Cost Effective Group Key Agreement Protocol for Ad-Hoc Networks

S.Devakani Packiavathy
Assistant Professor/CSE
Dr.Sivanthi Aditanar College Of Engineering
Tiruchendur.

P.Chanthiya
Assistant Professor/CSE
Dr.Sivanthi Aditanar College Of Engineering
Tiruchendur.

K.Kumar
Professor/CSE
Government College Of Technology,
Coimbatore

ABSTRACT
Group communication plays a vital role in collaborative and group-oriented applications. It supports the dissemination of information from a sender to all the receivers in a group. The information needs to be encrypted using a secret key to ensure security in group communication over open networks. Group key establishment involves creating and distributing a common secret key for all group members. This paper proposes a group key agreement protocol that minimizes computation and storage overhead of nodes involved in group communication. Hence this method is mainly suited for ad-hoc networks in which nodes have limited resources and short life time. Group member nodes form a logical tree structure among them. Group key is generated from the leaf to the root node. Then root node unicasts the computed key to every other member. This key is used for the encryption and decryption of group messages. The proposed scheme uses key tree structure to minimize the number of operations on each node. Elliptic Curve Diffie-Hellman minimizes computational overhead at each node and the key with smaller bit size can achieve higher security levels. The tree structure is always maintained as height balanced to minimize the key convergence time among group nodes.

1. INTRODUCTION
Applications in wireless networks have reached an extraordinary success in the last years. wireless communications are usually transmitted by radio frequencies, which are highly vulnerable, since it is possible for an unauthorized party, located within the transmitter’s communication radius, to listen to the communication. An infra-structure less wireless networks is known as ad-hoc networks. There is no centralizing entity like the access point. Ad-hoc networks are used in wide range of applications like military operations, rescue operations, remote site construction, communication among a group of islands or ships, communication among disaster management groups, conferencing without the support of a wired infrastructure, and interactive information sharing.

1.1. Group Communication
Group communication means dissemination of information from a sender to a group of intended receivers. Consider a scenario where there are n users in a network, of which some t (t<<n) of them would like to discuss on a common concern. These t parties (called privileged users) must communicate among themselves over a public channel in a secure manner in that others must not be able to listen in to the conversation between these t parties. Therein, lays the need to find new technology for such confidential communication which is effectively called secure group communication or secure conferencing. This paper approaches the problem of establishing safe group communication among wireless devices on an ad hoc network.

1.2. Group Key Agreement
Group Key Agreement means that multiple parties want to create a common secret key that to be used to exchange information securely. This group key should be updated when there are membership change occurs due to node mobility (when the new member joins or current member leaves) in the group. This can be done either periodically or at the time of every membership changes.

1.3. Contributory Group Key Agreement
The group key is generated in a contributory fashion, where all members contribute their own share in computing the group key. This approach is fault tolerant and diminishes the risks of vicious key generation by a single entity.

1.4 Objective
The main aim is to establish a secret key among all group members in order to preserve the security of group communication. In case of a change occurs in the group membership by joining or leaving the group, the group key should be updated to maintain backward secrecy and forward secrecy. This approach addresses a new technique to form a hierarchical key tree to efficiently compute and update group keys and makes use Elliptic Curve Diffie-Hellman key agreement protocol to compute the group key in a distributive manner. Only the secret group key establishment and management are analyzed. The multicast communication, the ad hoc routing and the congestion and error control are not within the scope of this work.

2. LITERATURE REVIEW
The two party key exchange protocols such as Diffie-Hellman, can be extended to group key generation and management. Many contributory group key agreement schemes have been developed to extend the two party key establishments to the group scenario.

2.1. The Diffie-Hellman Algorithm
This algorithm allows the establishment of a cryptographic secret key between two entities by means of data exchange through an insecure communication channel. The algorithm executed between two entities A and B is defined as follows:

- A and B agree on two randomly chosen numbers p & g, so that p is a large prime number and $g < p$;

- A chooses a secret random number S_A and B chooses a secret random number S_B;
- A computes a public value $T_A = g^{S_A} \, mod \, p$ and B computes a public value $T_B = g^{S_B} \, mod \, p$
- A sends T_A to B and B sends T_B to A;
- A computes $T_B^{S_A} mod \, p = (g^{S_B})^{S_A} mod \, p$
- B computes $T_A^{S_B} mod \, p = (g^{S_A})^{S_B} mod \, p$.

Since $(g^{S_B})^{S_A} mod \, p = (g^{S_A})^{S_B} mod \, p = K$, these two entities share a secret cryptographic key K. The security of this algorithm is based on the difficulty to calculate the secret key $K = g^{S_A S_B} mod \, p$ despite of knowing the public values $g^{S_A} \, mod \, p$ and $g^{S_B} \, mod \, p$, when the prime number p is sufficiently large.

2.2 The Ingemarsson Et Al. Protocol [1]

This protocol, presented by Ingemarsson et al.,(1982) was one of the first attempts to extend the Diffie-Hellman algorithm to group communications. The n group members must be arranged in a logical ring. At the beginning of the protocol execution, each participant $M_i \, , i \in [1 \dots n]$ generates a random value S_i that is its contribution for the group key. At each round, each participant raises the value received in the previous round to its secret value S_i and sends it to the next node in the sequence. Therefore n messages are exchanged among the n group members at each round. After n-1 rounds every node computes the same group key $K_n = g^{S_1 S_2 \dots S_n}$.The main disadvantages of this approach are the high number of messages exchanged and the high number of exponential operations executed.

2.3 The Burmester And Desmedt Protocol [2]

This protocol was presented by Burmester and Desmedt (1994) and is executed in three rounds. Each participant $M_i \, , i \in [1 \dots n]$ executes the following operations:

- generates a secret random value *Si* and broadcasts $Z_i = g^{S_i}$ to the other participants
- computes and broadcasts $X_i = \left(\frac{Z_{i+1}}{Z_{i-1}}\right)^{S_i}$ to the other participants;
- computes the group key $K_n = Z_{i-1}^{nS_i} X_i^{n-1} X_{i+1}^{n-2} \dots X_{i-2} \, mod \, p$

This group key has the form $K_n = g^{S_1 S_2 + S_2 S_3 + S_3 S_4 + \dots + S_n S_1}$ and shares the security characteristics presented by the Diffie-Hellman algorithm. This protocol is efficient with respect to the total number of rounds. This characteristic could allow faster execution, but each round requires n simultaneous broadcasts. Simultaneous broadcasts are usually not possible, even in wireless networks, because there can be only one broadcast message at a given moment. Due to this characteristic, the deployment of this protocol must use sequential broadcast messages. Since each broadcast message acts like a round, there is no longer a low number of rounds advantage. Another disadvantage is that this protocol makes use of a high number of exponential operations.

2.4 The Hypercube Protocol [3]

The Hypercube protocol was presented by Becker and Willie (1998) [3] and intents to overcome the high number of messages needed by the protocol proposed by Ingemarsson et al. by logically arranging the nodes in a hypercube. For a network consisting of four nodes positioned as a square, a key is established between A and B $\left(g^{S_a S_b}\right)$, and another key between C and D $\left(g^{S_c S_b}\right)$. These keys are used to establish a single key $\left(g^{\left(g^{S_a S_b}\right)\left(g^{S_c S_b}\right)}\right)$ among the four entities. This behavior can be generalized for higher numbers of nodes, as long as the number of participants equals 2^d, This protocol executes in d rounds. The number of exponentiation operation and round is increased with the increase in the number group members.

2.5 Group Diffie Hellman Procol (Gdh) [4]

Steiner et al.(2000) proposed CLIQUES protocol suite that consist of group key agreement protocols for dynamic groups called Group Diffie-Hellman (GDH). It consists of three protocols namely GDH.1, GDH.2 and GDH.3. These protocols are similar since they achieve the same group key but the difference arises out of the computation and communication costs.
The GDH.3 protocol was proposed in order to minimize the demanded computational cost of other two GDH approaches. It comprises four steps. At the first stage, contributions are collected from the n-2 first group members by means of a single message sent from one member to the next that gathers all the previous contributions. At the second stage, M_{n-1} adds its contribution to the received message and broadcasts this new message to the n-2 first members. At the third stage each member factors out its own contribution and sends this result to the last group member. At the last stage, M_n collects all the sets from the previous stage, raises each one of them to its contribution S_n and broadcasts these results to the other group members, allowing them to compute the group key.

Based on the description above, the group members know each other. However, it is difficult for a node to know other members of the group in advance in a dynamic MANET, which results in a chaotic situation while endeavoring to establish the security of group communications.

2.6 Tree Based Group Diffie Hellman Protocol (Tgdh) [5]

Y. Kim, A. Perrig, and G. Tsudik,(2004) did some improvements over contributory group key agreement. In doing so, it unifies two important trends in group key management:

(1) key trees to efficiently compute and update group keys

(2) Diffie–Hellman key exchange to achieve secure and fully distributed protocols.

They developed a simple, secure, robust and efficient key management solution, called Tree-based Group Diffie–Hellman (TGDH). The number of levels for the creation of the group key is shrinking to the logarithm of the group size. Figure 2.1 shows an example of a key tree. This scheme lack efficient key management mechanisms for members to participate in or leave MANETs dynamically.

Number of rounds and number of exponentiation operation needed to calculate the group key is getting increased with the increase in the height of the tree.

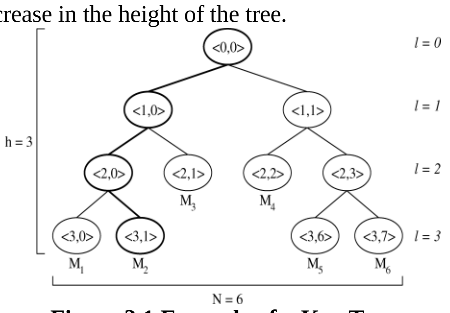


Figure 2.1 Example of a Key Tree

At every time if a user joins or leaves, one sponsor node is chosen to start the rekeying mechanisms. The sponsor node has to send the new updated key tree with all blinded keys to every other group users, so that they can calculate the new group key. Hence it needs large bandwidth to send the information. Also every member needs to know every key along the path from itself to the root node. It requires lot of information has to be stored in memory for processing.

2.7 Motivation and Contribution

The inefficiencies addressed from the existing approaches, motivate to develop a new technique such that the group key is generated from the contribution of all group members with less number of operations, less number of rounds, and reduces the usage of resources .Thus to make the technique more suitable for ad-hoc networks.

3. PROPOSED SYSTEM

The intent of this work is to design a cost effective protocol for generating a group key for ad-hoc networks. The group key is computed based on Tree based ECDH algorithm, which is then distributed to all group members. Each group is organized in a logical key hierarchy which reduces the complexity for a member who join or leave from O(n) to O(log n). The members in a group coordinate with each other to generate the group key. The process of group key generation takes place from leaf to the root node.

3.1 The Elliptic Curve Diffie Hellman Algorithm

Elliptic Curve Cryptography (ECC) is a public key cryptosystem based on elliptic curves. The elliptic curve cryptosystem (ECCS) is a crypto-algorithm method of utilizing a discrete logarithm problem (DLP) over the points on an elliptic curve. The attraction of ECC is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead. However, the methods for computing general elliptic curve discrete logarithms are much less efficient than those for factoring or computing conventional discrete logarithms and it indicates that more computation time is required to break ECC system.ECC has become the cryptographic choice for ad hoc networks and less powerful communication devices .

The elliptic curve Diffie-Hellman (ECDH) is a variant of the Diffie-Hellman (DH) key agreement protocol, using elliptic curve cryptography that allows two parties to establish a shared secret key (session key) over an insecure channel. The ECDH with 160-bit key lengths provides the same security level to a 1024-bit DH secret sharing protocol.

3.1.1. Elliptic Curves

An Elliptic Curve is usually defined over two finite fields: the prime finite field F_p containing p elements and the characteristic 2 finite field containing 2^m elements. Prime finite field is used in software implementations, while characteristic finite field is used in hardware implementations. This project focuses on the prime finite field.

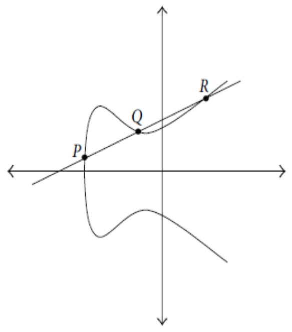


Figure 3.1 A Elliptic Curve with P+Q+R=O

An elliptic curve is topologically equivalent to a torus over a finite field GF (a Galois field of order p), as shown in Figure 3.1 and comprises a set of finite points (x_i, y_i), where coordinates x_i, y_i are integers and satisfy

$$y^2 + a_1xy + a_3 y = x^3 + a_2x^2 + a_4x + a_6. \quad (1)$$

The coefficients a_i are elements in GF(p), since the field GF(p) ($p \in$ prime) is generally adopted in cryptographic applications, such that the elliptic curve can be translated into $Ep(a, b)$

$$y^2 = x^3 + ax + b \pmod p, \qquad (2)$$

Where a and b belong to GF (p).and a and b satisfies the condition $4a^3+27b^2 \neq 0$. Here the elements of the finite field are integers between 0 and p – 1. All the operations such as addition, substation, division, multiplication involves integers between 0 and p – 1. This is modular arithmetic. The prime number p is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC specifies curves with p ranging between 112-521 bits .Considering two points on curve $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, and a point at infinity O ,which also lies on the elliptic curve, where $P \neq Q \neq O$, points P, Q and O satisfy the following rules:

(1) $P + O = O + P = P, P + (-P) = O,$

(2) $(x_1, y_1) + (x_1 - y_1) = P + (-P) = O,$

(3) $P + Q = R = (x_3, y_3)$ on the curve, where $x_3 = \lambda_2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, where

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & if\ P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & if\ P = Q \end{cases}$$

(3)

Cryptographic schemes based on ECC rely on scalar multiplication of elliptic curve points. Given an integer k and a point P ∈ E_p(a,b), scalar multiplication is the process of adding P to itself k times. The result of this scalar multiplication is denoted k×P or kP. Scalar multiplication of elliptic curve points can be computed efficiently using the addition rule together with the double-and-add algorithm or one of its variants. Consider the equation $Q = kP$, where $Q, P \in Ep(a, b)$. It is relatively easy to calculate Q given k and P, but it is relatively hard to determine k given Q and P. This is called the discrete logarithm problem for elliptic curves.

3.1.2. Analog Of Diffie-Hellman Key Exchange

For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters. The domain parameters for Elliptic curve over Fp are **p**, **a**, **b**, **G**, **n** and **h**. p is the prime number defined for finite field Fp . a and b are the parameters defining the curve y2 mod p= x3 + ax + b mod p. G is the generator point (x_G, y_G), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The order n of a point G on an elliptic curve is the smallest positive integer n such that nG=O The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = #E(Fp)/n. #E(Fp) is the number of points on an elliptic curve

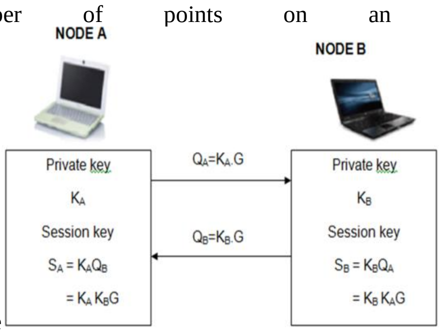


Figure 3.2 ECDH Key Agreement Protocol

Both end have a key pair consisting of a private key k (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key Q = k * G (G is the generator point, an elliptic curve domain parameter). Let (k_A, Q_A) be the private key - public key pair of A and (k_B, Q_B) be the private key - public key pair of B.

- The end A computes $S_A = (x_A, y_A) = k_A * Q_B$
- The end B computes $S_B = (x_B,y_B) = k_B * Q_A$
- Since $k_AQ_B = k_Ak_BG = k_Bk_AG = k_BQ_A$. Therefore $S_A= S_B$ and hence $x_A= X_B$
- Hence the shared secret is x_A or x_B.

Since it is practically impossible to find the private key k_A or k_B from the public key Q_A or Q_B, it's not possible to obtain the shared secret for a third party. Elliptic curve cryptography is effective for power saving due to the usage of lesser number of bits for secure communication

3.2 Tree Structure

The logical tree structure formed among group members is totally different from the one which is used with TGDH approach, where only leaf nodes are the group members. In this system, a binary search tree is organized based on the ip addresses of the members. The steps to form a logical tree is given below

1. The members which starts the group will act as a root node
2. When a new member M_i joins, it sends its ipaddress with join message.
    - If its ipaddress < parent node's ip address

      The node is joined at left side of the parent node
    - If its ipaddress > parent node's ip address

      The node is joined at right side of the parent node

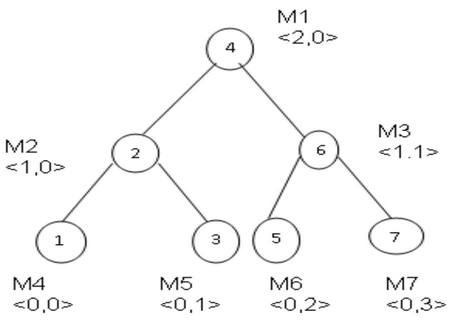


Figure 3.3 Tree Structure of the Proposed System

3.3 Key Structure

Our group key management scheme uses a TGECDH algorithm for computing the group key from the contributions of all group members using a binary search tree. A binary search tree T is a key tree in which every node can be denoted as <h, i> where h is the height (level) of the node and i is the index of the node at level h. Thus, every node is a group member and uniquely identified. Each member at <h, i> is associated with a private key, PR<h, i>, a public key, PU<h, i>.and a secret key SK<h,i>. Initially every node assigns their secret key value to be 0. The private key of the group member is defined by

$PR\langle h, i\rangle = r_i$ (3.1)

Where ri is a random integer assigned by the group member Mi. The PU<h, i> is computed from the private key PR<h, i> from equation (3.2). Elliptic Curve Diffie Hellman algorithm is used to compute the group key. It is effective for power saving due to the usage of lesser number of bits for secure communication. G is a base point of an Elliptic Curve Equation Ep(a,b), '•' is the scalar multiplication operation, and both $E_p(a,b)$ and G are shared by all group members in advance.

$PU\langle h, i\rangle = PR\langle h, i\rangle \bullet G.$ (3.2)

Public key is a point on the elliptic curve having two co-ordinates x and y. The key at the root node, i.e. PR<h, 0> represents the group key shared by all group members. Group key computation is started at leaf members.

3.4. Group Key Agreement Module

If the member has a sibling, they will establish a two party ECDH to compute the shared secret key (SK) between them.

$SK\langle h, i\rangle = PR\langle h,i\rangle \bullet PU\langle h,i+1\rangle$ (3.3)

$SK\langle h,i+1\rangle= PR\langle h, i+1\rangle \bullet PU\langle h,i\rangle$ (3.4)

SK<h, i> = SK<h,i+1>.As shown in figure 3.4, the leaf nodes agree on a shared key with its siblings. Then the secret key of these two nodes are updated to have the new shared secret value. If a node does not having the sibling node, then its secret key is assigned to its public key value. Once the shared key is generated, these nodes send that key to its parent. Then the parent node takes only the x co-ordinate of the received key and calculates the scalar multiplication of the received key's x co-ordinate with the generator point and updates its public key as,

PU<parent> = (PR<parent> **.** x co-ordinate (Received SK)) **.**G (3.5)

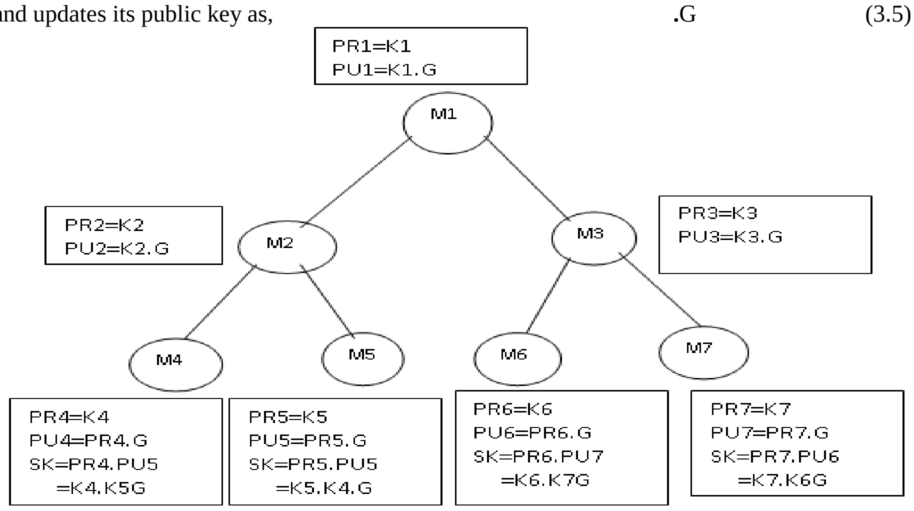


Figure 3.4 Members At The Leaf Performing Group Key Agreement Module

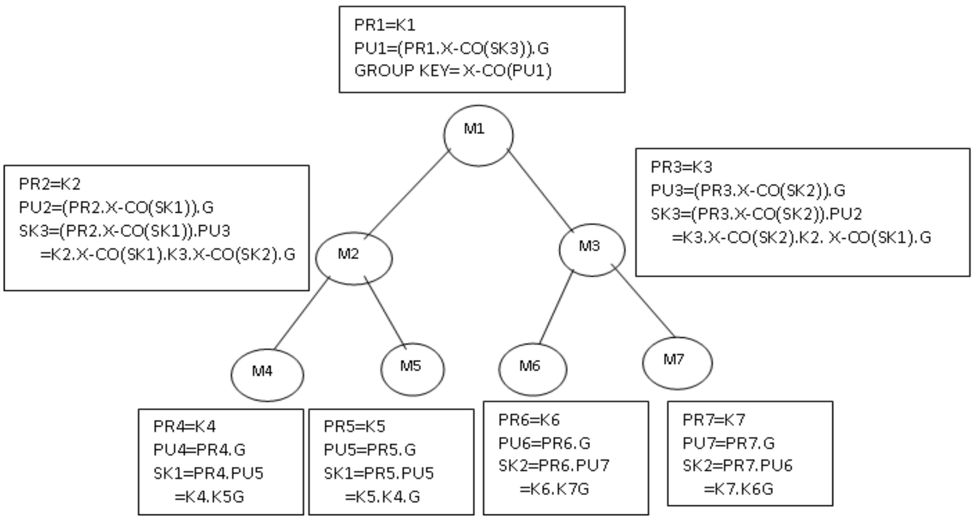


Figure 3.5 Members At The Intermediate Nodes And The Root Performing Group Key Agreement Module

Then the same procedure is repeated for the parent node until the parent node is a root node. But if the node is an intermediate node, the secret keys are updated as follows
SK <node> = (PR<node> **.**x coordinate(received key) **)**.PU(sibling) (3.6)

These steps are shown in figure 3.5 .The x co-ordinate of the updated public key at root node is taken as a group key. This key is securely unicasted to each group member by the root node.

3.5 TGECDH Protocols

In this section, the two basic protocols that form the TGECDH protocol suite: join, leave are introduced.

- Join: a new member is added to the group.
- Leave: a member is removed from the group.

These protocols share a common framework with the following notable features:

3.5.1 Join Protocol

Assume that the group has n members $\{M_1, M_2, \dots\dots, M_n\}$. The new member M_{n+1} initiate the protocol by broadcasting a join request message that contains its own public key and the ip address. Each current member receives this message and determines the insertion point in

- Each group member contributes an equal share to the group key. The key is computed as a function of all current group members' shares.
- Each share is secret (private to each group member) and is never revealed.

After every membership change, all remaining members independently update the key tree structure. This protocol also requires each member to know all public keys in the entire key tree. As part of the protocol, a group member can take on a special sponsor role, which involves computing keys and broadcasting to the group. Each broadcasted message contains the sender's view of the key tree, which contains each Public key known to the sender. Any member in the group can take on this responsibility, depending on the type of membership event. This makes the handling of future membership changes more efficient and robust.
the tree. The insertion point is based on its ip address. The sponsor is the node at which the new member joins or the sibling of the new node.

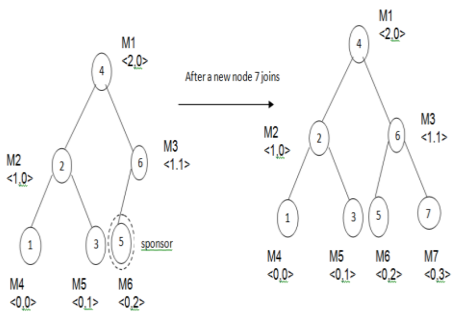


Figure 3.6 Tree Structure after a Join Operation

Next, the sponsor unicasts all nodes ip addresses and their public keys in the key tree. The new member constructs the key tree accordingly. Then the sponsor proceeds to update its share by invoking the group key agreement module.

From the figure 3.6, when a node M_7 joins in a group, it broadcasts its ip address and the public key value to every group members. Every member in the group updates its key tree by receiving the new member's information. M_6 is the sponsor node. M6 broadcasts the ip addresses and the public keys of nodes (M_1,M_2, M_3, M_4, M_5) which are already present in the tree, So as the new member form the key tree. Then M6 invokes the group key agreement module, in which it updates its secret key value by establishing a shared secret key with its sibling node and send the value to its parent. Then the parent node performs the same operation by invoking the group key agreement module. Finally the root node unicasts the new group key to every group member.

3.5.2 Leave Protocol

Consider the group of n members. Assume that member M_d ,d$\in \{1 \ldots n\}$ leaves the group by broadcasting the leave message that contains its ip address. Upon receiving it every other member in a group updates its tree structure using binary search tree deletion rules. There are three cases for updating the tree and selecting a sponsor node, which is defined as follows

- If the leaving node
    - Does not have any children,
        - Take the sibling as the sponsor node if it is present ,otherwise parent node is the sponsor node
        - Then just remove the node from the tree
    - Have only one child
        - Replace the leaving node by the child node
        - Then the same node is taken as a sponsor node
    - Have two children
        - The sibling or parent node of the smallest valued node in the right sub tree if the leaving node is replaced by the node with smallest value from its right sub tree
        - Or the sibling or parent node of the largest valued node in the left sub tree if the leaving node is replaced by the node with largest value from its left sub tree After the selection of sponsor node,

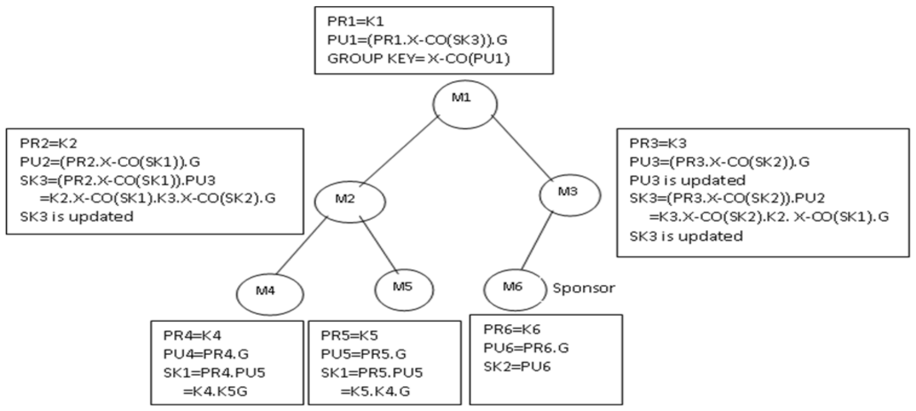


Figure 3.7 after a Node 7 Leaves the Group from Figure 3.3

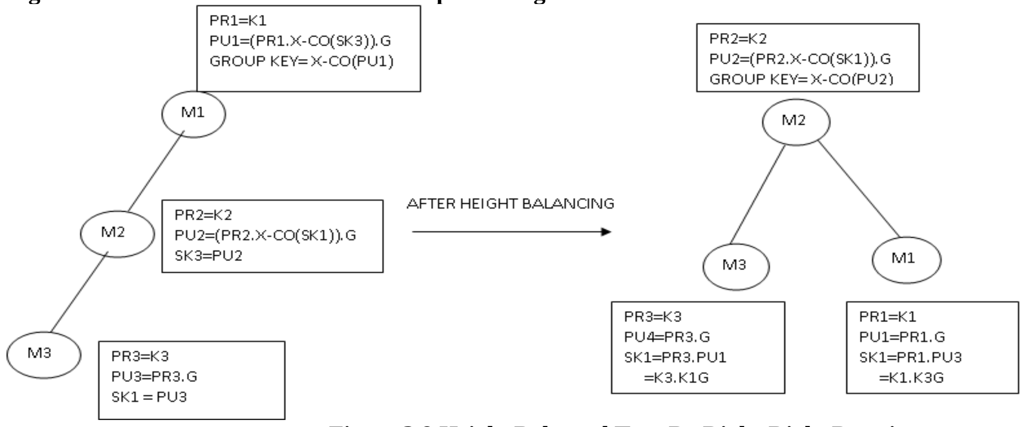


Figure 3.8 Height Balanced Tree By Right-Right Rotation

it invokes the group key agreement module and thus the group key is recomputed.
The proposed mechanism needs to recalculate the key value from the part of the joining (leaving) node sub tree without recalculating the entire tree, thus saving a tremendous amount of operational time. Thus the proposed approach is effective and efficient.
The number of messages exchanged among the group members and the number of rounds for constructing the group key is based on the height of the tree. To reduce the height of the tree, and to make the tree becomes balanced, the tree is height balanced dynamically using AVL tree rules when a node joins or leaves from the group.The adjusting procedures are classified into categories of left rotation (LL), left-right rotation (LR), right rotation (RR), and right-left (RL) rotations. The procedure is described as follows.

- According to the binary search tree rule, place (or remove) the new joining (or leaving) node in (or from) the correct place, depending on its ID (MAC or IP address).
- Calculate the BF of each node. Balance factor (BF) denotes the height difference of left and right sub trees, BF= $|HL - HR|$, where HL denotes the height of a left sub tree, and HR denotes the height of a right sub tree. If it does not belongs to (0, −1, or 1), the tree loses balance.
- Adopt LL, RR, LR, and RL mechanisms to perform unbalanced adjustments
- Reconstruct the balanced tree.

In figure 3.8, at node M1, the balance factor is two. Therefore adjusting procedures are executed to balance the tree. It reduces the height of the tree and reduces the number of rounds to be carried out to find out the group key. In figure 3.8, number of rounds needed before height balancing is 2. Then it is reduced to 1 because of height balancing. The system only needs to modify the link point of the data structure and thus takes $O(1)$ time complexity. As the key tree is unbalanced, in a worst case scenario, the adjusting procedure must move a leaf node from the bottom to the root position and at most takes O (log n).

3.7 Secure Group Communication

After getting the unicast message from the root node, every node starts to communicate to other members in a group. Every group message is encrypted using the received group key at the sending node and decrypted at the receiving nodes. Only the legitimate users can encrypt and decrypt the group messages. The scheme can ensure that the data transmission is confidential and authentic. To make this method more secure, some security functions like hashing and message authentication code can be applied on the group key and group messages.

4. IMPLEMENTATIONS

This proposed TGECDH protocol needs to implement the tree structure and ECDH group key agreement at every level of the tree structure to form a group key.

4.1 Implementation Of ECDH

As mentioned in 1.4.4, to establish a shared secret key the elliptic curve domain parameters need to be agreed up among the members. Hence Ep(a,b) ,n, G values are shared secretly among group members when the member joins to the group. Ep(a,b) defines the elliptic curve equation y^2 mod p= x^3 + ax + b mod p and n is the order of the curve, G is the one point on the curve , known as generator point.

4.1.1. Generation Of Possible a And b Values

For the elliptic curve equation y^2 mod p= x^3 + ax + b mod p , the coefficients should satisfy the condition, $4a^3+27b^2 \neq 0$. if the value of p is set to 23, a and b will take the value from [0-22]. The possible a and b values are,
(0 , 1) (0 , 2) (0 , 3) (0 , 4) (0 , 5) (0 , 6) (0 , 7) (0 , 8) (0 , 9) (0 , 10) (0 , 11) (0 , 12) (0 , 13) (0 , 14) (0 , 15) (0 , 16) (0 , 17) (0 , 18) (0 , 19) (0 ,20) (0 , 21) (0 , 22) (1 , 0) (1 , 1) (1 , 2) (1 , 3) (1 , 4) (1 , 5) (1 , 6) (1 , 7) (1 , 8) (1 , 9) (1 , 10) (1 , 11) (1 , 12) (1 , 13) (1 , 14) (1 , 15) (1 , 16) (1 , 17) (1 , 18) (1 , 19) (1 , 20) (1 , 21) (1 , 22) (2 , 0)
(2 , 1) (2 , 2) (2 , 3) (2 , 4) (2 , 5)
(2 , 6) (2 , 7) (2 , 8) (2 , 9) (2 , 10)
(2 , 11) (2 , 12) (2 , 13) (2 , 14) (2 , 15)
(2 , 16) (2 , 17) (2 , 18) (2 , 19) (2 , 20)
(2 , 21) (2 , 22) (3 , 0) (3 , 1) (3 , 2)
(3 , 3) (3 , 4) (3 , 5) (3 , 6) (3 , 7)
(3 , 8) (3 , 9) (3 , 10) (3 , 11) (3 , 12)
(3 , 13) (3 , 14) (3 , 15) (3 , 16) (3 , 17)
(3 , 18) (3 , 19) (3 , 20) (3 , 21) (3 , 22)
(4 , 0) (4 , 1) (4 , 2) (4 , 3) (4 , 4)
(4 , 5) (4 , 6) (4 , 7) (4 , 8) (4 , 9)
(4 , 10) (4 , 11) (4 , 12) (4 , 13) (4 , 14)
(4 , 15) (4 , 16) (4 , 17) (4 , 18) (4 , 19) (4 , 20) (4 , 21) (4 , 22) (5 , 0) (5 , 1)
(5 , 2) (5 , 3) (5 , 4) (5 , 5) (5 , 7)
(5 , 8) (5 , 9) (5 , 10) (5 , 11) (5 , 12)
(5 , 13) (5 , 14) (5 , 15) (5 , 16) (5 , 18)
(5 , 19) (5 , 20) (5 , 21) (5 , 22)
Totally 135 values are obtained. Any one of the combination can be taken to generate elliptic curve.

4.1.2. Generation of Points

Each node can generate the elliptic curve points from the given information. For example take p=23, a=1,b=1.Points over the elliptic curve are,

(0,1) (0,22) (1,7) (1,16) (3,10) (3,13) (4,0) (5,4) (5,19) (6,4) (6,19) (7,11) (7,12) (9,7) (9,16) (11,3) (11,20) (12,4) (12,19) (13,7) (13,16) (17,3) (17,20) (18,3) (18,20) (19,5) (19,18).

4.1.3. Plotting the Points

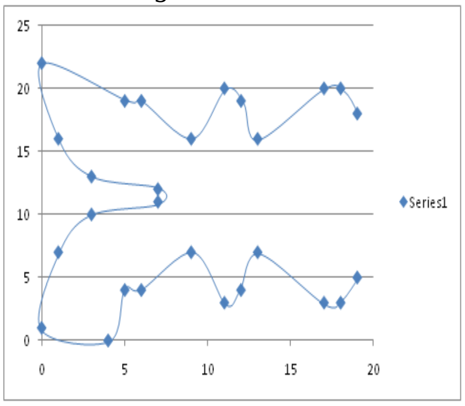


Figure 4.1 An Elliptic Curve From The Calculated Points

From figure 4.1 , points are symmetric about y=11.5.Shared Keys will be calculated using this points

4.1.4 Scalar Multiplication Performed Over The Elliptic Curve

Scalar multiplication is performed using repeated point addition on the curve.

POINT ADDITION:

- Take two points on an elliptic curve $P(x_1,y_1)$ and $Q(x_2,y_2)$
- $P + Q = R = (x_3, y_3)$ on the curve, where $x_3= (\lambda^2 - x_1 - x_2) \bmod p$, $y_3= (\lambda(x_1 - x_3) - y_1) \bmod p$

where

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & if\ P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & if\ P = Q \end{cases}$$

Example 1: Take P=Q=(3,10),

$x_1=3, y_1=10, x_2=3, y_2=10$

λ = $((3*3^2 + 1)/(2*10)) \bmod 23$

= (28/20) mod 23

=(5/20) mod 23

= (1/4)mod 23

= 6

x_3 = $((6)^2 - 3 - 3) \bmod 23$ =(30)mod 23

= 7

y_3 = (6(3-7)-10) mod 23 = -34 mod 23

= 12

This is also described as 2P=P+P=(3,10)+(3,10)=(7,12)

Example 2: Take P=(3,10) Q=(5,19) , $P+Q=(X_3,Y_3)$

λ = ((19-10)/(5-3)) mod p

= (9/2) mod 23

= (9 mod 23 * (1/2) mod 23)mod 23

= (9*12) mod 23

= 16

X_3 = $(16^2-3-5) \bmod 23 = 18$

Y_3 = (16(3-18)-10) mod 23

= -250 mod 23

= 3

4.2 Key Excahange

Take G=(0,22) , and n=27

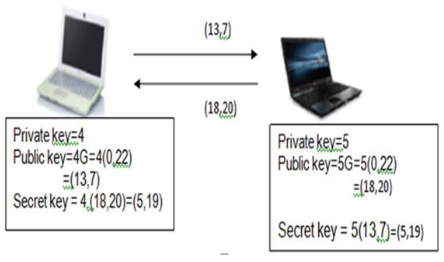


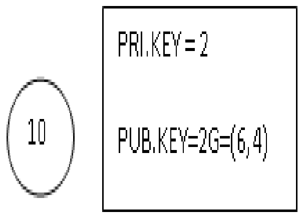
Figure 4.2 Key Exchange Using Ecdh

Figure 4.2 shows the way of public key computation over the elliptic curve and describes the steps for key exchange between two members.

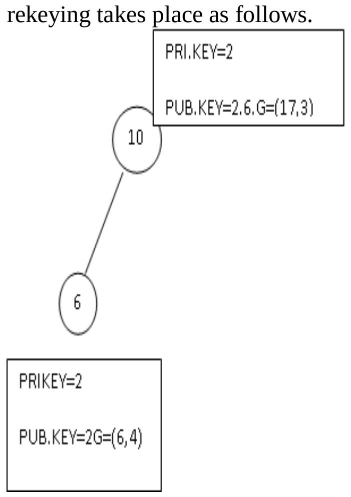
4.3 Group Key Exchange at Node Joining

Take G=(0 , 22)

1. A node with the value 10 starts the group. It holds its own private key and a public key.

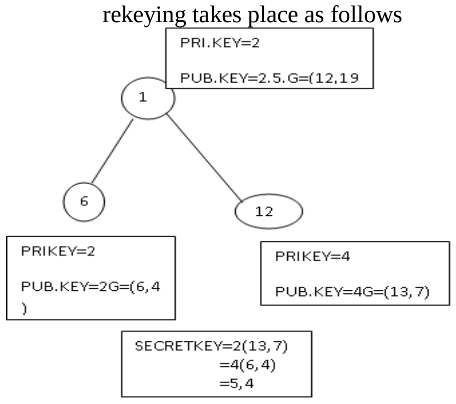


2. When node 6 joins, the resulted key tree and rekeying takes place as follows.



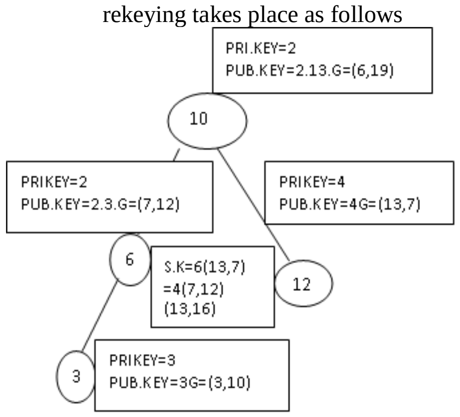
The value 17 is taken as group key and unicasted to node 6.

3. When node 12 joins, the resulted key tree and rekeying takes place as follows



The new group key is 12.

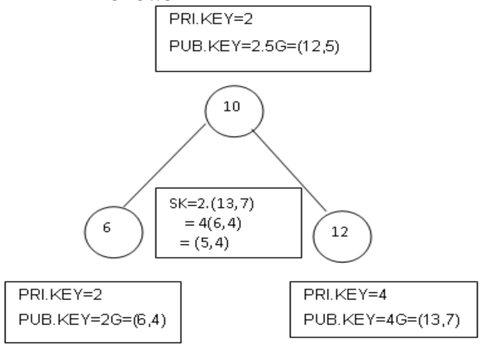
4. When node 3 joins, the resulted key tree and rekeying takes place as follows



The new group key is 6.Thus every time a node joins, rekeying is taking place. New group key is constructed and further group messages are encrypted using the new key

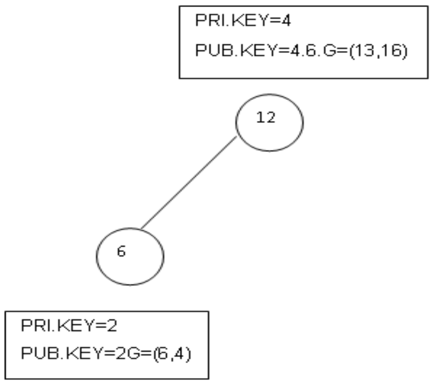
4.4 Group Key Exchange at Node Leaving

1. When a node 3 leaves from the group, the resulted key tree and rekeying takes place as follows



Group key value is change to 12.

2. When a node 10 leaves the group



The new group key value is 13.

Thus every time a node leaves, rekeying takes place. New group key is constructed and further group messages are encrypted using the new key

The proposed mechanism needs to recalculate the key value from the part of the joining (leaving) node sub tree without recalculating the entire tree, thus saving a tremendous amount of operational time

5. RESULTS AND ANALYSIS

The group key agreement protocol TGECDH is implemented using java language. The implementation performs the ECDH group key agreement module. When members join and leave the group dynamically, it constructs a height balanced tree and generates the new group key.

5.1. Comparison Of Ecdh With Other Key Agreement Protocol

ECC key with smaller bits can give same security levels that can be achieved using RSA or DH protocols with large bit size. Table 5.1 lists out the key length of symmetric keys, ECC,RSA,DH and security levels achieved using these technique , key size ratio between ECC and RSA

Table 5.1 Comparison Of ECC With RSA And DH Protocols

Security Level	Symmetric Key Length(Bits)	ECC Key Length(Bits)	RSA/DH Key Length(Bits)	ECC/RSA Key Size Ratio	MIPS Year Time To Break Key
2^{80}	80	160	1024	1/6	10^{12}
2^{112}	112	224	2048	1/9	10^{24}
2^{128}	128	256	3072	1/12	10^{28}
2^{192}	192	384	7680	1/20	10^{47}
2^{256}	256	512	15360	1/30	10^{66}

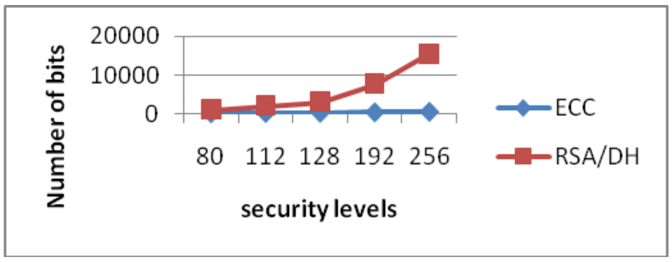


Figure 5.1 Number of Bits Needed To Achieve The Security Levels

5.3. Comparison of Existing Approaches with the Proposed System

This comparison is done based on the number of messages needed and the number of operations performed to compute the group key. Number of messages are analyzed to define the communication complexity .Number of operations define the computational complexity.

Table 5.2 Comparisons Of Existing Protocols With The Proposed Protocol

PROTOCOLS	MESSAGES	OPERATIONS
BD (Burmester and Desmedt Protocol)	2n	n(n-1)
GDH-3	2n-1	5n-6

TGDH	3	n(3h-1) (h is height of the tree)
PROPOSED TGECDH	n+2	2n-1

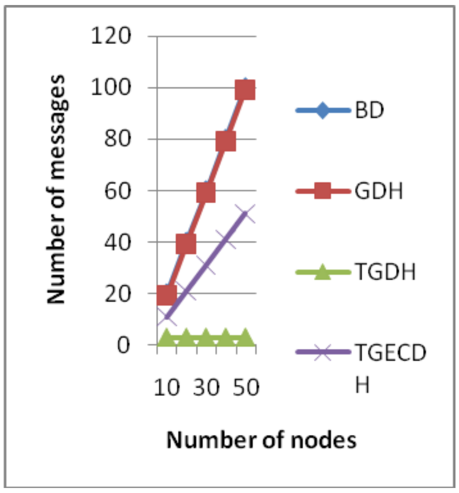


Figure 5.2. Number of Messages Need To Be Sent For Computing Group Key

X-axis denotes the number of members in the group and Y-axis gives the number of messages needed to compute the group key. Although the proposed TGECDH needs more number of rounds than TGDH, the message size is significantly lower than that of TGDH. In TGDH, at every time a node joins or leaves, the sponsor node has to send the updated key tree with blinded key. Thus it needs larger bandwidth to send this information. TGECDH does not need to send the whole tree. Instead of doing this, it just sends the ip addresses of the group members. The new member can construct a logical view of the tree from the received ip addresses.

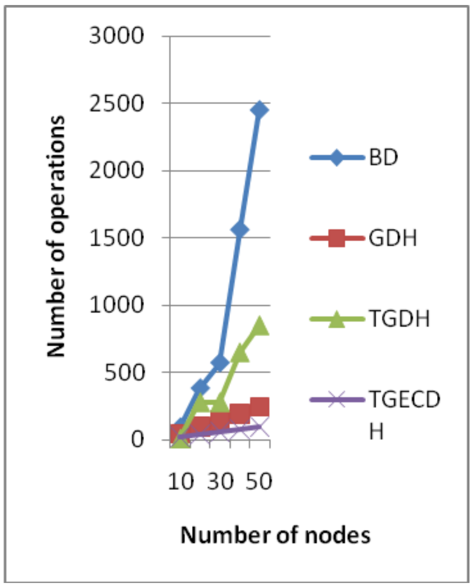


Figure 5.3. Number of Operations Need To Be Performed To Compute The Group Key

In TGDH approach each node has to store the key values of every node along the path from itself to the root node. Therefore it needs to remember (h+1) values throughout the group key communication. In the proposed TGECDH protocol each node needs to hold its private key, public key and its secret key. Hence the storage need for TGDH increases, as the number of nodes increases.

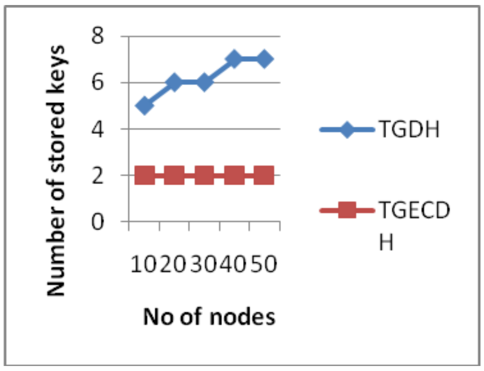


Figure 5.4 Total Numbers Of Keys Needed To Be Stored In Every Node

The proposed TGECDH protocol uses scalar multiplication, while all other techniques make use of exponentiation operation. By doing so, it considerably reduces the computation overhead. The total number of operations needed is also considerably less when compared to other approaches.

From the analysis given above, this protocol can be applied to the ad-hoc devices which are having low computing power and storage capabilities. By reducing the communication time, key convergence time is getting reduced. This is desirable at the environments where node joins and leaves the group dynamically because of its mobility.

6. CONCLUSION

This proposed scheme provides resilient mechanisms for dynamic group key management. The proposed scheme replaces exponential operations with point multiplications when performing ECDH, thus reducing the CPU overhead significantly. Therefore, the proposed scheme is highly promising for dynamic key operations in ad-hoc networks. It achieves the same security level as RSA and Diffe-Hellman do with shorter keys. Additionally, the proposed scheme performs very well for dynamic nodes joining or leaving.

7. FUTURE WORK

Though it performs well in dynamic environments, it should be applicable for large scale ad-hoc networks. The scalability should be considered. Future works should incorporate region based approaches to improve scalability of members. Group merge and partition protocols should be considered along with join and leave protocol.

8. REFERENCES

[1] Ingemar Ingemarsson, Donald T. Tang, and C. K. Wong,1982, “A conference key distribution system,” *IEEE Transactions on Information Theory*, vol. IT-28, no. 5, pp. 714–720.Mike Burmester and Yvo Desmedt,1994, “A secure and efficient conference key distribution system,” in *Advances in Cryptology – EUROCRYPT ’94*, pp. 275–286.

[2] Klaus Becker and Uta Wille,1998, “Communication complexity of group key distribution,” in *5th ACM*

conference on Computer and Communication Security.

[3] Michael Steiner, Gene Tsudik, and Michael Waidner,2000, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769– 780.

[4] Y. Kim, A. Perrig, and G. Tsudik, 2004, "Tree-based group key agreement," ACM *Transactions on Information and System Security*, vol. 7, no. 1, pp. 60–96.

[5] Hua-yi lin , Tzu-chiangchiang, 2011, "Efficient key agreements in dynamic multicast height balanced tree for secure multicast communications in ad hoc networks", *EURASIP journal on wireless communication and networking*. ArticleID 382701.

[6] S.Maria Celestin Vigila ,K. Muneeswaran,2011, "ECC based contributory group key computation scheme using one time pad", *Journal of computing,*volume 3,issue 6.

[7] William stallings,2005," cryptography and network security": *principles and practices*", fourth edition, prentice hall.

[8] Eric Ricardo Anton, Otto Carlos Muniz Bandeira Duarte,2002," Group Key Establishment in Wireless Ad Hoc Networks", Workshop em Qualidade de Serviço e Mobilidade, Brazil.