International Journal of Computer Applications (0975 – 8887)
Volume 61– No.19, January 2013

Distributed Hierarchical Group Key Management using Elliptic Curve and Hash Function

Uday Pratap Singh
M.Tech, CSE Department
Galgotias College of Engineering & Technology
Greater Noida

Rajkumar Singh Rathore
Asst. Prof. in CSE Department,
Galgotias College of Engineering and Technology
Greater Noida

ABSTRACT

Key management is an essential cryptographic primitive upon which other security primitives are built. However, there are many existing key management schemes that lack on some points and those are not much suitable for ad hoc networks. In this paper we are going to present a distributed hierarchical group key management approach that uses Elliptic Curve Cryptography and Hash function for secure generation and distribution of group key.

General Terms

Key Management in Ad-hoc Network,

Keywords

MANET, Group Key Management, Public Key, Hash Function.

1. INTRODUCTION

A Mobile Ad-hoc Network (MANET) consists of a number of mobile wireless nodes, among which the communication is carried out without having any centralized control. MANET is a self organized, self configurable network having no infrastructure, and in which the mobile nodes move arbitrarily.

In MANET nodes are mobile in nature, due to the mobility, topology changes dynamically. Due to its basic Ad-Hoc nature, MANET is vulnerable to various kinds of security attacks. So the secure key management scheme is prime requirement of mobile ad-hoc network.

Key management is the management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use, and replacement of keys. It includes cryptographic protocol design, key servers, user procedures, and other relevant protocols.[1]

2. 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.

DHSA[2] (Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm), 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.

3. PROBLEM STATEMENT

The key management scheme that was previously being used was DHSA [2] and in this scheme Diffie-Hellman key agreement was used to deliver the group key between nodes. The Diffie-Hellman key agreement scheme has following issues:

· Key size is very large

· Modulo operation takes long time in computation and it makes the computation slow.

4. PROPOSED WORK

Mobile devices have limited battery life so the requirement of key management algorithm is that it has to be computationally fast in order to reduce the power consumption of key management process to insure maximum battery life. Our approach hierarchical key management uses Elliptic curve cryptography for key exchange between leaf nodes. Elliptic curve cryptography provides greater security with small key size and scalar multiplication is computationally fast. So use of ECDHSA protocol will provide more suitable and efficient technique for key management in MANET.

Now I'm going to present my efficient approach, ECDHSA, for distributed secure group communication. The inspiration of this approach is to decrease re-keying overhead at join and leave operation of nodes. ECDHSA focuses on member collaboration for key calculation instead of key delivery by centralized sponsor or co-distributor. For this reason, I'm introducing three basic characteristics of ECDHSA.

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 the corresponding parent node is stored in a list shared by group members. This list will be updated on each membership change and from time to time.
3. All group members have the same capability and are equally trusted and equally responsible for group key generation.

ECDHSA 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, $\{u_1, ..., u_8\}$, 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 member's 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 g^{x1}, g^{x2} 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, g^{x3}, and the associated parent binary code, *00*.

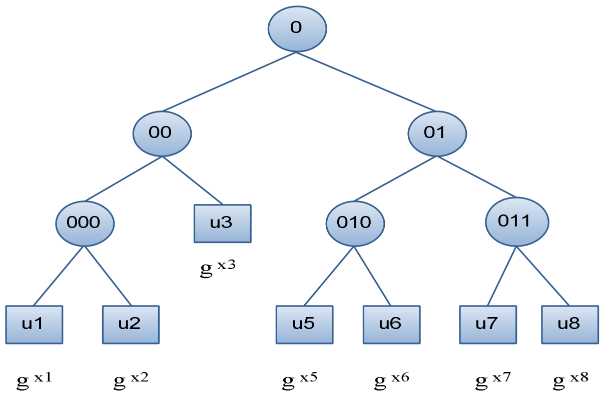

Fig 1: Parent Binary Code for Member Position Discovery

As stated before, the other code type in ECDHSA 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 below.

$$\textbf{Key}_{\textbf{intermediate\_node}} = f\,(\textbf{Key}_{\textbf{group}} \oplus \textbf{Code}_{\textbf{intermediate\_node}})$$

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

$$\boldsymbol{Code_{child\_node} = (Code_{parent\_node} \,||\, Random\ digit).}$$

Fig. 2 illustrates the node code management in the key tree with *8* members, $\{u_1, ..., u_8\}$. 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 3.1 the intermediate node key computation for members $\{u_1,...,u_8\}$ is illustrated. For example, $K_{1,4}$ is calculated as $f(K_G \oplus 04)$.

In ECDHSA, 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.

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

$$K_{1,4} = f(K_G \oplus 04) \qquad K_{5,8} = f(K_G \oplus 08)$$

$$K_{1,2} = f(K_G \oplus 042) \qquad K_{5,6} = f(K_G \oplus 081)$$

$$K_{3,4} = f(K_G \oplus 046) \qquad K_{7,8} = f(K_G \oplus 087)$$

Table 1. LIST OF BINARY CODE AND ASSOCIATED MEMBERS PUBLIC KEY

Parent Binary Code	Member Public key
000	g^{x1}, g^{x2}
00	g^{x3},
010	g^{x5}, g^{x6}
011	g^{x7}, g^{x8}

5. DETAILED DESIGN

To explain the detailed approach, consider our simple example with *8* members illustrated in Figs.1 and 2 for join operation, and Fig. 4 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 Diffie-Hellman key agreement. Here, a member who replies is responsible to authenticate new member. We assume that each group member is equipped with some authentication capability [3].

Once authentication operation is completed, the public key of new member and his/her corresponding parent binary code is stored in the list, and the updated information is multicast 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.

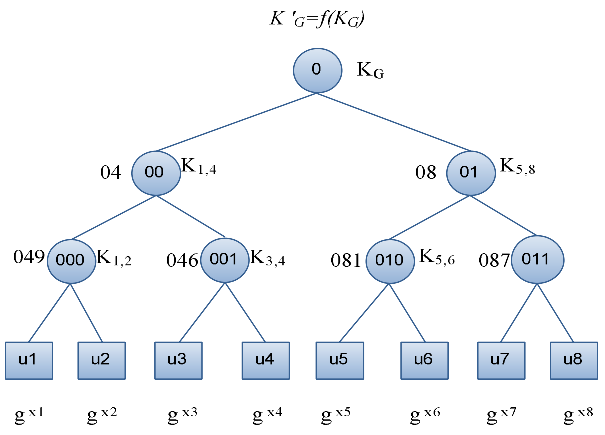

Fig 2: INTERMEDIATE NODE CODE AND CORRESPONDING NODE KEY CALCULATION

5.1 JOIN OPERATION OF NODE

Fig. 1 illustrates a multicast group of 7 members, {*u1*, *u2*, *u3*, *u5, u6, u7, u8*} as current members when a new member *u4* joins the group (Fig.2). Re-keying procedure at join for this example in ECDHSA is as below.

i. ***u4*** broadcasts a hello message for member discovery.

ii. ***u3*** who does not have a sibling node, replies this member.

iii. ***u3*** shares a key with ***u4*** by ECDH key agreement.

This key is $_g^{x3}\,_g^{x4}\,\boldsymbol{P}$.

iv. ***u3*** 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 3.2).

v. ***u3*** calculates the new intermediate node code for his parent.

$Code\_K_{3,4} = (04 \,||\, 6) = \mathbf{046}.$

vi. ***u3*** generates new group key as below.

$K'_G = f(K_G)$

vii. ***u3*** sends *K'G*, and the new node code to *u4* being encrypted by the shared key between them.

$\boldsymbol{u3} \xrightarrow{\textit{unicast}} (K'_{g,}046)_{g}^{x3}\,_{g}^{x4}$

viii. Existing members, {*u1*, *u2*, *u3*, *u5*, *u6*, *u7*, *u8*}, renew the group key as describe in step(vi)

ix. 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 2: LIST OF PARENT BINARY CODE AND ASSOCIATED MEMBERS PUBLIC KEY

Parent Binary Code	**Member Public key**
000	g^{x1}, g^{x2}
001	g^{x3}, $\mathbf{g^{x4}}$
010	g^{x5}, g^{x6}
011	g^{x7}, g^{x8}

As you notice 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 ECDHSA provide this task with just one key delivery. The flow of data for Join operation of node can be seen from the figure 3.

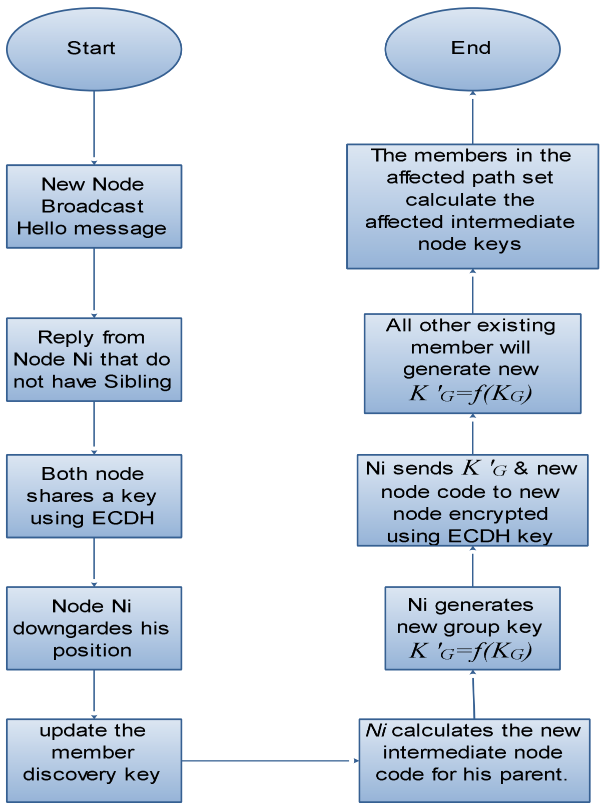
Start
New Node Broadcast Hello message
Reply from Node Ni that do not have Sibling
Both node shares a key using ECDH
Node Ni downgardes his position
update the member discovery key
Ni calculates the new intermediate node code for his parent.
Ni generates new group key K 'G=f(KG)
Ni sends K 'G & new node code to new node encrypted using ECDH key
All other existing member will generate new K 'G=f(KG)
The members in the affected path set calculate the affected intermediate node keys
End

Fig 3: DFD FOR NODE JOIN OPERATION ON ECDHSA

5.2 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.4 illustrates a multicast group of 8 members, {*u1*, *u2*, *u3*, *u3, u5, u6, u7, u8*} when *u8* leaves the group.

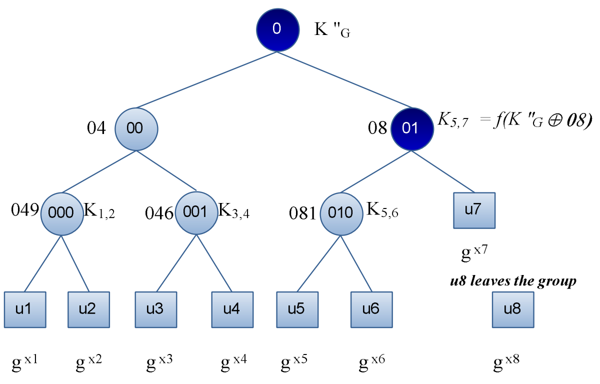

Fig 4 LEAVE OPERATION ON ECDHSA

i. *u7* is promoted to his/her parent position.

ii. *u7* updates the member discovery list by deleting the leaving node's public key, and changes his/her parent binary code. *u7* also informs the other nodes about the updated information.

iii. *u7* generates new group key K''_G , by using symmetric algorithm.

iv. *u7* checks his/her list and use Elliptic Curve Diffie-Hellman 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.

$$u_7 \xrightarrow{\textit{unicast}} u_1{:}(K''_G)_{g^{x1}g^{x7}}$$

$$u_7 \xrightarrow{\textit{unicast}} u_5{:}(K''_G)_{g^{x5}g^{x7}}$$

v. Now u_1 and u_5 multicast the received new group key K''_G ,to members of their branch as follow:

$$u_1 \xrightarrow{\textit{multicast}} u_{2,} \ldots, u_4{:}(K'_G)_{K1K4}$$

$$u_5 \xrightarrow{\textit{multicast}} u_6{:}(K'_G)_{K5K6}$$

vi. Finally the members in affected path calculate the code of the affected intermediate node by the formula below.

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

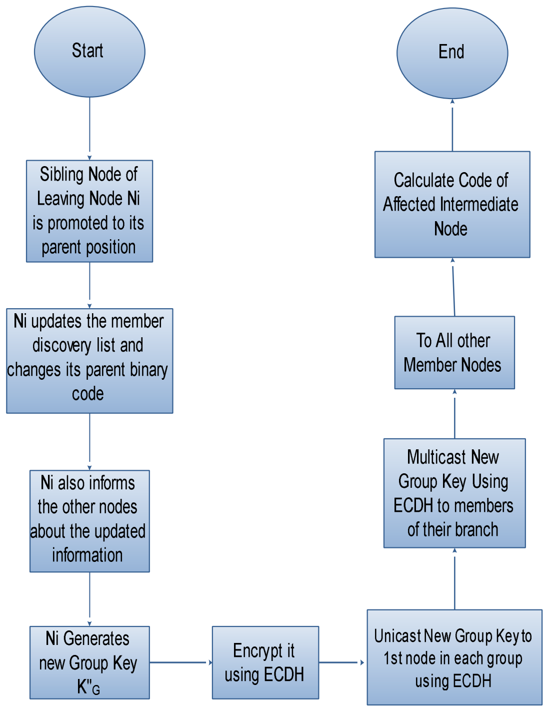
Start
Sibling Node of Leaving Node Ni is promoted to its parent position
Ni updates the member discovery list and changes its parent binary code
Ni also informs the other nodes about the updated information
Ni Generates new Group Key K"G
Encrypt it using ECDH
Unicast New Group Key to 1st node in each group using ECDH
Multicast New Group Key Using ECDH to members of their branch
To All other Member Nodes
Calculate Code of Affected Intermediate Node
End

Fig 5: DFD FOR NODE LEAVE OPERATION ON ECDHSA

Fig 5 shows the flow diagram of the node leave operation. In this a node is being deleted and its sibling node becomes parent node and now this node is responsible secure generation and delivery of group key using ECDH.

Table 3: UPDATING MEMBER DISCOVERY LIST WHEN A MEMBER

LEAVES THE GROUP

Parent Binary Code	**Member Public key**
000	g^{x1}, g^{x2}
001	g^{x3}, g^{x4}
010	g^{x5}, g^{x6}
011	g^{x7}

Table 3 shows the updating of member discovery list after a member leave a group. This is necessary to insure the backward secrecy. I t can be seen from above table after deletion of node u8 its corresponding public key g^{x8} was deleted.

6. IMPLEMENTATION

The simulations are performed using Network simulator (Ns-2),[4] particularly popular in ad hoc networks. The MAC layer protocol IEEE 802.11 is used in all simulations. The Destination Sequence Distance Vector (DSDV) routing protocol is chosen for the simulations. The simulation parameters used are summarized in Table 4:

Table 4: SIMULATION PARAMETERS

Parameter	Value
Simulation time	299 sec
Topology size	1000m X 1000m
No. of nodes	100
Routing protocol	DSDV
Mobility Model	Random Waypoint
MAC	IEEE 802.11
Node Mobility	0 to 20 m/sec
Node Energy	100 joules

6.1 SIMULATION RESULT

The simulation of the proposed scheme gives the following result.

Table 5: SIMULATION RESULT

PDR	0.974112
Control Overhead	496
Normalized Routing Overhead	0.0258468
Delay	0.243332 4669.54
Throughput	783265
Jitter	0.0102397

6.2 ADVANTAGES OF ECDH OVER RSA AND DH

ECDH have many advantages over RSA and DH. Some of the advantages that come with ECDH systems can be briefly explained in terms of its resistance from attacks, strong encryption with less number of bits in key etc. These differences are given in brief below:[5]

6.2.1. More Complex for Attacks

In spite of multiplication or exponentiation in finite field, ECC uses scalar multiplication. Solving Q=k.P (utilized by ECC) is more difficult than solving factorization (used by RSA) and discrete logarithm (used by Diffie-Hellman (DH), ElGamal, Digital Signature Algorithm (DSA)). So ECC is much stronger than other public key agreement and signature authentication methods.

6.2.2. Involvement of Less Number of Bits

ECDH requires much lesser numbers (and thus less number of bits) for its operation thanks to ECDLP. The security level of a 160-bit ECC, 1024-bit RSA, and (160/1024)-bit DSA are similar. Table 6 [6] gives detail about this:

Table 6: COMPARABLE KEY SIZES IN TERMS OF COMPUTATIONAL EFFORT FOR CRYPTANALYSIS [6]

Symmetric Scheme (Key size in bits)	**ECC- Based Scheme (Size of n in bits)**	**RSA/DSA (Modulus size in bits)**
56	112	512
80	160	1024
112	224	2048
128	256	3072
92	384	7680
256	512	15360

6.2.3. Power Consumption

ECC requires less power for its functioning so it is more suitable for low power applications such as handheld and mobile devices.

6.2.4. Computational Efficiency

Implementing scalar multiplication in software and hardware is much more feasible than performing multiplications or exponentiations in them. As ECDH makes use of scalar multiplications so it is much more computationally efficient than RSA and Diffie-Hellman (DH) public schemes. So we can say without any doubt that ECC is the stronger and the faster (efficient) amongst the present techniques.

7. CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In this work I have proposed a new group key management approach in distributed network. This protocol is based on logical key hierarchy. I have proposed usage of symmetric cryptosystem along with asymmetric cryptosystem. For asymmetric key, Elliptic Curve Diffie-Hellman key agreement is introduced.

We conclude our proposal with following of its contributions. We have used Elliptic Curve Cryptography and it provides much stronger security with smaller key size, as shown in Table 6. [6]

ECDH[7] uses scalar multiplication and performing scalar multiplication takes less time in comparison with the modulus and exponent operation performed in previous existing DHSA method. This is the main advantage of our approach because mobile devices have limited battery life and using ECDH for key agreement work faster than using Diffie Hellman Key Agreement. In ECDHSA, intermediate node keys are calculated by group members rather than distributed by a sponsor member.

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.

7.2 FUTURE WORK:

Since we have seen using Elliptic Curve Cryptography have enhanced the security level with small size of key because it provide same level of security that RSA and DH provide with large size of keys. We have also seen that scalar multiplication of ECDH makes it very suitable to be used for MANET because mobile devices have limited battery life and scalar multiplication takes less time in computation. As a future work authentication capability can be added to nodes that reply to the new node. For this purpose we can use ECDSA or any other suitable protocol for Authentication purpose of new nodes. We can use variance of ECDH protocol that will provide more advance security mechanism.

8. ACKNOWLEDGMENTS

I would like to thanks to my dad for financial and moral support for this work.

9. REFERENCES

[1] Key Management Definition. Available at:

[2] http://en.wikipedia.org/wiki/Key_management

[3] S. Anahita Mortazavi, Alireza Nemaney Pour, An Efficient Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm: DHSA *2011 International Symposium on Computer Networks and Distributed Systems (CNDS), February 23-24, 2011.*

[4] Y. Kim, A. Perrig and G. Tsudik, Tree-based Group Key Agreement, A*CM Transactions on Information and System Security (TISSEC)*, Vol. 7/1, Feb. 2004, pp. 60-94, doi: 10.1145/984334.984337.

[5] The Network Simulator - ns-2. Available at

[6] www.isi.edu/nsnam/ns.

[7] Muhammad Yasir Malik Efficient Implementation of Elliptic Curve Cryptography Using Low-power Digital Signal Processor, *ICACT 2010*, ISBN 978-89-5519-146-2 Feb. 7-10, 2010.

[8] Lauter Kristin, The Advantages of Elliptic Curve Cryptography For Wireless Security**,** *IEEE Wireless Communications,* February 2004.

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