An Enhanced Hybrid Key Management Protocol for Secure Multicast in Ad Hoc Networks

Mohamed Salah Bouassida[1], Isabelle Chrisment[1], and Olivier Festor[2]

[1] LORIA-UHP
[2] LORIA-INRIA
name.surname@loria.fr
MADYNES, Campus scientifique
B.P. 239, 54506 Vandœuvre-lès-Nancy Cedex - France
tel : +33-3-83-59-30-49 - fax : +33-3-83-41-30-79

Abstract. An ad hoc network is a collection of wireless mobile hosts forming a temporary network without the aid of any established infrastructure or centralized administration. This flexibility in space and time induces new challenges towards the security infrastructure needed to support secure unicast and multicast communications. Especially, traditional group key management architectures meant for wired networks are not appropriate in such environment due to high dynamics and mobility of nodes. In this paper, we propose an enhanced hybrid key management protocol for secure multicast dedicated to operate in ad hoc networks. Built on a protocol called BAAL dedicated to key distribution in wired networks, our approach integrates threshold cryptography and the services of the AKMP protocol to deliver fast, efficient and mobility aware key distribution in a multicast service.

Key words: ad hoc networks, multicast security, group key management, threshold cryptography

1 Introduction and motivation

The last decade saw the exponential deployment of wireless networks thanks to the emergence of new technologies and standards (e.g. the 802.11 series, Hiperlan [7], ...). The wireless networks can be used either in a base or in an ad hoc mode where hosts do not rely on any fixed infrastructure. The combined use of wireless ad hoc networks with wired gateways enables an easy network coverage extension at low cost. These networks also called hybrid networks are gaining more and more interest by the community including operators which see the clear advantages in deployment of these technologies. Those networks which are dynamic in space and time, offer great flexibility. However, this flexibility associated with the wireless connection vulnerability, requires an increased need in securing users and data.

The characteristics of ad hoc networks pose challenges in achieving the main security goals : authentication, confidentiality, non repudiation, integrity and availability. Using wireless links makes an ad hoc network vulnerable to link attacks ranging from passive eavesdropping to active impersonation, message replay and message distortion. Eavesdropping might give an adversary access to secret information, violating confidentiality. Active attacks might allow the adversary to delete messages, to inject erroneous messages, to modify messages and to impersonate a node, thus violating availability, integrity, authentication and non repudiation. The nodes in an ad hoc network are heterogenous and may have relatively poor physical protection, so they can be compromised. Therefore, we should not only consider malicious attacks from outside a network, but also take into account the attacks launched from within the network by compromised nodes. A major constraint placed on candidate security architectures in ad hoc networks is the absence of any possibility to put a centralized component in the solution. Security mechanisms must also be dynamic and efficient to adapt themselves to the dynamic and scalable nature of ad hoc network.

In parallel with the deployment of wireless ad hoc networks, multicast services gained acceptance through several applications like software distribution, multimedia conferencing, radio casting. The combination of an ad hoc infrastructure with multicast services which have to be operated induces new challenges towards the security infrastructure needed to enable acceptance and wide deployement of these multicast services.

In this article, we propose an approach to enable secure group communication within an ad hoc environment. Our solution focuses on group key management which is the major issue in group security. The principle of our approach is not to develop another new and specific solution but to adapt a protocol we have already tested and validated within wired networks.

Thus, our approach combines the functional architecture of BAAL [5], which is an group key management protocol in wired networks, with the dynamic support of AKMP [3]. And for ensuring secure generation of group keys, we use the threshold cryptography.

To present our solution, this paper is organized as follows. Section 2 identifies the multicast security challenges that emerge in ad hoc environment. In Section 3, we present related works concerning multicast security. Then, before describing our enhanced hybrid key management protocol for secure multicast in ad hoc networks, we give the main building blocks we have reused. Finally, we present the initial simulations and results obtained. Finally, we summarize the contribution and identify the directions for future work.

2 Multicast security challenges in an ad hoc environment

Securing multicast communications is challenging because they present more opportunities for traffic interception. In addition, the identity and the addresses of multicast groups are known in a large scale, which help adversaries to orient theirs attacks. Multicast routing information can also be attacked, which can prevent a node from knowing the exact way to join an group (e.g. : wormhole problem).

Several other challenges raised by ad hoc network features. These challenges are :

- The lack of infrastructure that implies there is no central authority to be referenced for trust decisions about other parties within the network. The transient relationships do not help in building trust based on direct reciprocity and incite some malicious nodes to cheat.
- The size and dynamicity of multicast group which can be very high in ad hoc networks : the size cannot be controllable and, in the same way, dynamicity of members addition or removal.
- The mobility of nodes which has to be taken into account in security architecture. Some members move and still want to be able to receive multicast data. Nodes can also disappear without leaving the group (battery problem,...). Thus, when they appear again, they want to receive multicast flow. In this case, the multicast tree changes frequently but not the group members.
- The scalability, in the context of group communications in an ad hoc network, refers to the capacity of security mechanisms to cover great size multicast groups, without affecting the performances of the whole security system. The problem concerns the group key and security policies management and distribution.
- The trust model in an environment without fixed infrastructure concerns entities generating, distributing and managing cryptographic keys and security policies. Thus, we need one trust model to answer to the following questions : to which entities trust is granted for ensuring security services, which level of trust must be granted to them and which is the authority alive source.

3 Related work

3.1 Secure Multicast Communication and State of the Art

IP multicast is an efficient communication mechanism for group-oriented applications, such as video conferencing, interactive group games and video on demand. IP multicast saves bandwidth by sending the source traffic on a multicast tree that spans all the members of the group. Group communication confidentiality requires that only valid users can decrypt the multicast data even if the data is broadcasted to the entire network. We assume that multicast data is encrypted using a symmetric cryptosystem ; the same key called Traffic Encryption Key (TEK) is used to encrypt and decrypt data.

The confidentiality requirements can be mainly translated into two key distribution rules :

- Forward confidentiality : users that left the group should not have access to any future key. This ensures that a member can not decrypt data after leaving the group.
- Backward confidentiality : a new user should not have access to any old key. This ensures that a member can not decrypt data sent before joining the group.

In order to meet the above requirements, a re-key process should be triggered after each Join/Leave procedure. It consists in generating a new TEK and distributing it to the members including the new one in case of a joining, or the residual members in case of a removal. We classify group key management proposals into three approaches:

- Approach A : all group members share a unique single symmetric key (TEK). This approach is mainly used within a centralized architecture where a single key server is responsible for generating and redistributing the new TEK whenever a member joins or leaves the group. This approach does not meet the scalability requirements since the number of transmitted messages to update TEK is proportional to n, the number of group members. This is known as the "1 affects n" phenomenon [12] where a single group membership changes results in a re-keying process that disturbs all group members to update TEK. In addition, the use of a single key server leads to a bottleneck problem during TEK distribution and suffers from a single point of failure. The BAAL protocol [5] belongs to this approach.
- Approach B : the multicast group is divided into multiple subgroups. Each subgroup shares a local TEK managed by a special entity : the subgroup controller. Protocols proposed within this approach are more scalable than centralized protocols, they also attenuate the "1 affects n" phenomenon. However, the drawback of this approach is that subgroups have different TEKs, multicast packets should be decrypted and re-encrypted by subgroup controllers whenever they pass from a subgroup to another. IOLUS [12] and AMAM [14] protocols belong to this approach.
- Approach C : To solve the "1 affects n" problem, without generating a great overhead due to the encryption/decryption process, this approach consists on merging the two preceeding approaches. The basic idea is to start a multicast session with centralized key management (approach A), and to divide the network dynamically in order to delegate key management to local controllers (approach B). AKMP [3](An Adaptive Key Management Protocol for secure Multicast) is proposed within this hybrid approach.

3.2 Secure Multicast Communication and ad Hoc environment

Few research results were published so far on how to secure multicast comunications within an ad hoc network.

The proposal of [15] defines the NTDR ad Hoc networks (Near Term Digital Radio). In NTDR architecture, there is a set of clusters, each containing a clusterhead, which when linked together form a routing backbone. A cluster has a single level consisting of nodes within one hop of a clusterhead. Intercluster communication is restricted to clusterheads only and intracluster communication between nodes that are within one hop of each other must traverse the clusterhead. This cluster based control structure promotes more efficient use of resources in controlling large dynamic networks, but generates more computing overhead due to the network clustering, the clusterheads election, and the establishment of the routing backbone.

For ensuring authentication, [15] makes use of public key systems and protocols and involves the use of certificates and certification authorities. Thus, all network participants have some time access to a public key infrastructure in their own fixed network domains. In this sense, [15] assumes that some hierarchical based PKI in fixed networks to which the participants have access to sometime before getting involved in a mobile ad hoc network. However, having an established PKI within an ad hoc environment suffering from lack of infrastructure is a very big challenge.

For ensuring secure group key management, [15] assumes that the clusterhead in an NTDR network could serve as a trusted entity to coordinate packet routing and manage security for the cluster. Thus, each clusterhead manages the cluster keys for its cluster and mediates all communication between its cluster and other clusters. Key generation and distribution is ensured by having two types of keys : cluster group key which is used to encrypt all cluster traffic to secure intra-cluster communication, and key encryption key which is a shared secret between a clusterhead and a node. This group key management is vulnerable because it is established around clusterheads which can be compromised. A computing overhead is also generated due to clusterheads movements, clusterheads deletion,...

[4] defines an authentication framework for hierarchical ad hoc Sensor networks. In this proposal, the sensor network consists of three tiers of devices with varying levels of computational and communication capabilities. This architecture is more scalable than flat ad hoc networks. The lowest tier consists of compute-constrained sensors that are unable to perform public key cryptography. Thus, [4] presents a new type of

certificate, called a TESLA certificate. This certificate can be used by low-powered nodes to perform entity authentication. TESLA is more detailled in [8, 2]. This three-tier architecture consists of three classes of wireless devices : high power access points that route packets received via radio links to the wired infrastructure, mobile medium-powered forwarding nodes that relay information from sensor nodes to access points, and low-powered mobile sensor nodes. [4] assumes that each forwarding node and access point has an RSA-key-pair along with its certificate. Like in [15], this assumption poses a very big challenge.

The proposal from [6] presents a new secure multicast communication approach using the measures from GPS units (latitude, longitude and altitude) and the Prüfer decoding algorithm. The group key is managed by using the Prüfer number and the Group Diffie Hellman key-exchange protocol (GDH). Any user in the multicast group can use the group key distributed by the source to securely receive multicast messages from the multicast source. This group key distribution model is efficient and robust but suffers from "1 affects n" phenomenon, and does not treat mobility factor. Computing Prüfer algorithm can also generate an overhead within ad hoc network nodes which have a relatively low power computing.

[10] proposes to reduce the communication and computation load on the source by having active group members which participate to the group security. This approach is based on IOLUS. The reliability is improved by allowing a node to maintain more than one link and the security is increased by requiring a joining node to authenticate with at least k members of the group. On the other hand, this approach generates a computing overhead due to the encryption/decryption process of IOLUS.

4 The Building Blocks

The aim of our approach is to solve the problems described in section 3.2 and to reduce the "1 affects n" phenomenon while limiting the computing overhead due to encryption-decryption process. For ensuring authentication within our network, we use threshold cryptography instead establishing a PKI infrastructure over an ad hoc environment. Finally, our approach treats mobility factor which is a big challenge to secure ad hoc networks.
In this section, we describe how we start from an existing group key management protocol we have already tested and validated over wired networks and how we adapt it to an ad hoc environnment by adding other building blocks.

Figure 1 shows the main three building bloks of our enhanced approach :

- the functionnal architecture from the BAAL protocol [5]
- the hybrid support from the AKMP protocol [3]
- the cryptography issue from the threshold cryptography [17]

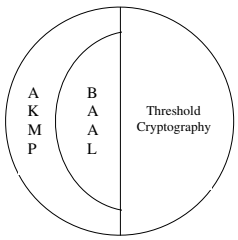

Fig. 1. Enhanced Approach Architecture

4.1 BAAL

BAAL is a group key management protocol which ensures access control, data confidentiality and authentication of group members within wired networks. These services are achieved by distributing only one key K_{grp}, this makes BAAL belongs to the approach A presented in Section 3.

The three actors met in the BAAL architecture are :

1. The Global Controller (GC): may be an organizer of conferences, which can create one or several secured groups in the Internet. It holds a list of future participants in the group. This list can be created by others means like e-mail or fax. The GC creates the key group and distributes it to all the participants via the local controllers. Moreover, the GC has to re-key periodically or sometimes occasionally.
2. The Local Controller (LC) : is delegated by the GC. It receives the group key and distributes it to all the participants in its network, during the initial configuration of the group. The LC can create and distribute a new group key, accept or refuse a member and notify the others LCs in the case of group changes.
3. The Members of Group (MG) : member of the list of participants, or any member which joins the group later on.

The operations defined for group key management are :

- Group initialisation. Within this operation, K_{grp} is safely distributed to all elements of the participant list. The delegation of the LCs by the GC is also done, in order to guarantee the local access to the group and the cooperation with others controllers to manage the group key. The group initialisation is performed in two phases : invitation phase which is reserved to the invitation of all members of the participant list, and group key distribution phase which is dedicated to safely distribute the group key to group members and to delegate the LCs.
- Addition of a new entity. To join a multicast group, an entity must first be allowed to do so. This condition is verified by the LC which decides to add the entity to the group or not. In the affirmative case a rekey process must be triggered. If the LC is already delegated, it will generate and distribute a new key for all members of the group, else it will negociate with the GC to obtain permission for triggering the re-key process.
- Withdrawal of an entity. A member wanting to leave the group sends a Leave message in order to stop group traffic flow. If a member is detected as malicious, it will be automatically excluded. This is done by renewing the group key.
- Periodic re-key. Generally, cryptographic keys have a limited time to live and must be periodically renewed. This renewal can be done by the GC or by the LCs.

4.2 AKMP

The main idea of AKMP (Adaptive Key Management Protocol) is to meet approach A as long as no frequent membership change is depicted by group members, and to switch to approach B whenever members show a certain level of dynamicity. The decryption/re-encryption process is only restricted to sub-networks that are subject of high dynamicity.

The protocol begins with a single group that shares a unique TEK. This group is initially managed by one AKMP router. During the multicast secure session, if an AKMP router detects a local dynamicity, it initiates a subgroup with an independent local key. To do so, the concerned AKMP router generates and distributes the local key to the members in the constructed subgroup. This key is called a Downstream Key (DK). Then, the router decrypts received packets using its parent AKMP router key (called Upstream Key UK), and re-encrypts the packets using DK. The AKMP router has so switched from an inactive state to an active state. Thus, AKMP reduces decryption/re-encryption overhead to the minimum while attenuating the "1 affects n" phenomenon. Within each AKMP router, an evaluation function f_i is implemented and sets the AKMP router state according to the mcf: number of members changes per unit of time.

Each AKMP router holds DK_i and UK_i , upstream and downstream keys, and a pair of keys (public and private) allowing secured exchange between different AKMP routers. When an AKMP router i detects a high dynamicity within his sub network (f_i = true), it switches to active state, generates a new DK_i and distributes it to all local members. Then it must send its old DK_i to its AKMP router parent J so that J generates and distributes its new key DK_j for all its child local members in the case of $oldDk_i = DK_j$. In the case of (f_i = false), the AKMP router stays passive. Thus, when it detects a Join or a Leave event, it must notify its AKMP router parent in order to update and distribute DK_j.

4.3 Threshold cryptography

BAAL uses a public key infrastructure which involves the presence of a CA (Certification Authority). But, within an ad hoc network, having only one CA presents a single point of failure. If the CA is unavailable, secured communications between nodes become impossible. The adversaries can also use this failure to compromise all the network. The CA duplication within ad hoc networks would bring more reliability but also more risks of malicious attacks since it also duplicates the possibility to compromise one CA.

Threshold cryptogpraphy [17] solves this problem. The new key management service, having $(n, t+1)$ configuration, consists on having n special nodes called servers within the ad hoc network. Every server holds its pair keys, and public keys of all nodes in the network, particularly those of the others servers. This fact allows servers to communicate together securely. [16] proposes to distribute trust to nodes having a relatively hight physical security and a good computing power. The authors call these nodes MOCA (Mobile Certificate Authority). In the $(n, t+1)$ configuration, the n servers share capability to sign certificates for the other nodes in the network. The private key of all the service (k) is divided into n secrecies $(s1, s2, ..., sn)$, each secrecy corresponding to one server.

Each server generates a partial signature of node cetificate and sends it to a combiner, which needs t+1 partials signatures to compute the complete signature. [17] assumes that $(n >= 3t + 1)$; t represents the maximum number of compromised servers. Thus, even if t servers are compromised, the combiner is able to generate the node signature.

The choice of t is detailed in [16], a great t offers more safety but at the same time, generates more traffic overhead.

The combiner, essential for the node signatures generation, can also be compromised. [9] proposes a replication of the combiner in many CA : a co-operative architecture of combiners, which can be able to be formed with stolen around the node, and generate for it its signature.

[16] presents a certification protocol, called MP (Moca Certification Protocol). The clients according to this protocol, broadcast messages Send Request (SREQ). Each MOCA server receiving this message, responds with a message Certif Response (CREP), containing a partial signature. When the client collects the $t + 1$ valid CREPs, it can constitute its signature. To reduce the flooding, a proposed solution is the B-unicast, which allows nodes to send by unicast to $t + 1$ MOCAs, if it holds in its routing table enough information concerning their routes. Otherwise, the node will be obliged to broadcast its SREQ over all the network.

5 An enhanced hybrid key management protocol for secure multicast in ad hoc networks

The context of our approach is a set of ad hoc nodes, having capability of communicating in both unicast and multicast mode. This proposal has to be independent of used routing protocols, and thus, will start after construction of the multicast tree.

To ensure different security services for group communication in ad hoc networks : e.g. authentication, confidentiality, integrity and non repudiation, we need a group key management architecture.

Within our environment, we establish a threshold cryptography infrastructure which gives to each entity a public and private key (K_i , k_i). Group key generation is also achieved using threshold cryptography.

5.1 Approach Architecture

Like in BAAL, the main actors in our architecture are the global controller (GC), local controllers (LCs) and Group Members (GMs).

- The Global Controller (GC) is the group source. Initially, this entity holds group participants list named Participant_List. The GC is responsible for the generation and distribution of the group key, it also ensures periodic renewal of this key, and group security management (controlling local controllers and group members behavior). All the controllers (global and local) hold a same list named Recovery_List, which contains members excluded of the group, this list will be used at the time of Join and Leave of entities in the group.

– The Local Controller (LC). Every mobile node, belonging to the multicast tree, as a group member or a simple participant to the multicast tree, and having child nodes to which it conveys multicast flow, is considered as a passive local controller. Every local controller holds its local members list, named Local_Participant_List. It must convey multicast flow, sent by source, to all members of this list. If a passive LC is a group member, it must hold the same cryptographic key as its parent node. Otherwise, if the passive LC is a simple participant, it will not need the node-parent key. When the local dynamicity rate and the local members number reaches certain thresholds, the LC decides to switch to an active state. Thus, it generates a new local key, distributes it to all its members and starts a decryption/re-encryption process. We say that this active LC forms with its local members a new cluster. To decide on its state, every LC holds a dynamicity-evaluation function, described in what follows :

if ($mcf >$ d1 or $mn >$ d2) then {switch to dec/rec process}
  f_i = true;
else
  f_i = false;
end;
with mcf: number of members changes per unit of time, d1 : predefined fequency threshold.
mn : local members number, d2 : predefined member number.

To obtain mn, a LC counts its passive child nodes with their child nodes, and its active child nodes without their child nodes. Figure 2 gives an example of calculation of mn.

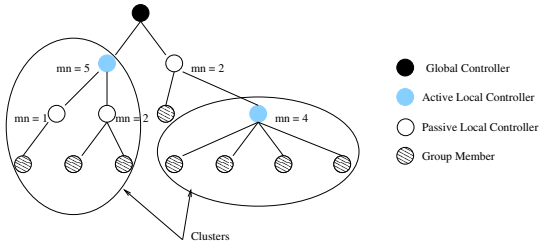

Fig. 2. Example of mn evaluation

This function differs from the AKMP evaluation function by taking into account not only the members-changes frequency but also their number. This is necessary to secure group communications in ad hoc networks for two reasons. First, all group members are also routers and so they can be considered as local controllers (if they have child nodes). Second, at the time of a leave, the active local controller is obliged to renew the local key and to distribute it, in unicast, at all its cluster members. Thus,time necessary for this renewal is proportional to cluster-members number.
– Group Member(GM) is a member of the list Participant_List or a member joining the group later.

We now present the different operations we have extended in our approach.

5.2 Operations

Group initialization The GC initializes the both lists Participant_List and Recovery_List. Then, it starts the phase of the group key generation and distribution. The group key generation is realized using threshold cryptography with B-Unicast as decrypted below. If an $(n, t+1)$ configuration is established, the GC consults its routing table, and checks whether it holds routes to t+1 servers. If it is the case, it will send them a Key_Query message, otherwise it will broadcast this message to the whole network. At the reception of one Key_Query message a server starts by authenticating the sender. If the authentication succeeds, it generates a partial signature and includes it within a Key_Resp message. This message will be sent to the GC encrypted with its private key. The GC remains on standby of t+1 valid partial signatures, sent by the t+1 first servers.

On their arrivals, it combines them to obtain the key group. The servers are provided with algorithms allowing them to generate, randomly, partial signatures for other nodes. In order to secure communications during key distribution, the sent message includes the sender's signed token. Thus, the signed token is essential for the authentication process in our approach. It allows a receiver to check the message origin and the sender's identity. A token contains :

1. an identity of the sender, e.g. its IP address ;
2. a timestamp ;
3. a random number, used to protect receivers against the replay of messages.

Each token is included within the message, signed with the private key of its sender. The GC sends the following message to Participant_List members (M_i), encrypted with their respective public keys (K_i). This message contains the group key, the group identity, the identity and the signed token of the GC.

GC –> M_i : {K_{grp}, IDG, ID_{GC}, [token_GC]‘Prv_GC}‘K_i
with IDG : group identity, ID_{GC} : GC identity, [token_GC]‘Prv_GC : GC signed token.

After receiving this message, each group member decrypts it, authenticates the GC and extracts the key group. Then, it sends to GC a Report message containing the group identity, its identity and its signed token, encrypted with GC public key.

M_i –> GC : {IDG, ID_M_i, [token_M_i]‘Prv_M_i}‘Pbk_GC
with Pbk_GC : GC public key, [token_M_i]‘Prv_M_i : M_i signed token.

Addition of a new entity An entity wanting to join the group, sends a Report message to its LC. This message contains its signed token, its identity and the group identity.

M_i –> LC : {IDG, ID_M_i, [token_M_i]‘Prv_M_i}‘Pbk_LC
with LC : Local controller of the new entity.

After receiving this message, the LC authenticates the signed token. If the authentication succeeds, the LC checks whether the new entity does or not appear in the Recovery_List. At this stage, the LC calculates its dynamicity-evaluation function. Two cases are identified :

1. if (f_i = true) : the LC switches to active state. Thus, it must generate a new local key using threshold cryptography and distribute the new key K_i^{loc} to its local old members, encrypted with the old key old_K_i^{loc} , and send the same key to the new member encrypted with its public key. Moreover, the LC must send its old local key to its active parent node M_l in order to update the key K_l^{loc} if old_K_i^{loc} = K_l^{loc} (when the LC switches, at the first time, to active state, its key is the same as its active parent node key).

   for j:1..nb_old_attached_members
   LC –> M_j : {K_i^{loc}}‘old_K_i^{loc}
   LC –> M_i : {K_i^{loc}}‘K_i with M_i : new member
   LC –> Active_Parent_Node M_l: {old_K_i^{loc}}‘K_l where K_l : public key of the parent node M_l.

2. if (f_i = false) : the LC remains in passive state. Thus, it sends a request for key renewal to its active parent node, which will start generation and distribution of a new key to all its local members.

Withdrawal of a group entity We distinguish two cases, voluntary withdrawal and expulsion. The first is realized when a member decides to leave the group. Thus, it sends a Leave message to its LC, containing its signed token. In this case, the LC removes the entity from its Local_Participant_List and starts the renewal key phase. The second case (expulsion) is named member revocation, which takes place when the member can place the safety of the group in danger. Thus, the LC adds this member in Recovery_List, removes it from the Local_Participant_List and starts the renewal key phase. Within the renewal key phase, which is basically the same as entity addition, two cases are identified :

1. If (f_i = true) : the LC switches to the active state. Thus, it must generate a new local key using threshold cryptography and distribute then new key K_i^{loc} to its local old members, while excluding the leaving member, encrypted with their respective public keys. Moreover, the LC must send its old local key to its active parent node M_l in order to update the key K_l^{loc} if old_K_i^{loc} = K_l^{loc}.

   M_s member leaving the group,
   for j:1..nb_attached_members, j different from s,
   LC –> M_j : $\{K_i^{loc}\}$‘K_j
   LC –> Active_Parent_Node M_l: $\{\text{old\_}K_i^{loc}\}$‘K_l with K_l : public key of the parent node M_l.

2. If (f_i = false) : the LC remains in passive state. Thus, it sends a request for key renewal to its active parent node, which will start generation and distribution of a new key to all its local members.

Periodic renewal of the group key Cryptographic keys have a limited time to live. Thus, they must be renewed periodically. The renewal period is determined according to the key length and the key generation algorithm. The periodic key renewal is done by the GC or all the active LCs. This renewal is operated in two stages : key generation using threshold cryptography and key distribution to the local members.

5.3 Mobility Processing

Our approach has to be adapted to mobile ad hoc network features. So we have studied some mobility scenarii involving the main actors of the key distribution architecture :

1. When an LC moves, all its local members will be unable to receive multicast data sent by the source. To solve this problem, these members must immediately connect themselves to another LC in order to continue to receive multicast flow. It remains to see how the transition between clusters is done and how much data members transiting between clusters will lose. We distinguish two cases :
   (a) The LC moves with notification : this notification can be a message sent by the LC, through multicast, to all its local members. Thus, the local members will send Report messages to another LC in order to obtain its local key and continue multicast data reception.
   (b) The LC moves without notification : the local members will realize, after a certain period of time, that the route to the source is not assured any more by their LC. They must join the group via another cluster.
2. When a group member moves outside its cluster, it will lose the multicast connectivity. It must choose another route to the group source by connecting itself to another LC and by authenticating itself to others LCs when it begins to move.

To notify of its movement, a node must detect as soon as possible changes in its multicast routing table. These solutions are dependent of security and QOS policies, established in the ad hoc network. According to these policies, we can allow or not a latency time necessary for a member, in movement, to find the multicast data reception.

6 Simulation and Results

In this section, we present simulations realized to define frequency and member number thresholds. Beyond these thresholds, a local controller switches from passive to active state. For the simulation, we have studied the 1 affects n behavior, using the threshold cryptography with (5,3) configuration. We have also measured the time necessary for the group key renewal, after a Join or a Leave message, according to the event frequency and the group members number.

These two thresholds give us a first evaluation of our solution. They garantee that the time necessary for a Join or a Leave cannot exceed an upper value. Having these threshold, we can also vary the threshold cryptography configuration, in order to enhance the processing within our architecture.

6.1 Simulation Model

To do the simulation, we used the NS simulator, version ns2.1b9a [11]. The simulated network is an ad hoc network, composed of 100 nodes, and using MAODV [13] as the multicast routing protocol. Nodes moving is generated randomly in order to take account the mobility factor of ad hoc networks. To generate realistic multicast sessions, we use the model presented by Almeroth [1], which suggests that member arrival follows a Poisson processus ($\lambda = 10$ arrivals by time unit) and the membership duration is an exponential distribution (in average $\mu = 145$ time units). This model is deduced from real multicast sessions observed on the Mbone.

Figure 3 shows the variation of group members number by time, for the simulated model.

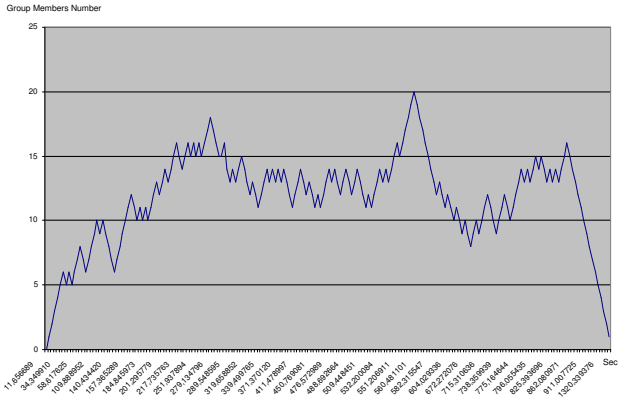

Fig. 3. Variation of group members number

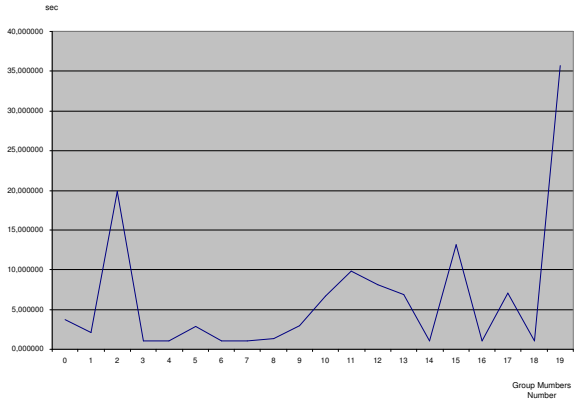

Fig. 4. Renewal Key Time following a Join by group members number

6.2 Simulation Results

Before starting simulation, we have calculated the time induced by a Join and a Leave procedure within the multicast group. For the Join event, the number of sent messages to renew the group key is 10, which are :

- 2 messages for new member authentication.
- 3 messages sent by the GC to 3 servers in order to generate the new group key, and 3 response messages.
- 1 message sent by the GC in multicast, to all old members, containing the new group key.
- 1 message sent by the GC in unicast, to the new member, containing the new group key.

The transmission of the new key needs operations of encryption/decryption, for which we chose the 3DES algorithm. Eight operations of encryption/decryption are necessary for a Join event:

- 3 operations of encryption realized by servers generating the new key.
- 3 operations of decryption realized by the GC while receiving the 3 partial signatures from servers.
- 1 operation of encryption of the new key while sending it in multicast, to all group members.
- 1 operation of decryption of the new key realized by the new member. We assume that old members will decrypt the key at the same time.

The average time to send a message in our network is $t = 0.002579s$ and the time to achieve one decryption or encryption operation, according to 3DES, is $d = 0.000712s$ (1500 bytes). Thus, we estimate the average cost for the renewal key during a member addition to $10 * t + 8 * d = 0.031486s$. This cost is added to the latency time between the Join query and the effective Join.

The same study is done for a Leave procedure: the number of sent messages for renewing a group key is $(7+n)$ with n : group-members number :

- 1 message of Leave query.

– 3 messages sent by the GC to 3 servers in order to generate the new group key, and 3 response messages.
– n messages sent by GC in unicast to residual members, containing the new group key.

To ensure key confidentiality after a Leave event, we need (6+2n) encryption/decryption operations :

– 3 operations of encryption realized by servers generating the new key.
– 3 operations of decryption realized by the GC while receiving the 3 partial signatures from servers.
– n operations of encryption of the new key while sending it in unicast, to all residual group members.
– n operations of decryption realized by residual group members while receiving the new group key, using their private keys.

Figure 4 presents the time necessary for the rekey-process following a Join procedure by the group members number.

Considering that the generated cost for the renewal key is constant, the different variations shown in the curve are due to the fact the latency time between a Join query and the effective Join can vary due to the MAODV protocol and also to the location of the new member compared to the others group members. Thus, this curve does not permit to define the threshold of group members number in a cluster.

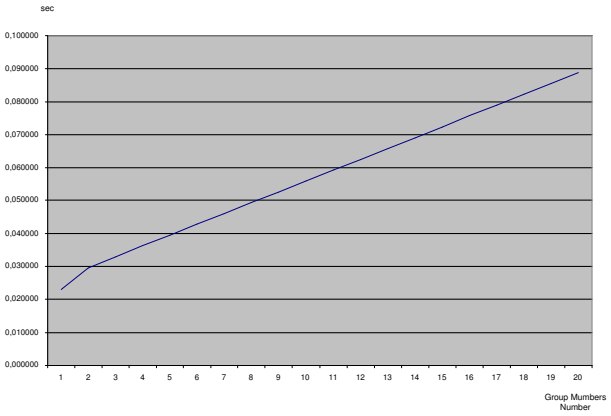

Fig. 5. Renewal Key Time following a Leave by group members number

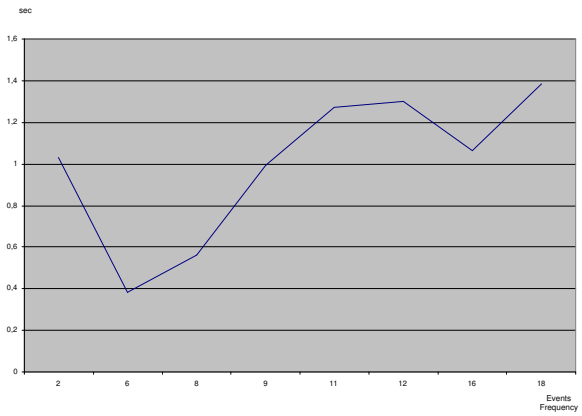

Fig. 6. Renewal Key Time by events frequency

Figure 5 shows that the time necessary for the rekey-process following a Leave event, is proportional to the group members number. Thus, we can define the threshold of members number within a cluster. If we take, for example, as a constraint that the time necessary for the rekey-process following a Leave event cannot exceed 0.05s, the threshold will be 8 members per cluster.

To carry out Figure 6, we calculated the average time necessary for the rekey-process following a Join or a Leave procedure by events frequency calculated within equal-time intervals. If we take as a second constraint the fact that the time for the rekey-process by the frequence cannot exceed 1s, the curve shows that, once established the steady operation, the threshold of frequence is 9 events per unit of time.

7 Conclusion and future works

In this paper, we addressed the security of one service within an ad hoc infrastructure namely multicast.

We presented an enhanced architecture starting from BAAL which is a group key management protocol, already tested and validated over wired networks. To adapt BAAL to ad hoc environment, we use the hybrid support of AKMP in order to ensuring dynamicity and scalability. Our enhanced architecture ensures also secure generation of group keys by using the threshold cryptography.

Our approach meets some challenges posed by securing multicast communications within an ad hoc network, namely :

– The problem of lack of infrastructure to ensure group source and members authentication is solved by using the threshold cryptography. In fact, this method consists on sharing a central authority into many nodes which we call servers. These servers share the capacity to sign certificates for all the ad hoc network nodes.
– The support of AKMP ensures dynamicity and scalability of our approach. Indeed, AKMP solves the problem "1 affects n" met in BAAL while limiting the encryption/decryption computing overhead due to the clusterisation of the group in sub-groups. Thus, our enhanced architecture allows to cluster the multicast group, dynamically, according to the frequence of the events Join and Leave, and to the members-number per cluster.
– The mobility of nodes within our environment has been taken into account. In fact, we studied some mobility scenarios involving the main actors of the key distribution architecture.
– To solve the trust problem concerning securing cryptographic keys generation, we also used the threshold cryptography which allows the keys generation by servers having a great computing power and a good physical security.

We realized simulations to define frequency and member number thresholds. Beyond these thresholds, a local controller switches from passive to active state. Thus, these thresholds garantee that the time necessary for a Join or a Leave cannot exceed an upper value.

As future works, we plan to improve this approach by integrating service availability and securing routing informations within the network. We plan also to achieve the reliability of the keys distribution using acknowledgements.

References

1. K. Almeroth and M. Ammar. Collecting and modelling the join-leave behaviour of multicast group members in the mbone. In *The Symposium on High Performance Distributed Computing*, Syracuse NY, 1996.
2. M. Archer. Proving correctness of the basic tesla multicast stream authentication protocol with tame. In *Workshop on Issues in the Theory of Security, 2002*, 2002.
3. H. Bettahar, A. Bouabdallah, and Y. Challal. An adaptive key management protocol for secure multicast. In *11th International Conference on Computer Communications and Networks ICCCN*, Florida USA, October 2002.
4. M. Bohge and W. Trappe. An authentication framework for hierarchical ad hoc sensor networks. In *WISE'03, San Diego, California, USA*, September 2003.
5. G. Chaddoud, I. Chrisment, and A. Schaff. Baal : Sécurisation des communications de groupes dynamiques. In *The Proceedings of the 8th Colloque Francophone sur l'Ingénierie des Protocoles CFIP'2000*, Toulouse, France, October 2000.
6. T. Chiang and Y. Huang. Group keys and the multicast security in ad hoc networks. In *Proceedings of the 2003 International Conference on Parallel Processing Workshops (ICPP 2003 Workshops)*, 2003.
7. ETSI. High Performance Radio Local Area Network (Hiperlan), draft standard ETS 300652, March 1996.
8. T. Hardjono and L. Dondeti. *Multicast and Group Security*. Computer Security Series. Artech House, Librarie Eyrolles, 2003.
9. V. Legrand, F. Abdesselam, and S. Ubéda. Etablissement de la confiance et réseaux ad hoc - un état de l'art. Technical report, Laboratoire CITI - INRIA ARES, 2003.
10. G. Lin and G. Noubir. Secure multicast over multihop wireless ad hoc networks. In *Workshop on Mobile Ad Hoc Networking and Computing*, March 2003.
11. C. Lindemann. http://rul-www.cs.uni-dortmund.de/MobileP2P/mainE.html, 2003.
12. S. Mittra. Iolus: A framework for scalable secure multicasting. In *SIGCOMM*, pages 277–288, 1997.
13. E. Royer and C. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) routing, IETF Internet Draft: draft-ietf-manet-maodv-00.txt, 2000.
14. H. Sallay, A. Lahmadi, O. Festor, and I. Chrisment. Extension de l'architecture active amam pour le support des services de sécurité multicast. In *GRES'2003 Colloque Francophone sur la Gestion de Réseaux et des Services*, February 2003.
15. V. Varadharajan, M. Hitchens, and R. Shankaran. Securing ntdr ad-hoc networks. In *IASTED International Conference on Parallel and Distributed Computing and Systems 2001*, pages 593–598, Anaheim California, August 2001.
16. S. Yi and R. Kravets. Key management for heterogeneous ad hoc wireless networks. Report Research UIUCDCS-R-2002-2290, UILU-ENG-2002-1734, University of Illinois at Urbana-Champaign, Department of Computer Science, 1304 West Springfield Avenue, Urbana, IL 61801-2987 USA, July 2002.
17. L. Zhou and J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.