log_2 n + 2$	$2\log_2 n + 1$	$\log_2 n$	2
Key graph	$\log_4 n + 2$	$4\log_4 n - 1$	$\log_4 n$	$\log_4 n$
Logical key tree	$\log_2 \lceil n/m \rceil + 2$	$\log_2 \lceil n/m \rceil$	$\log_2 \lceil n/m \rceil$	0
Proposed protocol	$\log_a \lceil N/M \rceil + 2$	$\log_a \lceil N/M \rceil$	$\log_a \lceil N/M \rceil + 2$	0

Table 5: Key Storage during Join and Leave Operations

Protocols	Key server	Member node
Simple application	n	2
LKH	2n	$\log_2 n + 1$
OFT	2n	$2\log_2 n + 1$
Key graph	$\left\lceil \frac{d}{d-1} \right\rceil n$	$\log_4 n + 1$
Logical key tree	$n \le 2m \quad 2m$	
$n>2m \quad 2(\lceil N/M \rceil + n)$	$\log_2 \lceil N/M \rceil + m - 1$	
Proposed protocol	$S^*_\infty = \frac{\ln \log_a [N/M]}{\ln a}$	$(\log_a [N/M]) + 1$

Optimal mechanism for rekeying: Here an efficient mechanism for rekeying is presented and this mechanism reduces rekeying overhead that is the number of encryptions, decryptions and the size of multicast message during leaving and joining of nodes are considerably reduced compared to other existing schemes.

Our focus is to distribute and manage the group key among large group after the changes in membership.

In our scheme, there is a twisted key server which is responsible for generating required keys and distributing those keys to the valid group members. Here all the cluster controller heads act as key server.

When a new member joins the group, the SEK(Session Encryption Key) must be updated. The SEK is generalized as:

$$GSK(SEK) = K_{P_{GC}} \times \{K_{m1} + K_{m2} \cdots K_{mn}\}$$

where 'n' depends on cluster size M.

The key handled by the cluster controller is called as "Group Session Key"(GSK) and the key generated / handled by the group controller head is called "Domain Key(DK):

$$DK = K_{P_{DC}} \times \{K_{G1} + K_{G2} + K_{G3} + \cdots K_{Gn}\}$$

where 'n' depends on M. In this scheme the height of the tree is considered as $\log_a \lceil N/M \rceil$ .For generating GSK and DK, the server secrets $K_{P_{GC}}$ and member secrets K^{ml} are used to ensure forward and backward secrecy.

Joining of a member: When a member joins the group, it has to obtain GSK and DK to have communication with the group members.

Once a member joins the group, the key server has to update the GSK and DK by transmitting following messages:- One broadcast message to existing members, one unicast message to the group controller head to notice the arrival of a new member and update the DK. Finally one unicast message is sent to the new member. It is indicated as follows.

Consider if M_3 wants to join the group under GC_1:

$$\begin{Bmatrix} \text{if } M=3, N=M^a, a=2 \\ \therefore N=3^2=9 \end{Bmatrix}$$

If M_3 sends a join request to GC_1 {key server}, GC_1 has to send following messages:

$$KS \rightarrow \{M_1, M_1\} : \{K_{1.1'}\} K_{1.1}$$

$$KS \rightarrow M_3 : \{K_{1.1'}\} K_3$$

$$KS \rightarrow GCH : \{K_{0'}\} K_0$$

$$GCK\{KS\} \rightarrow \{GC_2, GC_3\} : \{K_{0'}\}.\{K_{1.2}, K_{1.3}\}$$

The number of rekey messages for joining a new member is given as, $\log_a [N/M] + 1$ and this is the optimal rekey messages compared to other schemes

When the degree of the tree gets increased with increase in number of members, the number of rekey messages that must transmitted by the key server is lesser and optimal.

For example, if the degree of the tree is taken as 4, cluster size equal to 8, the total member of members become 4096. Then the total number of rekey messages for joining is calculated as:

$$= \log_4 \frac{N}{M} + 1 = \log_4 \left[\frac{4096}{8}\right] + 1 = \log_4 [512] + 1 = \log_4 \{4^4 + 4^4\} + 1$$

$$= \{\log_4 4^4 + \log_4 4^4\} + 1 = \{4+4\} + 1 = 9$$

The total number of rekey messages required is lesser compared to other existing schemes.

Leaving of a member: When a member wants to leave the group, the number of messages that must be text, is calculated as follows.

When an existing member wants to leave the group, the keys server has to update GSK and DK, computing as follows:

$$GSK' = K_{P_{GC}} \times \{K_{m_1} + K_{m_n} + \cdots K_{m_n}\} - \{K_{P_{GC}} \times K_{m_k}\}$$

K is the number of member:

$$DK' = K_{P_{DC}} \times \{GSK' + K_{GC_1} + \cdots K_{GC_n}\}$$

When a member wants to leave the group, the keyserver has to transmit following messages. One broadcast message to the other members who belong to the same group and the unicast message to the group controller head to update DK and GSK. Hence the total number of rekey messages required for the member to leave is given as $\log_a [N/M] - 1$.

For example, for M_1 wants to leave the group CC_1, the keyserver has to transmit the following messages:

$$KS \rightarrow \{M_2, M_3\} : \{K_{1.1}'\}.K_{1.1}$$

$$KS \rightarrow CCH : \{K_0'\}.K_0$$

If a = 4, M = 8, N = 4096 the number of rekey messages required is:

$$= \log_4 \left[\frac{4096}{8}\right] - 1 = \{\log_4 4^4 + \log_4 4^4\} - 1 = \{4+4\} - 1 = 7$$

Hence when 'N' grows larger, the number of rekey messages required is lesser and this scheme produces optimal overhead.

Number of rekey messages needed: To retain the forward and backward secrecy during the joining / leaving operation, the concept of rekeying is used. The proposed Protocol consumes less number of rekey messages than that of the existing protocols. The cost of rekey message is computed based on the dummy nodes through which the message has been passed. Table 3 shows for any cluster size, it will take only one rekey message for our proposed protocol. The subgroup size may be 8,16,32,64 and so on.

Cost of encryption/decryption: The cost of encryption while joining with the key server of proposed scheme is $\log_a [N/M] + 2$ because the key server has to send two encrypted messages to existing group members and the new member who sends join request. The overhead of encryption while leaving the key server is computed as $\log_a [N/M]$.

The decryption overhead of proposed scheme is $\log_a [N/M]$ at key server and it doesn't require decryption when a member node leaves the group. Hence the decryption overhead at member node becomes 0. Thus the proposed scheme produces optimal encryption/decryption overhead compared to all other existing schemes.

The Table 4 displays the analytical result of message encryption/decryption by proposed technique along with other existing models.

Key storage at join and leave operations: The following derivation is used to calculate the probability of key storage when any member leaves/joins the group.

N-ray tree: To reduce the storage at GC, the group of 'N' members is divided into clusters of size 'M'. To obtain an optimal tree, as in Fig. 4 a tree is constructed with the height of $\log_a [N/M]$ and $N=M^a$, where 'a' is the degree of the tree, M-cluster size, N – total number of member nodes.

$M = 4$ $a = 2$ $N = M^a = (4)^2$ $N = 16$

As per our proposed concept, the user needs to store $\{1+\log_a (N/M)\}$ keys.

'1' represents DK & $\log_a [N/M]$ represents GSK.

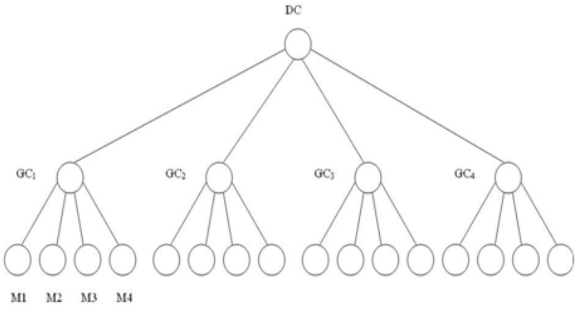

Fig. 4: Cluster Controllers with 4 members and a CCH

This is the optimal storage of key which members posses in the clusters.

For example, a member at cluster 1 has to store his secret key and GSK.

When M1 leaves the cluster, the GSK alone must be updated by CC1.

CC1 has to update the GSK by calculating $GSK = \left[K_{R_{GC_1}} \times \left\{ K_{p_{m_1}} + \cdots K_{p_{m4}} \right\} - \left\{ K_{R_{GC_1}} \times K_{p_{m1}} \right\} \right]$ therefore, the total number of key update messages per member leaving is denoted as:

$$C = (M-1) + \log_a (N/M) \quad (1)$$

In the minimal storage scheme, GC uses a secret key for generating SEK for each user.

Therefore the number of keys stored by the cluster controller is:

$$S = \lceil N/M \rceil + 1 \quad (2)$$

1 represents storage required for GSK and $\lceil N/M \rceil$ represents storage for the group member's public key (or) KEK.

Minimizing storage at key server/GC: To minimize the center storage it is necessary to take an optimal cluster size 'M'. Based on (1-2) the following expressions are formed as:

$$S = \lceil N/M \rceil + 1 \quad \text{w.r.t. } M \quad (3)$$

$$C = (M-1) + \log_a (N/M) \le \beta(N) \quad (4)$$

Where $\beta(N)$ is the number of key messages per update and it is an application dependent design parameter.

The Eq. (3-4) are used to derive optimal cluster size for the construction of n-array tree.

Theorem 1: For the optimal cluster size M that minimizes the storage function $S = \lceil N/M \rceil + 1$, by satisfying the update communication budget $C = (M-1) + \log_a \lceil N/M \rceil \ge \beta(N)$ is obtained by the largest root of the Equation $M - \lambda \ln M = \mu$ and $\lambda = \frac{1}{\ln a}$, $\mu = 1 + \beta(N) - \log_a \lceil N/M \rceil$

Proof: Since the storage is the decreasing function of M, the lowest value of M which satisfies the communication constraint will be the solution.

Hence, optimal value of the cluster size is computed as:

$$M^* - \lambda \ln M^* = \mu$$

$$M^* - \lambda \ln M^* - \mu = 0$$

$$M^* - \lambda \ln M^* - \left\{ 1 + \beta(N) - \log_a \lceil N/M \rceil \right\} = 0$$

$$M^* - \lambda \ln M^* - \left\{ 1 - \beta(N) + \log_a \lceil N/M \rceil \right\} = 0$$

$$M^* - \lambda \ln M^* + \log_a \lceil N/M \rceil - 1 = \beta(N) \quad (5)$$

The update communication constraint (4-5) are convex function of M and attain its minimum value at:

$$M^* - \lambda \ln M^* + \log_a \lceil N/M \rceil - 1 = (M-1) + \log_a \lceil N/M \rceil$$

$$M^* - \lambda \ln M^* = M$$

$$\lambda - \lambda \ln \lambda - \lambda = 0$$

$$\lambda [\ln \lambda] = 0$$

$$\lambda [\ln \lambda] \text{ at } M = \lambda [\ln \lambda]$$

Hence, the factor $\beta(N)$ must satisfy the following inequality function to solve (5):

$$\begin{aligned} \beta(N) &\ge \lambda \{\ln \lambda\} \\ &\ge 1.44 * \{\ln 1.44\} \\ &\ge 1.44 * 0.364 \\ &\ge 0.524 \end{aligned} \quad (6)$$

For larger value of 'N', the asymptotic lower bound of $\beta(N)$ approaches $\log_a N$ and the equation (5) can be rewritten as:

$$M^* - \lambda \ln M^* = \mu \quad (7)$$

Solution to storage optimization: The fixed point Eq. (7) is the contradiction mapping in the range of interest $[\lambda, \infty]$.

Set the initial value of $M_0 = \mu$.

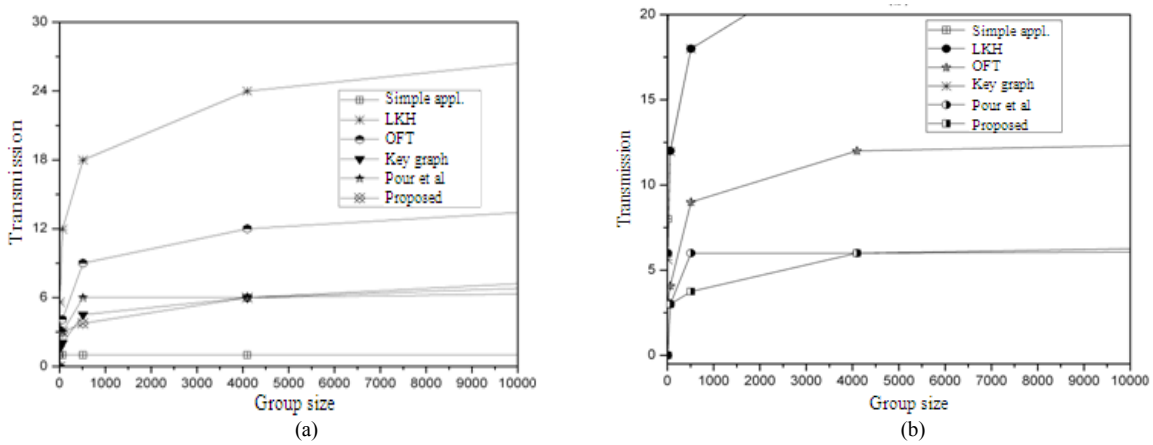

Fig. 5: shows computational cost on member joining (a) and member leaving (b)

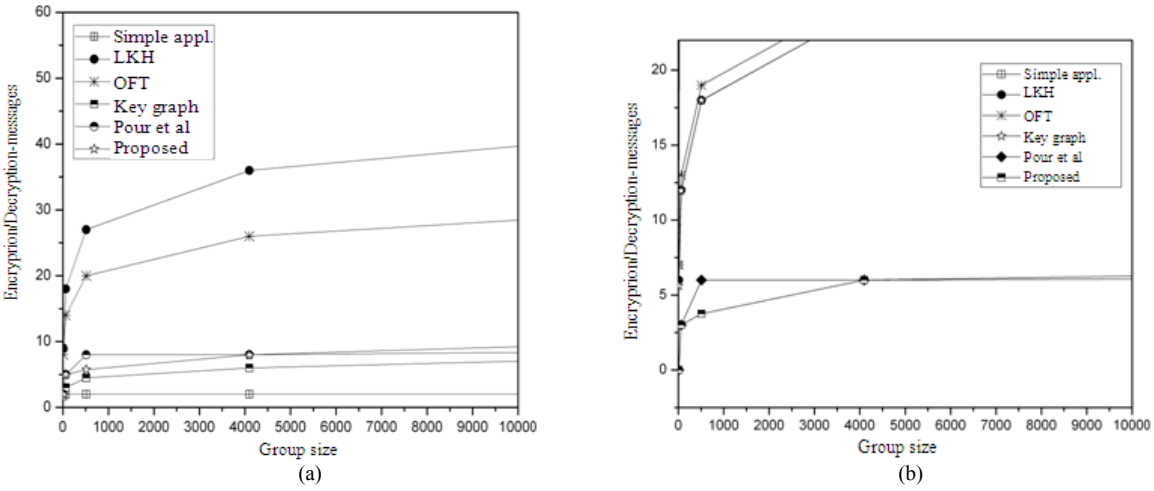

Fig. 6: Key Storage during Joining (a) and Leaving (b) Operations of Proposed protocol (Key Server)

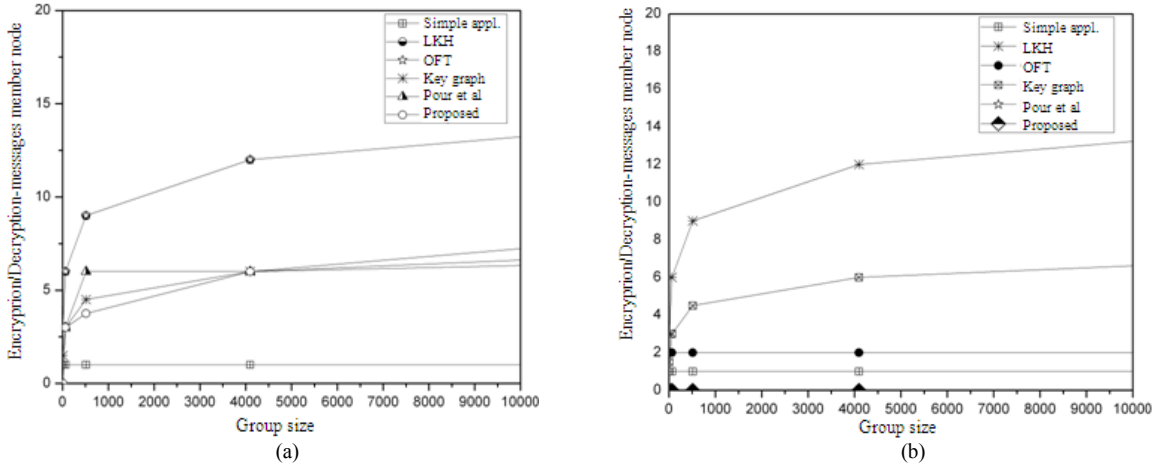

Fig. 7: Key Storage during Joining (a) and Leaving (b) operations of Proposed protocol (Member Node)

Theorem 2: When N grows larger (∞) and M grows larger (∞) at CC requires $\frac{\ln \log_a (N/M)}{\ln a}$ with the constraint that $a \geq 2$.

Proof: Assume that the initial value of M is $M_0 = \mu$, then (7) can be written as:

$$M^* = \mu - \lambda \ln M^*$$

$$= \mu - \lambda \ln \mu^*$$

$$= \mu \left[1 - \frac{\lambda}{\mu} \ln \mu^* \right]$$

After applying some approximation, M is given as:

$$M^* = \mu \prod_{i=1}^{\infty} \left\{ 1 - \left(\frac{\lambda}{\mu} \right)^i \ln \mu \right\}$$

The asymptotic value of M when $N \to \infty$, is given as:

$$M_\infty^* = \lim_{N \to \infty} \mu \prod_{i=1}^{\infty} \left\{ 1 - \left(\frac{\lambda}{\mu} \right)^i \ln \mu \right\}$$

$$= \lim_{N \to \infty} \left\{ \mu \prod_{i=1}^{\infty} 1 - \left(\frac{\lambda}{\mu} \right)^i \ln \mu \right\}$$

$$= \lim_{N \to \infty} \left\{ \mu \prod_{i=1}^{\infty} 1 - \mu \prod_{i=1}^{\infty} \left(\frac{\lambda}{\mu} \right)^i \ln \mu \right\}$$

$$= \lim_{N \to \infty} \left\{ \mu - \mu \prod_{i=1}^{\infty} \left(\frac{\lambda}{\mu} \right)^i \ln \mu \right\} \quad (8)$$

After the series of approximation, (8) becomes:

$$= \lim_{N \to \infty} \mu + \lambda \left\{ \frac{\ln \mu}{\mu^{\lambda - 1}} \right\}$$

$$= \mu + \lambda \cdot 0 \left\{ \frac{\ln \mu}{\mu} \right\}$$

$$= \mu + \lambda \cdot \ln \mu \quad (9)$$

Where $M = \mu$ and $N \to \infty$

After applying the values of μ and λ, (9) becomes:

$$= \left\{ 1 + \beta(N) - \log_a (N/M) \right\} + \frac{1}{\ln a} \cdot \ln \mu \quad (10)$$

$$= \left\{ 1 + \beta(N) - \log_a (N/M) \right\} + \frac{1}{\ln a} \cdot \ln \mu$$

$$= 1 + \beta(N) - \log_a (N/M) + \frac{1}{\ln a} \cdot \ln \mu$$

$$= 1 + \beta(N) - \log_a (N/M) + \frac{1}{\ln a} \left\{ \ln \left\{ 1 + \beta(N) - \log_a (N/M) \right\} \right\}$$

$$= \frac{\ln \log_a [N/M]}{\ln a}$$

By solving (10) the result of storage constraint becomes as follows:

$$S_\infty^* = \frac{\ln \log_a [N/M]}{\ln a} \text{ Where } a \geq 2 \text{ and } N \to \infty$$

S^* is the generalized notation for storage cost.

Hence the constraint optimization leads to the optimal growth of storage at member node as $\left(\log_a [N/M] \right) + 1$. The Table 5, Fig. 6 and 7 shows how our proposed protocol is far better than all other schemes in Key Storage during Joining/Leaving operations for Key Server and Member Node.

CONCLUSION

Our proposed protocol comparatively produces better results than the existing protocols in terms of less key computational cost and communication cost. The number of keys stored in the key server/member and the number of rekey messages needed by the introduction of the clustering technique are comparatively less. The cluster sizes of 8, 16, 32 and 64 have been empirically tested. The proposed architecture is efficient in the view of cost effective secure group communication in the context of distributed environment by applying the ECC. Our proposed model does not need any trusted key center for the distribution of the keys. The Cluster Controllers and the Cluster Controller Heads look after the root key formation for intra/inter communication between the members. Our proposed model can be extensively applicable to large groups, either wired or wireless with a low bandwidth channels or wide area network environment. As future scope of work, further reduction in computational cost and the time needed for rekeying while members joining/leaving can be focused.

REFERENCES

Al-Saadoon, G.M.W., 2009. A flexible and reliable architecture for mobile agent security. J. Comput. Sci., 5: 270-274. DOI: 10.3844/jcssp.2009.270.274

Al-Talib, S.A., B.M. Ali and S. Khatun, 2009. An approach to improve the state scalability of source specific multicast. Am. J. Applied Sci., 6: 1347-1351. DOI: 10.3844/ajassp.2009.1347.1351

Amir, Y., Y. Kim, C. Nita-Rotaru, J. Schultz and J.S.G. Tsudik, 2004. Secure group communication using robust contributory key agreement. IEEE Trans. Parallel Distributed Syst., 5: 468-480. DOI: 10.1109/TPDS.2004.1278104

Kim, H., S.M. Hong, H. Yoon and J.W. Cho, 2005. Secure group communication with multiplicative one-way functions. Proceedings of the International Conference on Information Technology: Coding and Computing, (ITCC'05), IEEE Computer Society Washington, DC, USA., pp: 685-690. DOI: 10.1109/ITCC.2005.252

Kulkarni, S.S. and B. Bruhadeshwar, 2010. Key-update distribution in secure group communication. Comput. Commun., 33: 689-705. DOI:10.1016/j.comcom.2009.11.014

Lee, F.Y. and S. Shieh, 2004. Scalable and lightweight key distribution for secure group communications. Int. J. Network Manage., 3: 167-176. DOI: 10.1002/nem.515

Manz, D., P. Oman and J.A. Foss, 2010. A framework for group key management protocol assessment independent of view synchrony. J. Comput. Sci., 6: 229-234. DOI: 10.3844/jcssp.2010.229.234

Mao, Y., Y. Sun, M. Wu and K.J.R. Liu, 2004. JET: Dynamic join-exit-tree amortization and scheduling for contributory key management. IEEE Trans. Network., 14: 1128-1140. DOI: 10.1109/TNET.2006.882851

Micciancio, D. and S. Panjwani, 2008. Optimal communication complexity of generic multicast key distribution. IEEE/ACM Trans. Network., 16: 803-813. DOI: 10.1109/TNET.2007.905593

Poovendran, M.L.R. and D.A. McGrew, 2004. Minimizing center key storage in hybrid one-way function based group key management with communication constraints. Inform. Proc. Lett., 93: 191-198. DOI: 10.1016/j.ipl.2004.10.012

Pour, A.N., K. Kumekawa, T. Kato and S. Itoh, 2007. A hierarchical group key management scheme for secure multicast increasing efficiency of key distribution in leave operation. Comput. Networks, 51: 4727-4743. DOI: 10.1016/j.comnet.2007.07.007

Prathap, P.M.J. and V. Vasudevan, 2009. Analysis of the various key management algorithms and new proposal in the secure multicast communications. IJCSIS, 2: 8.

Rafaeli, S. and D. Hutchison, 2003. A survey of key management for secure group communication. ACM Comput. Survey, 35: 309-329. DOI: 10.1145/937503.937506

Saroit, I.A. S.F. El-Zoghdy and M. Matar, 2009. A scalable and distributed security protocol for multicast communications. Int. J. Network Secur., 12: 61-74.

Sudha, S., A. Samsudin and M.A. Alia, 2009. Group re-keying protocol based on modular polynomial arithmetic over galois field GF(2^n). Am. J. Applied Sci., 6: 1714-1717. DOI: 10.3844/ajassp.2009.1714.1717

Wong, C.K., M. Gouda and S.S. Lam, 2000. Secure group communications using key graphs. IEEE/ACM Trans. Network., 8: 16-30. DOI: 10.1109/90.836475

Yi, X., 2005. Security of Chien's Efficient Time-bound hierarchical key assignment scheme. IEEE Trans. Knowledge Data Eng., 17: 1298-1299. DOI: 10.1109/TKDE.2005.152

Zheng, S., D. Manz and J. Alves-Foss, 2006. A communication-computation efficient group key algorithm for large and dynamic groups. Comput. Networks, 51: 69-93. DOI: 10.1016/j.comnet.2006.03.008