Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 3, No.7, 2012

An Efficient Distributed Group Key Management Using Hierarchical Approach with ECDH and Symmetric Algorithm

Uday Pratap Singh*, Rajkumar Singh Rathore
Computer Science & Engineering Department, Galgotias College of Engineering and Technology
Greater Noida (UP), India
* E-mail of the corresponding author: erdev.8@gmail.com

Abstract

Ensuring secure communication in an ad hoc network is extremely challenging because of the dynamic nature of the network and the lack of centralized management. For this reason, key management is particularly difficult to implement in such networks. Secure group communication is an increasingly popular research area having received much attention in recent years. Group key management is a fundamental building block for secure group communication systems. We will present an efficient many-to-many group key management protocol in distributed group communication. In this protocol, group members are managed in the hierarchical manner logically. Two kinds of keys are used, asymmetric and symmetric keys. The leaf nodes in the key tree are the asymmetric keys of the corresponding group members and all the intermediate node keys are symmetric keys assigned to each intermediate node. For asymmetric key, a more efficient key agreement will be introduced. To calculate intermediate node keys, members use codes assigned to each intermediate node key tree. Group members calculate intermediate node keys rather than distributed by a sponsor member. The features of this approach 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.

Keywords: Elliptic Curve, Distributed Group Key Management, Hierarchical Key Management, Mobile Ad-hoc network (MANET).

1. Introduction

Mobile ad hoc network (MANET) can be used in many fields such as battlefields, airplanes, aircraft carriers monitoring, outdoor rescue and other emergent environments without infrastructures. They are easier to be deployed and updated than other conventional wired networks. Most of the users change their locations randomly, so the network topology may change at any time. MANETs can meet this requirement. [1] Mobile ad hoc network (MANET) is a low-cost solution for wireless communication because it does not require any prior investment in a fixed infrastructure.

Recent literatures have sought to address the key management issues in MANETs. Most schemes that have been proposed depend on certificate-based cryptography (CBC) and ID-based cryptography (IBC). In traditional certificate-based public key cryptosystems, a user's public key is certified with a certificate, which is issued by a certification authority (CA). Any participant that wishes to use a public key must first verify its validity using the corresponding certificate. The main concern with this approach is the need for public key distribution.

The concept of ID-based public key systems was introduced by Shamir in 1984. The main idea of such systems is that each user uses his identity securing information, such as a telephone number or email address, as a public key. In other words, the user's public key can be determined directly from his identifying information, rather than having to be extracted from a certificate issued by a CA. ID-based systems enable any pair of users to communicate securely without exchanging public key certificates, without keeping a public key directory and without using the online services of a third party. This is enabled by a trusted Private Key Generator (PKG), which issues a user a private key corresponding to each user's identity when he first joins the network. Therefore, ID-based systems are a powerful alternative to CA-based systems in terms of both efficiency and convenience. However, the PKG represents a single point of failure. If the private key of the PKG is compromised, the entire system is compromised. To counter this, Boneh and Franklin have suggested spreading the PKG private key using threshold cryptography.

In contrast to fixed networks, a centralized public key infrastructure (PKI) or a centralized certification authority is

not feasible in ad hoc networks. Distribution of a signing key and CA functionality over multiple nodes using secret sharing and threshold cryptography is a possible solution to this problem. However, a number of issues must first be resolved. First, the security of the entire network is broken when a threshold number of shareholders are compromised. Second, efficiency is greatly reduced because updating public/private keys requires each node to individually contact a threshold number of shareholders, which represents a significant communication overhead in large-scale MANETs. Third, shareholders joining and evicting need an efficient share update protocol to reduce the communication costs.

Group key agreement (GKA) is another important challenge of key management in MANETs. GKA protocols allow two or more parties to agree on a common group key and exchange information among them over an insecure channel. A key agreement that provides mutual key authentication among parties is called an authenticated key agreement (AKA). Authenticated group key agreement (AGKA) protocol applications proliferate in many modern collaborative and distributed environments. As a consequence, the design of a secure and efficient protocol for group key agreement has received much attention as a significant research area.

Group communication protocols are classified into three main categories:

(1) Centralized (one-to-many) protocols: A key server is solely responsible for managing and distributing group key to group members.

(2) Decentralized protocols: The group is divided into multiple domains. Each domain is managed by a domain controller which is responsible for generating the keys for that domain.

(3) Distributed (many-to-many) protocols: Instead of key server, group members collaborate with each other to establish the group key. The responsibility of the members is equal.

This paper presents an efficient many-to-many group key management protocol in distributed group communication. In this protocol, group members will be managed in the hierarchical manner logically. Two kinds of keys will be used, asymmetric and symmetric keys. The leaf nodes in the key tree are the asymmetric keys of the corresponding group members and all the intermediate node keys are symmetric keys assigned to each intermediate node. For asymmetric key, a more efficient key agreement method will be introduced in place of previously existing Diffie-Hellman method. To calculate intermediate node keys, members will use codes assigned to each intermediate node key tree. Group members calculate intermediate node keys rather than distributed by a sponsor member. The features of this approach should be that, no keys are exchanged between existing members at join, and only one key, the group key, is delivered to remaining members at leave.

This paper is organized as follows. Section II discusses related works. Section III and IV present our proposal. Section V compares our proposal with other protocols. Finally, the conclusion and future work is given in section VI.

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. 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.

In order to establish a group key for secure distributed group communication, many approaches have been proposed. As stated before, most of these approaches are based on different types of n-party Diffie-Hellman key agreement protocol. The other approaches are based on other methods. 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 two of typical approaches in terms of efficiency in communication and computation, and scalability.

EDKAS [3] is very similar to OFT [4] 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 $Ki,j=f(g(Ki), g(Kj))$ 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 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.

DHSA[5] 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. By considering this approach, the re-keying cost is $O(1)$ at join and $O(\log n)$ at leave. However, the cost of modular exponentiation leads the protocol to delay because the protocol needs initiation after each membership change.

Our proposal, ECDHSA (Distributed Group Key Management using Hierarchical Approach with Elliptic Curve Diffie-Hellman and Symmetric Algorithm), uses hierarchical key tree to manage the keys logically. In this protocol, the combination of Elliptic Curve Diffie- Hellman key agreement and symmetric key is used. ECDH[2] 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. By considering this approach, the re-keying cost is $O(1)$ at join and $O(\log n)$ at leave.

3. Design Principles

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.

The public key of each member is generated by Elliptic curve Diffie-Hellman (ECDH) Key Establishment Protocol. The operation of Elliptic curve Diffie-Hellman key exchange protocol is as follows: [6, 7]

• User A and B choose a group of system parameters and make it public: (q, F_q, E, P, n):

q $\in$ {p, 2m}, p is an odd prime, F_q is a prime finite field, Elliptic curve over F_q , and the points whose degree is a big primes n: P $\in$ E (F_q) .

• User A (B) choose the random number a(b) $\in$ Z_n as its private-key K_a (K_b). A (B) calculates $Q_a = K_a P$ ($Q_b = K_b P$) and sends it to each other.

• User A (B) can calculates $S = K_a K_b P$ with its random number and the value getting from B (A).

$S = K_a (Q_b) = K_b (Q_b)$

S is the shared key of A and B.

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, $\{u_1, \ldots, u_8\}$, 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*.

As stated before, the other code type in DHSA 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.

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

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

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

Fig. 2 illustrates the node code management in the key tree with *8* members, $\{u_1, \ldots, u_8\}$. 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 $\{u_1, \ldots, u_8\}$ 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.

$$K'_G = f(K_G)$$
$$K_{1,4} = f(K_G \oplus 04)$$
$$K_{5,8} = f(K_G \oplus 08)$$
$$K_{1,2} = f(K_G \oplus 042)$$
$$K_{5,6} = f(K_G \oplus 081)$$
$$K_{3,4} = f(K_G \oplus 046)$$
$$K_{7,8} = f(K_G \oplus 087)$$

4. 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. 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 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 [8].

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.

4.1. Join Operation Of A 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 $g^{x3}g^{x4}P$.

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 2).

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

Code_K3,4 = (04 || 6)= 046.

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

$K'_G = f(K_G)\, g^{x3}g^{x4}P$

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

$u3 \xrightarrow{unicast} (K'_g, 046)$

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: $K_{3,4} = f(K'_G \oplus 046)$

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

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 DHSA provide this task with just one key delivery.

4.2. 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.3.3 illustrates a multicast group of 8 members, {*u1*, *u2*, *u3*, *u3, u5, u6, u7, u8*} when *u8* leaves the group.

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 Diffie-Hellman key agreement to share a key with one of the member in each branch. Then, it unicast new group key to each of them.

$u_7 \xrightarrow{unicast} u_1:(K''_G)_{K1K7}$

$u_7 \xrightarrow{unicast} u_5:(K''_G)_{K5K7}$

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

$u_1 \xrightarrow{multicast} u_2, \ldots, u_4:(K'_G)_{K2K4}$

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

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

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

5. Comparison

In this section we compare and analyze our proposal with two of other distributed key management approaches EDKAS and 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 Tables 4, 5 and 6 summarize our comparisons, focusing described metrics.

Table 4 shows the key generation overhead at join and leave operations. ECDHSA just like 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. This feature results in efficiency of group key management for a group with dynamic join and leave. The key generation overhead of EDKAS at join is larger than our approach.

Table 5 shows key encryption overhead at join and leave operations. In ECDHSA when a member joins the group, one encryption is performed. This encryption is done based on symmetric algorithm. The number of key encryption in DHSA is same as or protocol. The number of key encryption for EDKAS is larger than of ECDHSA that at join and leave because in EDKAS the sponsor encrypts the necessary keys by each member's individual key, and sends directly to that member.

Table 6 illustrates the communication overhead at join and leave. The communication overhead for ECDHSA at join operation is *1* because 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.

The communication overhead of EDKAS at join/leave is larger than the others because the sponsor node communicates with every single node to deliver the necessary keys. DHSA and ECDHSA has the same communication overhead at leave but the key size of ECDHSA is smaller for same level of security as in DHSA with large size of key.

6. Conclusion and Future Work

6.1 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.

At the end, 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 5.
- ECDH 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.
- 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.

6.2. 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. In future we can use any other efficient protocol for Authentication purpose of new nodes.

References

Dahai DuL and Huagang Xiong, "A Dynamic Key Management Scheme for MANETs," Cross Strait Quad-Regional Radio Science and Wireless Technology Conference 2011.

SECI, "Elliptic curve cryptography," Certicom Corp. Sep. 2000.

J. Zhang, V. Li, C. Chen, P. Tao and S. Yang, “EDKAS: An Efficient Distributed Key Agreement Scheme Using One Way Function Trees for Dynamic Collaborative Groups,” IMACS Multiconference on CESA, Bejing, China, pp. 1215-1222, Oct. 2006, doi: 10.1109/CESA.2006.313506.

D.A. McGrew and A.T. Sherman, “Key Establishment in Large Dynamic Groups Using One-Way Function Trees,” IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29/5, pp.444-458, May 2003, doi: 0.1109/TSE.2003.1199073.

S. Anahita Mortazavi, Alireza Nemaney Pour **“**An Efficient Distributed Group Key Management using Hierarchical Approach with Diffie-Hellman and Symmetric Algorithm: DHSA” IEEE 2011.

Standard specifications for public key cryptography, *IEEE standard,*pI363,2000.

Williams Stallings, Cryptography and Network Security, *Prentice Hall,*4th Edition, 2006.

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

About Authors:

UDAY PRATAP SINGH

Uday Pratap Singh pursuing his M.Tech in Computer Science and Engineering from Galgotias College Of Engineering and Technology, MahaMaya Technical University, Noida, India. He had completed his B.Tech from SIET, Allahabad. He have two year of experience as a lecturer in a reputed Institution. He has contributed several research papers at National/International Journals. His area of interest includes Computer Networks, Operating System, Cryptography and Network Security.

RAJKUMAR SINGH RATHORE

Rajkumar Singh Rathore is working as Assistant Professor in Computer Science and Engineering Department at Galgotias College of Engg. & Technology, Greater Noida (U.P.). He has written books on "Database Management System", "Operating System", "IT Infrastructure and Its Management" and "Software Engineering"for undergraduate and postgraduate courses. He has contributed several research papers at National/International Conferences/Journals. He is the reviewer of many International Journals. He is the member of ISTE, CSI, IEEE, IAENG, IACSIT, CSTA etc. His area of interest includes ITIM, Operating System, Database Management System, Compiler Design and Software Engineering.

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}

Table 2: List of Parent Binary Code and Associated Members Public Key

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

Table 3: Updating Member Discovery List When a member leaves the group

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

Table 4: Comparison of Key Generation Overhead at Join and Leave Operations

protocols	join	leave
EDKAS	2 log *n*	2 $\log_2$ n-2
DHSA	1	1
ECDHSA	1	1

Table 5: The Comparison of Key Encryption Overhead At Join/ Leave Operations

protocols	join	leave
EDKAS	*n*	n-2
DHSA	1	$2\log_2$n-2
ECDHSA	1	$2\log_2$n-2

Table 6: The Communication Overhead At Join and Leave Operations

protocols	join	leave
EDKAS	*n-1*	n-2
DHSA	1	$2\log_2$n-2
ECDHSA	1	$2\log_2$n-2

Fi ry

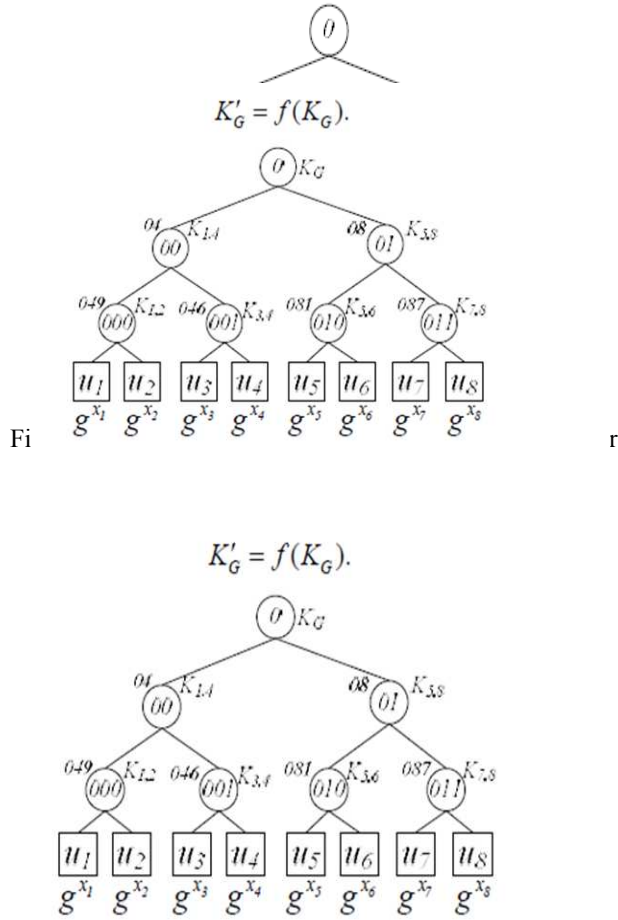

Figure 2. Intermediate node code and corresponding node key calculation

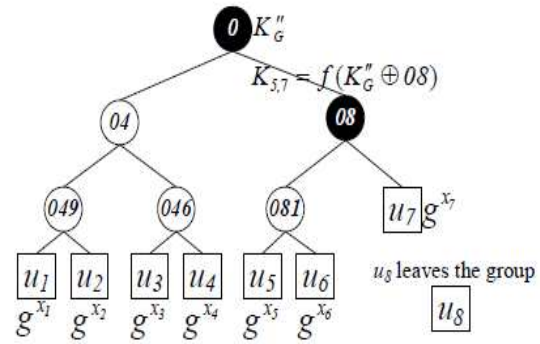

Figure 3. Leave Operation on ECDHSA