MULTICAST KEY AGREEMENT PROTOCOL FOR MOBILE AD HOC NETWORKS

André Boumso, Boucif Amar Bensaber, Ismail Biskri, Samir Khali
Department of mathematics and computer sciences, University of Quebec at Trois-Rivières, 3351 Bd des Forges C.P 500, Trois-Rivières, G9A 5H7 - QC - Canada

Keywords: Wireless network, Ad Hoc network, Multicast, Key Agreement, group activity, activity threshold.

Abstract: In this paper we address the problem of Multicast secure data over a multihop wireless ad hoc network. Many protocols that have been proposed are not really convenient for mobile ad hoc networks. We propose a key agreement protocol that aims to solve problems that are specific to ad hoc networks such as mobility, unreliable links, and multihop communication cost. The main idea is to focus on group dynamics and complete node mobility in ad hoc environment to develop an adaptive protocol that is suitable for the network and group changes. Doing so, we extend and adapt the proposed Tree based Group Diffie-Hellman (TGDH) protocol to pure mobile ad hoc network. Our method promotes the use of a fully balanced tree and eliminates the broadcasting of the entire tree as it appears in TGDH. We introduce a probabilistic value that gives the state of group dynamism. We simulated our protocol using C++ code and some results are presented in this paper and currently other simulations are going on the Network Simulator ns environment under various mobility, group size, and group dynamic scenarios.

1 INTRODUCTION

Multicast and mobile communications are emergent technologies that might enable interactive real time applications such video on demand and videoconference to name a few. In infrastructureless zones, ad hoc networks seem to be a natural extension of networks with fixed infrastructures. However, securing groups' communications in these networks is very challenging, since they suffer cruelly from a lack of resources and significant topology changes. Four principal research axes are presently considered for securing multicast communications: sender and recipient access control, authentication of the communicating entities, key management and fingerprinting (Paul Judge, Mostafa Ammar, 2003).

The purpose of key management is to optimize the generation of keys, their distribution and their maintenance. Unfortunately, the present key management methods are not optimal.

The aim of our work is to develop a scalable key management approach that introduces the notion of group activity threshold allowing the choice of key tree management and distribution model adaptable to groups' communications. This method will also try to reduce the encryption time, the number of keys transmitted or stored, and the quantity of bandwidth used.

Our paper arises as follows: first of all, we present a brief review of the literature on keys management; second, we propose our solution with the whole necessary architecture to its modeling. Then we detail encoding keys management through the various operations of group management.

2 RELATED WORKS

Many Key Agreement protocols have been presented in the literature. One of them is the cliques suite, a variety of protocols that extend the Diffie-Hellman two party protocol to groups (Michael Steiner, Gene Tsudik, Michael Waidner, 1998). In IKA.1 (Initial Key Agreement protocol 1), the last member who joins the group plays the role of group controller. The key agreement is a linear process and every member M_i, $i \in [1,n-1]$, contributes its own share in a round i upflow message. The last round n is a broadcast of data collected from the previous n-1

Boumso A., Amar Bensaber B., Biskri I. and Khali S. (2006).
MULTICAST KEY AGREEMENT PROTOCOL FOR MOBILE AD HOC NETWORKS.
In *Proceedings of the International Conference on Wireless Information Networks and Systems*, pages 86-94

rounds. In this protocol the number of rounds increases linearly with joining events, the last member joining the group (the controller) may be a single point of failure. The order in which packets are transmitted must be properly defined. Nevertheless, these protocols have the advantage of consuming less bandwidth and managing partition events efficiently.

Let g be a generator of a cyclic finite group G of order q and S_i a random contribution of member M_i, IKA.1 round i appears as follow:

M_i $\rightarrow$ M_{i+1}

$\{g^{S1*S2***Si/S_k} / k \in [1,i]\}, g^{S_1 * S_2 *** S_i}$

Figure 1: IKA.1 round i; Mi member sending a set of collected values in an upflow message to Mi+1, i ∈ [1,n-1].

Another group DH extension protocol is Octopus. It's extending the 2nd-hypercube protocol (K.Becker, U. Wille, 1998) to unlimited number of members. In Octopus, n members are divided in five subgroups. Four members A,B,C and D become controllers of four subgroups G_A,G_B,G_C,G_D and the n-4 remaining members are distributed among this subgroups in a rectangular way. Every controller collects each subgroup member contribution k_i in a 2-party Diffie-Hellman key exchange, computes a subgroup key and exchanges this key in a 4-party Diffie-Hellman with its neighbouring controllers.

Let g be a generator of a cyclic finite group G of order q, A, B, C and D the four subgroups controllers, k_a, k_b, kc, k_d their respective subgroups keys. The protocol operates as shown below:

Step 1: A and B, C and D exchanges their keys and compute the first round keys $k_1 = g^{k_a * k_b}$ and $k_2 = g^{k_c * k_d}$

Step 2: A and C, B and D exchanges the keys obtained in step1 and compute the group key $k = g^{(k_1 * k_2)}$

Figure 2: Octopus four party key agreement protocol steps.

The third contribution is the Tree-Group Diffie-Hellman (TGDH) proposed by Yondae Kim, Adriane Perrig and Gene Tsudik. TGDH is a contributive tree key management protocol which comes to optimize the performances of the IKA1/2 (Michael Steiner, Gene Tsudik, Michael Waidner, 1998). It is similar to OFT (One Way Function Trees) (D. Balenson, D. McGrew and A. Sherman, 2000) but each member can act as sponsor depending on his position in the key tree. The sponsor is responsible for the computation and the broadcasting of intermediate node keys to other members of the group. It is the rightmost member of the subtree or the rightmost member of the deepest subtree associated with the incoming or the leaving node. In the partitioning event, they may be several sponsors in the process of building a new group key.

When a membership event occurs, the sponsor changes his own share, computes keys on his keys path and blinded keys on his co-path from the leaves to the root and broadcast blinded key tree. All members update the key tree and compute the group key.

TGDH is not suitable for mobile ad hoc networks since the mobility of nodes might make it impossible for a simple node to broadcast a message to all members. Therefore, to operate properly, the network must be restricted so that nodes stay relatively close to each other throughout the entire multicast session so for instance, the bandwidth problem is resolved (Maria Striki, John S. Baras, 2003). The mobility of node can cause frequent link breakage, involving multiple partitioning events that may lead to a deeply unbalanced tree. Modular exponentiation is the most expensive operation in this protocol and depends on the key structure. In a deep unbalanced key tree, if a deepest node leaves, it might require O(n) exponentiations to compute the group key. Also, the broadcasting of the whole tree in membership events seems to be unnecessary and the protocol uses a lot of bandwidth notably in partitioning events. So, maintaining a well balanced tree almost in high dynamic environments such as ad hoc networks might maintain the computation cost to O(log(n)). The authors of TGDH did not describe the initialization phase.

Let M_s be the sponsor, $G^* = \{M_1..M_n\} - \{M_L\}$ the group of remaining members in a leaving event and T(BK*) the new blinded tree, the protocol operate as follow:

M_s T(BK*) G*

Figure 3: TGDH: Sponsor broadcasting new blinded key tree in leaving event.

3 GAKAP METHOD

Our method GAKAP, witch stand for **Group Activity based Key Agreement Protocol**, is based on TGDH. However, it promotes the use of a fully balanced tree, eliminates the broadcasting of the entire tree as it appears in TGDH, and uses the elliptic curve cryptography algorithm ECDH for key exchange instead of the traditional Diffie-Hellman (DH) as in TGDH. The term fully means that the key tree is always entirely balanced in a high dynamic environment. It introduces the concepts of group activity and activity threshold. Group activity is a probabilistic value that gives the state of group dynamism in:

- predominant additive state with high nodes mobility or note
- predominant leaving state with high nodes mobility or note
- equilibrium state which gives the value of the activity threshold suitable for the key management in a high dynamic ad hoc network.

Keys are identified by their name, according to their positions in the key tree. The tree key management depends on the group activity threshold and keys are shifted instead of being deleted (M. Steiner, G. Tsudik, M. Waidner, 1996) when a membership event occurs. The concept of Group Activity includes group's dynamics and node mobility in both the group and the entire network.
We will explain thereafter the encoding keys management through the various operations of group management.

4 GROUP MEMBERSHIP EVENTS

In high dynamic groups and ad hoc networks, joining events as well as leaving, merging and partitioning can arrive very frequently. Therefore, the keys management may become very laborious in term of computation and bandwidth consumption. The protocol that follows tends to minimize these effects.

4.1 Initialisation

In this phase, the initial key tree is built and the initial group key is computed. The protocol operates as follow:

Step 1: the initiator (or initial controller) publishes the opening of a multicast session, upon receiving the joining response in a predefined period of time, it builds up the list of participants, builds the tree containing the blinded keys and their names (or ID) and broadcasts them to the members.

Step 2: round i: almost $n/2^i$ (with n the number of initial members and $i \in [1,h]$ where $h=\log_2 n$) of the group members become the sponsors, compute the keys and blinded keys on their path and co-paths from the leaves up to the root and broadcast the blinded keys to the group.

Step 3: round h+1: Each member can compute the group key and the initial group activity probability.

Figure 4: GAKAP: Initialization protocol phase.

At the end of the initialization phase, all members have completed the same key tree and computed the same group key. With the key naming introduced in this phase, members can determine the order of parenthood that exists among them. This parenthood will permit later to "elect" the sponsor.

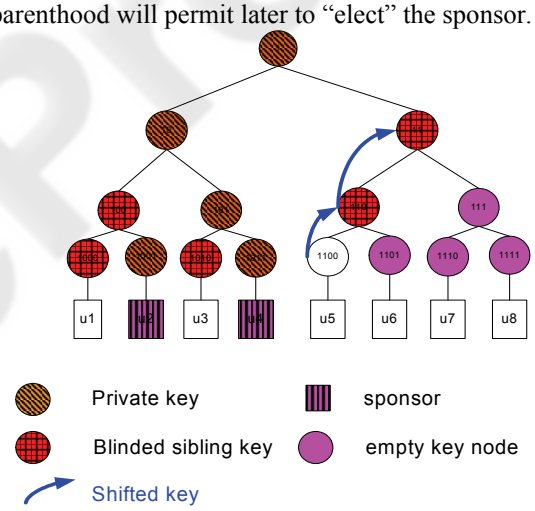

Figure 5: Initialization key tree phase.

4.2 Joining Event

The new member broadcasts a join request containing its blinded key (bkey) BK. Each current member determines the insertion point which is the first empty leaf node when traversing the key tree from left to rigth. The sibling node of the new member if it exists, becomes the sponsor, otherwise

one of its cousin node (Figure 7) becomes the sponsor.

Step1: The joining member M_{n+1} broadcast the join request containg its bkey BK_{n+1} to the group.

M_{n+1} BK_{n+1} $\{M_1 \ldots M_n\}$ →

Step2:

- Each member:
  - Updates its key tree by erasing all bkeys to be changed on the sponsor path;
  - Determines the insertion point.
- The sponsor M_s changes its share and broadcasts a set of all modified bkeys BK^* and sends the bkey tree $T\{BK^*\}$ to the new member.

$\{M_1 \ldots M_n\}$ $\{BK^*_s\}$ M_s ←

M_{n+1} $T\{BK^*_s\}$ M_s ⇠

Step3: Every member can compute the group key using the necessary keys.

Figure 6: GAKAP joining event protocol.

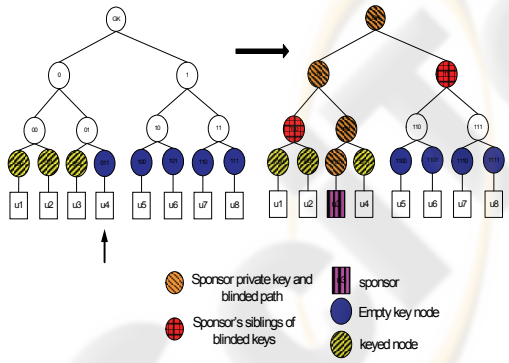

Figure 7: Key tree in joining event.

The sponsor updates its key tree by computing all the keys and blinded keys on his path to the root and broadcasts the blinded modified keys to the other members and sends the tree of blinded keys in unicast mode to the new member. Every member determines the insertion point, computes both the new key and the activity threshold. If, after the addition of a new member, all the node of the tree are completed, a new tree must be build as follow:

- A new group key node is constructed on the top of the tree;
- The previous key tree becomes the left sub-tree of the new tree;
- The right sub-tree is built as full as possible depending on the activity threshold.

Upon receiving the blinded key tree from the sponsor, the new member computes the missing keys of all the parents on his way to the root.

4.3 Leaving Event

The member broadcasts a leaving request containing its ID. The sponsor generates a new contribution, updates his key tree by erasing all the keys and blinded keys on its path to the root and broadcast the blinded modified keys to all the other members. Every member updates the key tree by determining and "deleting" the keys to be changed, and replacing them by those contained in the list received from the sponsor.

Step1: The outgoing member M_L broadcasts its leaving request containing its ID to the group.

M_L ID_L $G = \{M_1..M\}$ →

Step2: Every member

Updates its key tree by erasing the leaving member's key and all the blinded keys on the sponsor path.

- The sponsor additionally erases all keys on
- It changes his share, computes new keys and blinded keys on his path and broadcasts the bkeys.

$G^* = \{M_1..M_n\} - \{M_L\}$ $\{BK^*s_i\}$ M_s ←

Step3: Every member can compute the group key using the appropriate bkeys.

Figure 8: GAKAP leaving event protocol.

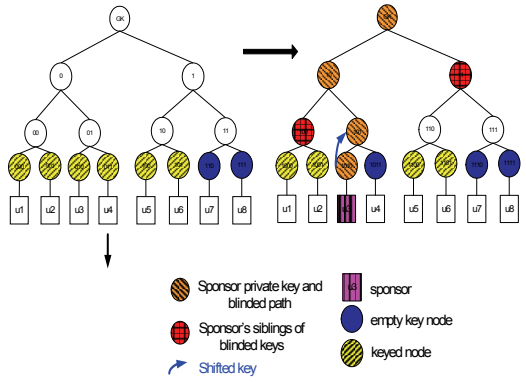

Figure 9: Key identification and key management in departing events.

4.4 Multiple Join

As in the simple join event, the joining members send their joining request at the same time. Since all joining requests are not received in the same order by all members, it is necessary to classify them in order to guarantee the same key structure. A temporary controller is then elected to this purpose. The temporary controller or group temporary sponsor is the rightmost member concerned by the additive event: this means the member in the righmost subtree concerned by the additive event. The temporary controller determines the various insertion points, inserts each new participant in an empty node.

It chooses a new contribution, computes its new share and all keys and blinded keys (bkeys) on his path to the root, and broadcasts both the tree containing the bkeys and the activity threshold of the moment to all participants. Every current member updates its key tree by erasing all keys and bkeys to be changed. New members received the key tree structure. Each sponsor broadcasts modified blinded keys to all other members. Every member replaces all modified bkeys by those received from the sponsors. Each participant can then compute the group key. The new key tree may be constructed as indicated in the simple additive event if at some moments in the process the tree is full. If two new members are leaves of a previous intermediate empty parent node, one of them, the leftmost becomes the sponsor.

In this method, we distinguish the multiple additive events from the merging event since the later suppose the addition of two different previous subgroups in the aim of forming a new group.

Let p be the number of the new members.

Step1: Every new joining member Mj broadcasts its joining request containing its BKj to the group

Mj BKj $G=\{M_1..M_n\}$

Step2: Each member:

- Updates its key tree by erasing all bkeys to be changed on the sponsor path.

Step3:

Round 1 to h:

- A temporary group controller M_{tc} computes its new share, builds a new bkey tree with all members' bkeys and broadcasts the tree to the group.

M_{tc} $\{M_1..M_n\}U\{M_k\}$

For $k \in [n+1,n+p]$:

- Every sponsor M_{si} computes keys and bkeys on his path and broadcasts a set of all modified bkeys BK^*_{SI} .to the group.

$G^*= \{M_1..M_n\}-\{M_{Li}\}$ $\{BK^*s_i\}$ M_{si}

Step4: Every member can compute the group key using the appropriate bkeys.

Figure 10: Multiple join protocol in GAKAP.

4.5 Multiple Leave

As in the leaving event, many members leave the group at the same time by sending individual leaving events. Each sponsor identifies all the participants leaving. It changes its contribution and computes secretes and blinded keys of all the parents' nodes on his way to the root. It broadcasts his parents' blinded keys. It replaces all others modified blinded keys. It computes the group key. Each member identifies the leaving participants. It erases the leaving nodes keys and the parents' nodes if it is necessary. It updates the tree by replacing all the blinded keys that have been modified by the sponsors. It computes the group key.

Let p be the number of leaving members.
Step1: Every outgoing member M_{Li} broadcasts its leaving request containing its ID to the group

M_{Li} ID_{Li} $G=\{M_1..M_n\}$ →

Step2: Every member:

- Updates its key tree by erasing the leaving member's key and all the blinded keys on the sponsor path.
- The sponsor additionally erases all keys on his path up to the root..

Step3:
Round 1 to h:

- A temporary group controller M_{tc} computes its new share, builds a new bkey tree with all members' bkeys and broadcasts the tree to the group.

M_{tc} $\{M_1..M_n\}U\{M_k\}$ →

For $k \in [n,n-q]$:

- Every sponsor M_{si} computes keys and bkeys on his path and broadcasts the bkeys.

$G^*= \{M_1..M_n\}-\{M_{Li}\}$ $\{BK^*s_i\}$ M_{si} ←

Step4: Every member can compute the group key using the appropriate bkeys.

Figure 11: Multiple leave protocol in GAKAP.

4.6 Merge Event

Each sponsor chooses a new contribution and computes all secrete and blinded keys on his way up to the root. It broadcasts his subtree containing the blinded keys. Upon receiving other subtree, he builds a new tree in which the deepest one becomes the left subtree. Depending on the activity threshold, it builds a fully balance tree by completing the right sub tree and computes the group key.

Each member determines the insertion point and depending of the activity threshold, it builds the new key tree. It finally computes the group key.

4.7 Sponsor Election

The election of the sponsor in a membership event is based on its parenthood closeness to the incoming or the leaving member. If the incoming or leaving node has a sibling, it becomes the sponsor; otherwise one of the nodes with which its share a common parent key in the smallest highest sub-tree becomes the sponsor.

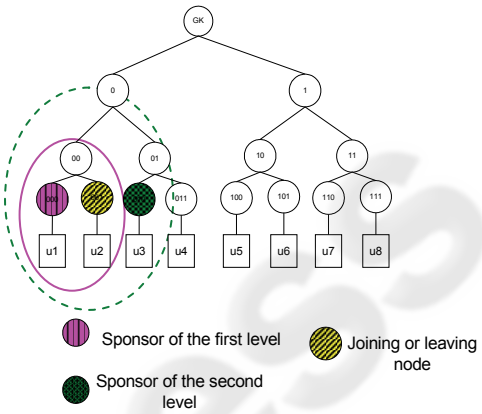

Figure 12: Order in which sponsor is chosen.

5 RESULTS

In this section, we introduce the simulation results. Simulations were done using C++ code and compare GAKAP to TGDH.

To study the behaviour of GAKAP and TGDH, we have implemented a C/C++ code (in visual studio and Kdevelop environments) of the tree structure behaviour when group events occur. Group events where generated randomly. Link breakage and connexion due to nodes mobility were generated randomly. We first use group that the number of participants vary from 25 to 250 and the number of group events from 10 to 100. After, we maintain a group of 250 participants. We calculated the number of messages transmitted by the sponsors, the number of rounds done when computing group key. The results we obtained could be interpreted as follow:

1. for both methods, the number of packets increases with the number of rounds;

2. the number of rounds grow with the number of events;

3. to maintain the equilibrium of the TGDH key tree in high dynamic MANET, they must be 1.8 to 1.9 joining member for 1 leaving member (Figure 19). This is the value of the activity threshold for which, in maintaining a constant flow of additive and leaving events, we can maintain a fully balanced tree, thus the group key computation order of O(logN).

5.1 Packets Exchanged

The packets sends by sponsors are of two types: Unicast and Multicast. Unicast packets are only sent in simple join events. The number of packets where count in simulation.

We can see that the number of total packets sends by sponsors in group events is higher in GAKAP than in TGDH, this is because sponsors delivered two packets: one multicast to the group and a unicast packet to the joining member (Figure 13). However, the size of packets sends in GAKAP protocol are fewer compare to that of TGDH, and the size of TGDH packets increases significantly with number of events (Figure 14).

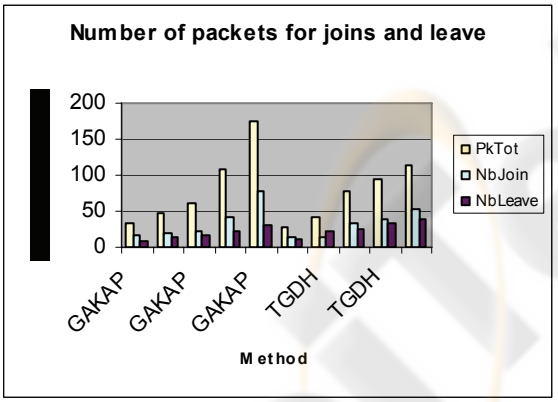

Figure 13: Number of total packets sends for joining and leaving events.

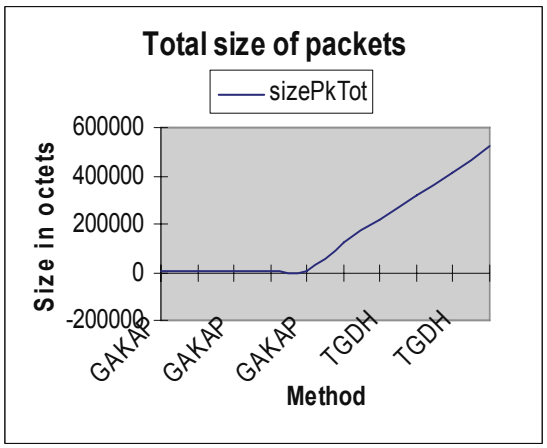

Figure 14: Size of packets sends for all group events.

5.2 Number of Rounds and Computation Cost

In this section, we find the number of rounds necessary to compute group key in join and leave events. Figure 15 shows us the number of rounds for simple joins and multiple joins events. Figure 17 gives us the number of rounds for simple leave and multiple leave events. Since the number of simple and multiple joins are taken as joins events and simple and multiple leave as leave events, figure 16 and 18 give more precise value of the percentage of the number of rounds for joining and leaving events.

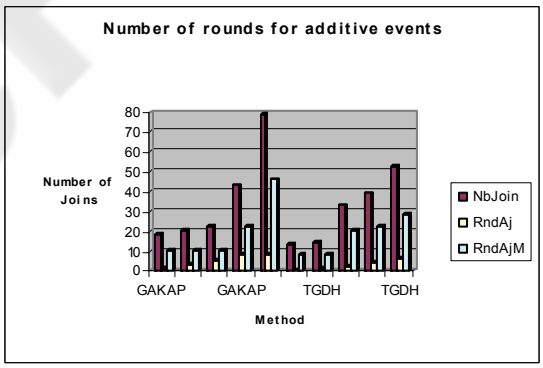

Figure 15: Number of rounds for joining events.

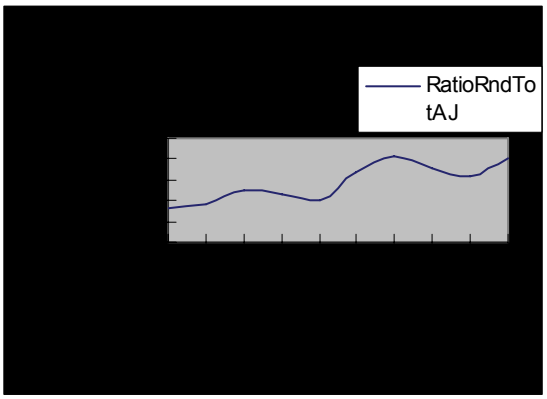

Figure 16: Ratio for Number of rounds in joining events.

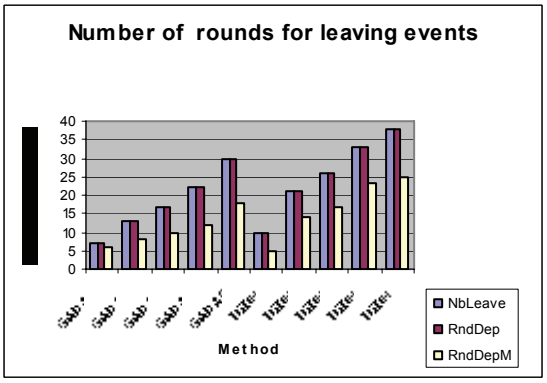

Figure 17: Number of rounds for leaving events.

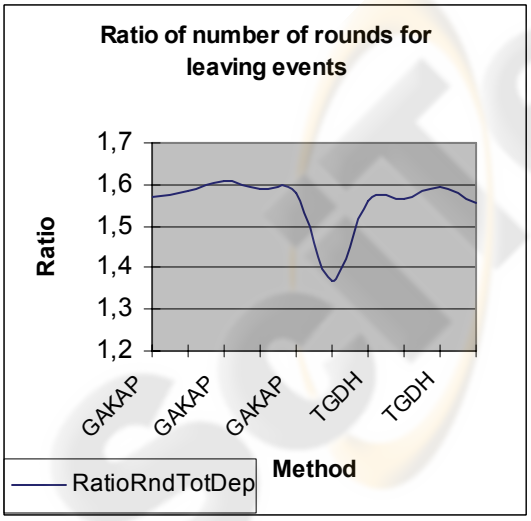

Figure 18: Ratio for number of rounds in leaving events.

We can conclude from figure 16 and 18 that the number of rounds in group key computation for joining events is less in our method, while it is a little bit greater in GAKAP in group key computation for leaving events.

The computation cost of the group key increases with the number of rounds and the size of tree node keys used for group key computation. Since the size of keys (123 bits) used in our method is less than 1024 bits used in TGDH, the computation cost of the group key is less in our method.

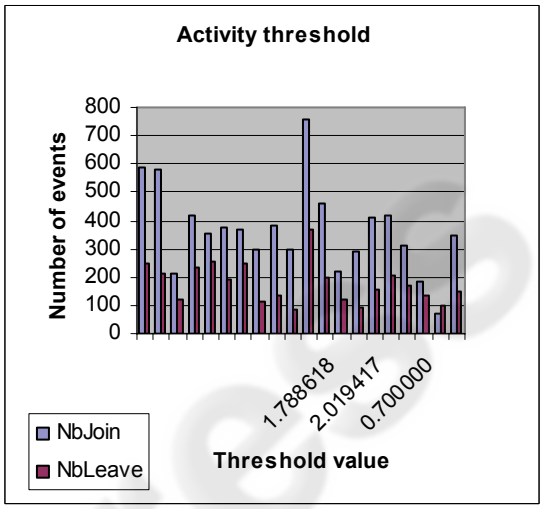

Figure 19: Activity threshold value.

6 CONCLUSION

We presented some researches trends for key agreement techniques. The aim of this proposal was to bring another point of view in key distribution for group communication in MANET. Most of the solutions proposed by now are limited to a MANET with restricted node mobility. Our method bring an additional element since it is apply to fast moving nodes and high dynamic multicast groups.

In an environment in which fast mobility and nodes dynamics are very concerned like mobile ad hoc networks, GAKAP provide efficient key agreement and tree management. It tends to reduce better the number of transmissions, the computation cost of both group keys and members keys. The activity threshold is the better mobility and group dynamics index for key tree management in fast changing topology multi-hop wireless ad hoc networks.

More simulations are currently under testing on NS simulator and the aim of these simulations is to confirm the results describe above, obtained using the C/C++ code.

REFERENCES

Paul Judge, Mostafa Ammar. "Security Issues and Solutions in Multicast Content Distribution: A survey". IEEE Network, January/February 2003.

Maria Striki, John S.Baras. "Key Distribution Protocols for Multicast Group Communication in MANETs", Technical Report, CSHCN, October 2003.

M.Steiner, G. Tsudik, M.Waidner. "Diffie-Hellman Key Distribution Extended to Group Communication". In Proceedings of the 3rd ACM conference on Computer and communications security, January 1996.

D. Wallner, E. Harder and R. Agee, "Key Management for Multicast: Issues and Architectures", Internet Engineering Task Force, no. 2627, June 1999.

K.Becker, U. Wille. "Communication complexity of group key distribution". In Proceedings of the 5th ACM conference on Computer and communications security, Novembre1998.

D. Balenson, D. McGrew and A. Sherman, "Key Management for Large Dynamic Groups: One-way Function Trees and Amortized Initialization" <draft-balenson-groupkeymgmt-oft-00.txt>, IETF, Sep.2000.

Michael Steiner, Gene Tsudik, Michael Waidner. CLIQUES: “a new approach to group key agreement”. In proceeding of the 18th international conference on distributed computing systems (ICDS’98), pp.380-387, Mai 1998

Yondae Kim, Adriane Perrig, Gene Tsudik.“Simple and fault-tolerant key agreement for dynamic collaborative groups”. In ACM conference on Computer and Communication security,pp.235-244, 2000.

Danilo Bruschi, Emilia Rosti. ``Secure Multicast in Wireless Networks of Mobile Hosts: protocols and issues``, Kluwer Academic Plublishers, 2002/pp.503-511,mobile ad hoc network and application, vol.7,no6,dec.2002.

A.Selcuk, C. McCubbin and D. Sidhu, "Probabilistic Optimization of LKH-based Multicast Key Distribution Schemes," Internet Engineering Task Force, Jan. 2000.