An Efficient Distributed Group Key Management Using Hierarchical Approach with Elliptic Curve Cryptography

Shikha Sharma
PG Scholar
Computer Science Department
NITTTR, Chandigarh, India
shikha.cs10@gmail.com

C. Rama Krishna
Professor
Computer Science Department
NITTTR, Chandigarh, India
rkc_98@hotmail.com

Abstract: **Secure and reliable group communication is an active area of research. Its popularity is fuelled by the growing importance of group-oriented and collaborative properties. The central research challenge is secure and efficient group key management. In this paper, we propose an efficient many-to-many group key management protocol in distributed group communication. This protocol is based on Elliptic Curve Cryptography and decrease the key length while providing securities at the same level as that of other cryptosystems provides. The main issue in secure group communication is group dynamics and key management. A scalable secure group communication model ensures that whenever there is a membership change, a new group key is computed and distributed to the group members with minimal communication and computation cost. This paper explores the use of batching of group membership changes to reduce the time and key re-distribution operations. The features of ECC protocol are that, no keys are exchanged between existing members at join, and only one key, the group key, is delivered to remaining members at leave. In the security analysis, our proposed algorithm takes less time when users join or leave the group in comparison to existing one. In ECC, there is only *1* key generation and key encryption overhead at join and leave operation. At join the communication overhead is key size of a node and at leave operation is $2\log_2 n - 2 \times key\ size$ of a node.**

Keywords: Group Communication; Distributed Group Key Management; Hierarchical Group Key Management

I. INTRODUCTION

With the rapid growth of technology, the secure multicasting is an important technology in the group communication. Previously, for sending the data, we usually used broadcast or peer-to-peer nodes. Nowadays, as rapid advances in the group communication, the unicast and broadcast are not efficient. Although, the benefit of using secure multicasting is for delivering data safely from one sender to multiple receivers. However, security and scalability are two important factors that need to be considered. Security requirements are of two types:

Forward Secrecy: This provides the future confidentiality means provides the security from the users who left the group.
Backward Secrecy: This provides past confidentiality means provides the security from the new users who join the group [1].
Group key management protocols are classified into three main categories:

- Centralized (one-to-many) protocols: A key server is mainly responsible for generating and distributing the group key to *n* group members. So, we have bottleneck problem.
- Decentralized protocols: The group is divided into multiple subgroups. Then each subgroup is managed by a subgroup manager who is responsible for generating and distributing the keys for that subgroup. This protocol helps in the bottleneck problem occurs in centralized protocols. But, in this we have single point of failure at subgroup level.
- Distributed (many-to-many) protocols: In this there is no centralized server. Group members communicate with each other for generating the group key. Each member is responsible for generating the group key.

One of the above protocols is used for specific application. There is no base station or no infrastructure of distributed applications such as MANETs and wireless sensors. For the confidentiality of the group, group members are responsible for exchanging the group key securely when members change. This results in high overhead. The main goal of distributed group key management is how to exchange group key securely and efficiently in the group.

Several approaches have been proposed to reduce the size of key in secure group communication in distributed protocol. Most of these approaches are based on different types of n-parity Diffie-Hellman key agreement protocol. The main issue in such approaches is that the asymmetric key size is larger. Since network overhead is increased.

We propose a distributed group key management using hierarchical approach with elliptic curve cryptography for self organized simple computational group key without central authority. In this, users themselves distribute keys effect on size of key, less rekeying computation and communication cost over existing schemes. In this protocol, group members are arranged in the hierarchical manner logically. Two types of keys are used, symmetric and asymmetric keys. All the intermediate node keys are symmetric keys assigned to each intermediate node. The leaf nodes in the key tree are the asymmetric keys of the corresponding group members. For asymmetric key, we incorporate elliptic curve cryptosystem. To calculate intermediate node keys, members use codes assigned to each intermediate node keys rather than distributed by a

978-1-4799-6023-1/15 $31.00 © 2015 IEEE
DOI 10.1109/CICT.2015.116

sponsor. The key feature of this approach is that by using ECC we decrease the key length while providing the same level of security as other cryptosystems.

This paper is organized as follows. Section II discusses related works. Section III and IV present our proposal. Results are discussed in section V. Finally, the conclusion is given in section VI.

II. RELATED WORK

In hierarchical approaches, the members of group are mapped with the leaves of a logical binary key tree. Each member maintains all the keys along the path from his/her leaf to the root, hereinafter called the path set. The root key is the group key. At join/leave, all the keys in the path set need to be changed to new ones. Based on the key management approach, the number of key generations, key encryptions and key delivery differs. Typically, each member maintains *O(log n)* keys which shows the height of the key tree where *n* is the number of members.

Large amounts of work has been done to establish a group key for secure distributed group communication, many approaches have been proposed [2-11]. As stated before, most of these approaches [2-4] are based on different types of n-party Diffie-Hellman key agreement protocol. The main purpose of these approaches is to reduce the overhead of group key management. The fact with all of them is that the evaluation measures of these approaches are not distinct. For example, they do not consider key generation, key encryption separately in their works. In this section we discuss five types of typical approaches [2-6] in terms of efficiency in communication, computation and scalability.

Distributed/Contributory group key agreement is basically different variations of *n*-party Diffie-Hellman key agreement/exchange. The main drawback using this key exchange mechanism is that the group members need synchronization to iteratively form parental keys from their two children's keys. Once one member is slow or one rekeying, the packet will be delayed and the key agreement process will be postponed or even mis-operates. Furthermore, there are dependencies among node keys (i.e., a blinded node key is dependent on the secret node key and a parental key on its two child's keys). In result, if one key is compromised then dependencies will break all ancestral keys. Due to this, we suffer from man-in-the-middle attack problem, so we can apply authentication capability to each group member, which is also implemented by public key cryptosystems [2].

Tree based Group Diffie-Hellman (TGDH) [3] approach extends the usage of two-party Diffie-Hellman key agreement protocol to a group. TGDH also introduces the concept of hierarchy key tree to manage group members. Each member assigned to the leaf of the key tree, which maintains the key tree. Starting with the leaf nodes, each intermediate node represents a key shared by its two child node keys that are computed with single Diffie-Hellman key agreement protocol. This approach has reduced the modular exponentiation from *O(n)* to *O(logn)* during initial establishment and after each membership change, join/leave. However, the cost of modular exponentiation leads the protocol to delay because the protocol needs initiation after each membership change.

Distributed Group Key Distribution (DGKD) [4] uses the concept of sponsor and co-distributors. A sponsor initiates the key generation and rekeying process at join/leave. The sponsor is chosen based on the ID size. The selected sponsor is responsible to change the keys along the path as well as distribute them to co-distributors and to the new member. A co-distributor is the sponsor of a branch on other path. The drawback of this approach is that all affected intermediate keys in the path have to be generated by the sponsor node. Moreover, this approach uses asymmetric cryptosystem for sending the necessary keys from sponsor to co-distributors which is slower than symmetric cryptosystem.

Efficient Distributed Group Key Agreement Scheme (EDKAS) [5] is very similar to One Way Function Trees (OFT) [12] in the sense of key structure. For each node, a secret key and its corresponding blinded key is associated. The blinded keys are computed by applying a given one-way function. Each member generates a unique secret key for itself by a secure pseudo random number generator (PRNG). The key of an intermediate node is computed by the blinded keys of two child nodes using $K_{i,j} = f(g(K_i), g(K_j))$ where f is a mixing function and g is a given one-way hash function. Each member maintains his/her own secret key and all the blinded keys of the nodes that are sibling of the nodes from the path set. The drawbacks of this protocol are expensive maintenance of secure channels between members and expensive communication cost as well as its message size cost.

Diffie-Hellman and Symmetric Algorithm (DHSA) [6] uses hierarchical key tree to manage the keys logically. In this protocol, the combination of Diffie-Hellman key agreement and symmetric key is used. Diffie-Hellman key agreement is introduced to the leaf nodes of the key tree where the members are assigned and symmetric key is introduced to intermediate nodes. The drawbacks of this protocol are that key size is very large and modulo operation takes long time in computation and it makes the computation slow.

Our proposal, an efficient distributed group key management using hierarchical approach with Elliptic Curve Cryptography is used for key exchange between leaf nodes. Elliptic curve cryptography [13] provides greater security with small key size and scalar multiplication and performing scalar multiplication takes less time in comparison with the modulus and exponent operation performed in previous existing DHSA method. So use of ECC protocol will provide more suitable and efficient technique for key management. The main advantage of ECC is that it has to be computationally fast [16-17] in order to reduce the power consumption of key management process to insure maximum battery life of devices and using ECC for key agreement work faster than using Diffie Hellman Key Agreement. In ECC, intermediate node keys are calculated by group members rather than distributed by a sponsor member.

III. DESIGN PRINCIPLES

Now, we present our proposed approach, ECC, for distributed secure group communication. This approach is to reduce re-keying overhead at join/leave.

ECC mainly focuses on member collaboration for key calculation instead of key delivery by sponsor or co-distributor. For this purpose, we introduce three basic features of ECC:

(1) The leaf key in the key tree is the public key of the corresponding group member, and all intermediate node keys are symmetric keys.
(2) The public key of each member along with binary code of the corresponding parent node is stored in a list shared by group members. This list will be updated on each membership change and periodically.
(3) All group members have the same capability and are equally trusted and responsible.

The public key of each member is generated by Elliptic Curve Cryptography for key agreement. Elliptic curve with parameters *a, b* and *q*, where *q* is a prime number or an integer of the form 2^m and G point on elliptic curve whose order is larger than *n*, the public key is obtained by $n \times G$. This public key is used to share a common key with other members in the group. For example, u_i can share a key with u_j by calculating $n_i \times n_j \times G$.

ECC introduces two types of codes in its key tree:

(1) Binary Code: This code will be used for member position discovery.

(2) Decimal Code: This Code will be used for intermediate node key calculation.

Fig. 1 illustrates a key tree with *8* members,$\{u_1, \dots \dots, u_8\}$, and its corresponding binary code. The binary code of first level of each intermediate node from the bottom of the key tree, and the corresponding two members' public key are stored in a list. Each member uses this list to find the public key of any member whom he/she wishes to establish a connection. As stated before, this list is updated whenever there is a membership change and is broadcasted to other members by multicast. Usually, the sibling member of affected branch is responsible to send the updated information to other members.

Table I shows the management of binary code and its associated members public key in the list. As shown in this table, the public keys of u_1 and u_2 are $n_1 \times G$, $n_2 \times G$ respectively, and their associated parent binary code is *000*. Since there is no sibling member for u_3, the list just shows its public key, $n_3 \times G$, and the associated parent binary code, *00*.

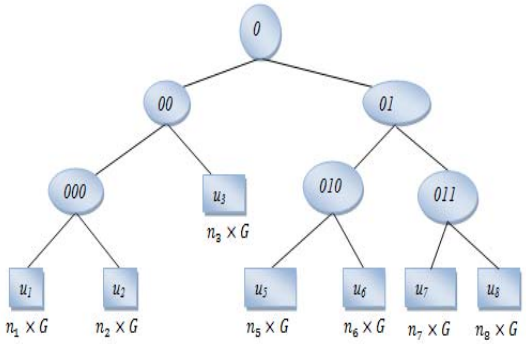

Fig.1 Parent Binary Code for Member Position Discovery

As stated before, the other code type in ECC is decimal code. This code is used just for intermediate node keys calculation, and is assigned to each intermediate node in the key tree. Each intermediate node key is updated by applying one-way hash function to the bitwise XOR of that intermediate node code and the group key by the formula given below:

$$Key_{intermediate\_node} = f(Key_{group} \oplus Code_{intermediate\_node})$$

Moreover, each intermediated node code is calculated by the formula formula below:

$$Code_{child\_node} = (Code_{parent\_node} \parallel Random\ digit).$$

Fig. 2 illustrates the node code management in the key tree with 8 members, $\{u_1, \dots \dots, u_8\}$. For example, when an intermediate node code is *04* and the generated random number is *6*, the code assigned to that new node will be *046*.

Finally, the number of digits in a code shows the number of nodes in the path set. In Table I the intermediate node key computation for members $\{u_1, \dots \dots, u_8\}$ is illustrated. For example, $K_{1,4}$ is calculated as $f(K_G \oplus 04)$.

In ECC, the group key at join is sent to new member being encrypted by the shared key with his/her sibling member. However, the current members can calculate it by applying one-way hash function to previous one. When f is a given one way hash function, and K_G is the previous group key, the new group key K_G' is calculated as follows.

$$K_G' = f(K_G)$$

$K_{1,4} = f(K_G \oplus 04)$ $K_{5,8} = f(K_G \oplus 08)$
$K_{1,2} = f(K_G \oplus 049)$ $K_{5,6} = f(K_G \oplus 081)$
$K_{3,4} = f(K_G \oplus 046)$ $K_{7,8} = f(K_G \oplus 087)$

Table I. List of Binary Code and Associated Members Public Key

Parent Binary Code	Member Public key
000	$n_1 \times G$, $n_2 \times G$
00	$n_3 \times G$
010	$n_5 \times G$, $n_6 \times G$
011	$n_7 \times G$, $n_8 \times G$

IV. DETAILED DESIGN

To explain the detailed approach, consider our simple example with 8 members illustrated in Fig.1 and Fig. 2 for join operation, and Fig. 3 for leave operation. Members decide a large prime number p and its primitive element g for each group. Initially, this value is selected at initial mode of key tree establishment. These values are publicly known in the group.

When a new member wants to join a group, he/she sends a hello message to discover the group members. Members, who receive the signal of this member, look up the list to know which member does not have a sibling member. A member who does not have a sibling member in his/her branch replies to this signal. But when each member has his/her corresponding sibling member in his/her branch, the member with lowest parent binary ID replies to that member. He/she exchanges the public key generated by Elliptic Curve key agreement. Here, a member who replies is responsible to authenticate a new member. We assume that each group member is equipped with some authentication capability [6].

Once authentication operation is completed, the public key of new member and his/her corresponding parent binary code are stored in the list, and the updated information is multicasted to existing members. Next, the current members as well as the new one can calculate the affected intermediate node keys by applying a given one-way hash function to bitwise XOR of new group key and the intermediate node code.

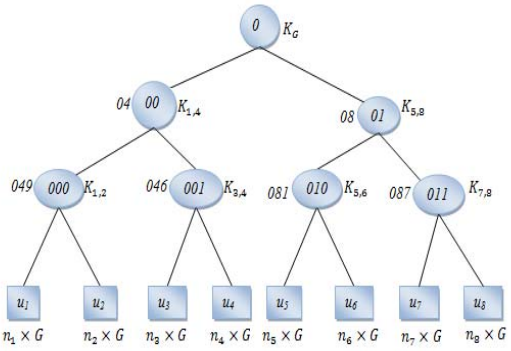

Fig.2 Intermediate Node Code and Corresponding Node Key Calculation

(1) Join Operation of a Node

Fig. 1 illustrates a multicast group of *7* members, $\{u_1, u_2, u_3, u_5, u_6, u_7, u_8\}$ as current members when a new member u_4 joins the group (Fig. 2). Re-keying procedure at join for this example in ECC is as below.

(1) u_4 broadcasts a hello message for member discovery.

(2) u_3 who does not have a sibling node, replies this member.

(3) u_3 shares a key with u_4 by ECC key agreement. This key is $n_3 \times n_4 \times G$.

(4) u_3 downgrades his/her position from *00* to *001*, updates the member discovery key by replacing the new parent binary code and new member's public key (Table II).

(5) u_3 calculates the new intermediate node code for his parent.
Code_$K_{3,4}$ = (04 || 6)= 046.

(6) u_3 generates new group key as below:

$$K'_G = f(K_G)$$

(7) u_3 sends K'_G and the new node code to u_4 being encrypted by the shared key between them.

$$u_3 \xrightarrow{unicast} (K'_G, 046) n_3 \times n_4 \times G$$

(8) Existing members, $\{u_1, u_2, u_3, u_5, u_6, u_7, u_8\}$, renew the group key as described in the step(6)

(9) Then, the members in the affected path set calculate the affected intermediate node keys by applying one-way hash function to bitwise XOR of intermediate node codes and the new group key.

$$u_3, u_4 : K_{3,4} = f(K'_G \oplus 046)$$
$$u_1, \ldots\ldots, u_4 : K_{1,4} = f(K'_G \oplus 04)$$

Table II. List of Parent Binary Code and Associated Members Public Key

Parent Binary Code	Member Public key
000	$n_1 \times G$, $n_2 \times G$
001	$n_3 \times G$, $n_4 \times G$
010	$n_5 \times G$, $n_6 \times G$
011	$n_7 \times G$, $n_8 \times G$

As you notice that just one key is delivered to new member. This is an important feature for distributed group communication in wireless network. Since members are mobile, in addition to dynamic join/leave, simultaneous join may occur in such networks. In order to solve such problem, the overload of join operation must be minimized. The features of ECC provide this task with just one key delivery.

(2) Leave Operation of a Node

When a member leaves a multicast group, his/her node is deleted from the key tree. The sibling member on that branch moves to his/her parent node position. And the sibling node is also responsible to delete the leaving node public key from the list, and to transmit updated information of the list to other members. After each leave, the group key and some intermediate node keys need to be updated. At leave operation, the key tree has divided into some parts. The number of these parts is equal to (*log n - 1*) where *n* is the number of group members. The sibling

of leaving member generates the new group key and sends it to one of the member in each part. To do this the sibling node checks his/her list and finds one of the available members in each part, shares a key with that member using his/her public key and send the group key for his/her via unicast. The member who receives the group key is responsible to multicast it to his/her branch members being encrypted with upper intermediate node which is not affected. Now the users are able to renew the affected intermediate node key. We use a simple example to explain leave operation. Fig.3 illustrates a multicast group of 8 members, $\{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$ when u_8 leaves the group.

(1) u_7 is promoted to his/her parent position.

(2) u_7 updates the member discovery list by deleting the leaving node's public key, and changes his/her parent binary code. u_7 also informs the other nodes about the updated information.

(3) u_7 generates new group key K_G'' , by using symmetric algorithm.

(4) u_7 checks his/her list and use Elliptic Curve key agreement to share a key with one of the member in each branch. Then it will unicast new group key to each of them as follow:

$$u_7 \xrightarrow{unicast} u_1 : (K_G'')n_1 \times n_2 \times G$$

$$u_7 \xrightarrow{unicast} u_5 : (K_G'')n_5 \times n_7 \times G$$

(5) Now u_1 and u_5 multicast the received new group key K_G'', to members of their branch as follow:

$$u_1 \xrightarrow{multicast} u_2, \dots\dots, u_4 : (K_G')K_{1,4}$$

$$u_5 \xrightarrow{multicast} u_6 : (K_G')K_{5,6}$$

(6) Finally the members in affected path calculate the code of the affected intermediate node by the formula given below:

$$u_5, u_6, u_7 : K_{5,7} = f(K_G' \oplus 08)$$

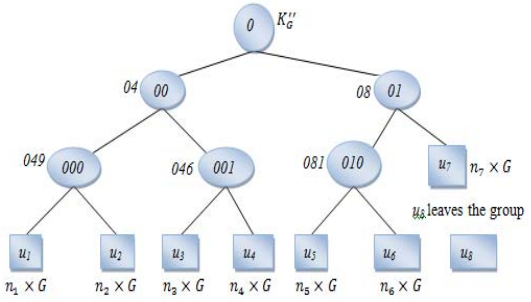

Fig. 3 Leave Operation on ECC

Table III shows the updating of member discovery list after a member leaves a group. This is necessary to insure the backward secrecy. It can be seen from the table given below that after deletion of node u_8 its corresponding public key $n_8 \times G$ was deleted.

Table III. Updating Member Discovery List when a Member leaves the Group

Parent Binary Code	Member Public key
000	$n_1 \times G$, $n_2 \times G$
001	$n_3 \times G$,$n_4 \times G$
010	$n_5 \times G$,$n_6 \times G$
011	$n_7 \times G$

V. RESULTS

In this section we compare and analyze our proposal with DHSA. The comparison metrics which are used include key generation, key encryption and communication overhead.

Key generation overhead is the number of keys generated during the join and leave operations by the sponsor. Key encryption overhead identifies the number of encryptions on any membership change by the sponsor and co-distributors. The last metric, key communication overhead is used to estimate the number of messages required to transmit in group rekeying process from the sponsor and co-distributors.

Proposed ECC and DHSA reduces the number of key generation at join and leave operations to *1* because on each membership change, only the group key is generated by the sibling member of new member, and the other necessary keys are computed by the members. In proposed ECC and DHSA key generation overhead is same but the size of key in ECC is smaller as this is the property of ECC. This feature results in efficiency of group key management for a group with dynamic join and leave.

Table IV shows key encryption overhead at join and leave operations. In proposed ECC when members join/leave the group, one encryption is performed between the new group member and the sibling member of the new member and decreases the key size as comparison to DHSA. This encryption is done based on symmetric algorithm. But in DHSA, there are three encryptions between the new member and the sibling node:

a) One encryption between the new member and the root node.
b) Second encryption between the root node and the neighbour of the new node
c) Finally third encryption between the neighbour node and the node.

There are three encryptions between the member who left the group and the sibling node:

a) One encryption between the sibling node and the root node.
b) Second encryption between the root node and one member in each branch
c) .Finally third encryption between the member who received the group key and remaining members in the branch.

Table IV. Comparison of Key Encryption Overhead of a Node at Join and Leave Operations

Protocols	Join	Leave
DHSA	*3*	*3*
Proposed ECC	*1*	*1*

The communication overhead for proposed ECC and DHSA at join operation is key size (key size of DHSA is *512* bits and proposed ECC key size is *112* bits) b. At join the sibling member of new member just communicates with that member, and delivers the group key to that member, and the other members compute the new group key by applying one-way hash function to previous group key but the size of the key is smaller in ECC. The communication overhead of DHSA at leave is larger than the proposed ECC because the leaving node communicates with every single node to deliver the necessary keys. The message size of proposed ECC is smaller because only the new group key is delivered to remaining members. $\log_2 n - 1$ is the height of the tree in which group members are arranged. If any member leaves the group, then tree is traversed twice from top to bottom and from bottom to top to check the position of the tree. So total communication overhead of a Node is:

$$Comm\_ECC_{leave\_node} = 2\,(\log_2 n - 1) \times key\ size$$

Table V illustrates the proposed method time taken by the nodes when user join, leave, display and read leaf nodes in the key tree. We will calculate the time taken by 100 nodes in a distributed environment for join, leave, and display and read leaf node. During implementation with Elliptic Curve Cryptography for public key exchange, our results are very efficient that takes less time than DHSA. ECC provides more security with less number of keys and it is more scalable than DHSA.

Table V. Time Taken by Nodes for Join, Leave, Display and Read Leaf by using Elliptic Curve Cryptography

Nodes	Join	Leave	Display	Read Leaf	Total
1 node	1 sec	4 sec	2 sec	5 sec	12 sec
2 node	1 sec	9 sec	3 sec	7 sec	20 sec
3 node	2 sec	11 sec	4 sec	10 sec	27 sec
4 node	2 sec	14 sec	5 sec	13 sec	34 sec
5 node	3 sec	17 sec	5 sec	16 sec	41 sec
6 node	3 sec	20 sec	6 sec	17 sec	46 sec
7 node	4 sec	21 sec	6 sec	22 sec	48 sec
8 node	4 sec	22 sec	7 sec	25 sec	63 sec
9 node	5 sec	24 sec	7 sec	26 sec	52 sec
10 node	5 sec	26 sec	7 sec	28 sec	66 sec
20 nodes	10 sec	52 sec	12 sec	45 sec	119 sec
30 node	14 sec	76 sec	15 sec	68 sec	173 sec
40 node	19 sec	119 sec	18 sec	91 sec	247 sec
50 node	22 sec	150 sec	20 sec	115 sec	307 sec
60 node	26 sec	197 sec	25 sec	145 sec	393 sec
70 node	29 sec	245 sec	30 sec	172 sec	476 sec
80 node	33 sec	290 sec	34 sec	201 sec	558 sec
90 node	37 sec	340 sec	37 sec	233 sec	647 sec
100 node	41 sec	390 sec	40 sec	257 sec	728 sec

VI.CONCLUSION

In this paper we have proposed a new group key management approach in distributed network. This protocol is based on logical key hierarchy because in this group members are arranged in hierarchical manner. We have proposed usage of symmetric cryptosystem along with asymmetric cryptosystem. For asymmetric key, Elliptic Curve Cryptography key agreement is introduced. We have used Elliptic Curve Cryptography and it provides much stronger security with smaller key size. The features of this protocol are that, at join no keys are needed to be exchanged between existing members, at leave only one key, the group key, is delivered to remaining members.

Proposed algorithm takes less time when users join or leave the group in comparison to existing one. In ECC, there is only 1 key generation and key encryption overhead at join and leave operation. At join the communication overhead is key size of a node and at leave operation is $2\,(\log_2 n - 1) \times key\ size$ of a node. In future, this work can be further improved using network parameters like network delay and network failure which can further increase the reliability and quality of service of the algorithm.

REFRENCES

[1] Jiang and Hu, "A Survey of Group Key Management," IEEE International Conference on Computer Science and Software Engineering, Vol. 3, pp. 994-1002, December 12-14, 2008.

[2] Fan, Ping, Kuan and Ming, "A Dynamic Layering Scheme of Multicast Key Management," 5th IEEE International Conference on Infomration Assurance and Security, Xian, China, Vol. 1, pp. 269-272, August 18-20, 2009.

[3] Kim, Perrig and Tsudik, "Tree-based Group Key Agreement," ACM Transactions on Information and System Security, Vol. 7, Issue 1, pp. 60-94, February 2004.

[4] Panja, Madria and Bhargave, "Energy and Communication Efficient Group Key Management Protocol for Hierarchical Sensor Networks," IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, Taichung, Taiwan, Vol. 1, pp. 8-15, June 05-07, 2006.

[5] Zhang, Li, Chen, Tao and Yang, "EDKAS: An Efficient Distributed Key Agreement Scheme using One-Way Function Trees for Dynamic Collaborative Groups," IEEE Multi-conference on Computational Engineering in Systems Applications, Bejing, China, pp. 1215-1222, October 2006.

[6] Mortazavi, Kato, "An Efficient Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm: DHSA," IEEE International Symposium on Computer Networks and Distributed Systems, pp. 49-54, Februrary, 23-24 2011.

[7] Zeng, Xia and Su, "A New Group Key Management Scheme based on DMST for Wireless Sensor Networks," 6th IEEE International Conference on Mobile Adhoc and Sensor Systems, Macau, China, pp. 989-994, October 12-15, 2006.

[8] Ye, Zhao, and Guo, "A Safety Group Key Management Scheme in Mobile Adhoc Network," 8th

IEEE International Conference on Reliability, Maintainability and Safety, Chengdu, China, pp. 512-515, July 20-24, 2009.

[9] Chen, Lin, Shen, Hashimoto and Kato, "A Group-Based Key Management Protocol for Mobile Ad Hoc Networks," IEEE Global Telecommunications Conference, Honolulu, Hawaii, pp. 1-5, November 30-December 04, 2009.

[10] Shoufan and Huss, "High-Performance Rekeying Processor Architecture for Group Key Management," IEEE Transactions on Computers, Vol. 58, Issue 10, pp. 1421-1434, October 2009.

[11] Dawood, Mneney, Aghdasi and Dawoud, "An Efficient Hierarchical Group Key Managemnt Protocol for Mobile Ad-Hoc Networks," IEEE 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, Aalborg, Denmark, pp. 619-623, May 17-20, 2009.

[12] McGrew and Sherman, " Key Establishment in Large Dynamic Groups Using One-Way Function Trees," IEEE Transactions on Software Engineering, Vol. 29, Issue 5, pp. 444-458, May 2003.

[13] Malik, "Efficient Implementation of Elliptic Curve Cryptography using Low-power Digital Signal Processor," 12th IEEE International Conference on Advanced Communication Technology, Phoenix Park, Vol. 2, pp. 1464-1468, February 07-10, 2010.

[14] Kristin, "The Advantages of Elliptic Curve Cryptography for Wireless Security," IEEE Wireless Communications, Vol. 11, Issue 1, pp. 62-67, February 2004.

[15] William Stallings, "Cryptography and Network Security Principle and Practice," Fourth Edition.

[16] Shen, Huang and Chen, "A Time-Bound Hierarchical Access Control for Muticast Systems" Proceedings of IEEE International Conference on Machine Learning and Cybernetics, Xian, Vol. 2, pp. 543-548, July 15-17, 2012.

[17] Pushpalatha and Chitra, "GAMANET: A Ganatic Algorithm Approach for Hierarchical Group Key Management in Mobile Adhoc Network," Proceedings of IEEE International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, pp. 368-373, February 21-22, 2013.