*Journal of Universal Computer Science, vol. 18, no. 6 (2012), 874-898*
*submitted: 15/10/11, accepted: 15/2/12, appeared: 28/3/12* © *J.UCS*

# Adaptive Group Key Management Protocol for Wireless Communications

**Saïd Gharout**
(Orange Labs - France Telecom, Caen, France
Computer Science department, University A/Mira of Bejaia, Béjaïa, Algeria
said.gharout@orange.com)

**Abdelmadjid Bouabdallah**
(Heudiasyc Lab. UMR CNRS 6599, University of Technology of Compiegne
Compiègne, France
Computer Science department, University A/Mira of Bejaia, Béjaïa, Algeria
bouabdal@hds.utc.fr)

**Yacine Challal**
(Heudiasyc Lab. UMR CNRS 6599, University of Technology of Compiegne
Compiègne, France
ychallal@hds.utc.fr)

**Mohammed Achemlal**
(Orange Labs - France Telecom, Caen, France
mohammed.achemlal@orange.com)

**Abstract:** Group-oriented services and wireless communication networks are among the emerging technologies of the last few years. Group key management, which is an important building bloc in securing group communication, has received a particular attention in both academic and industry research communities. This is due to the economical relevance of group-based applications, such as video on demand, videoconferencing, collaborative work. The key management concerns the distribution and updates of the key material each time a member joins or leaves the group. The dynamic aspect of group applications due to free membership joins and leaves in addition to members' mobility makes difficult the design of efficient and scalable key management protocols. Indeed, to secure group communication in mobile environment, the protocol must deal not only with dynamic group membership but also with dynamic member location. Thus, the challenges in designing secure and scalable key management protocols are: the dynamic updates of the key caused by frequent joins and leaves, the large size of the group and the mobility of group members. Many researches have addressed the first and the third challenges. However, the mobility challenge has not been widely addressed. In this paper, we present our solution for group key management with a mobility support. Our protocol focuses on the above three challenges. It is highly scalable to dynamic groups and treats the nodes' mobility with a null re-keying cost and keeps perfect backward and forward secrecies. Our simulation studies show that our protocol makes better performance compared to other protocols while reducing the overall overhead and the number of re-keying messages and has no security failures.
**Key Words:** confidentiality, key management, group communication, mobility
**Category:** C.2.0

## 1 Introduction and challenges

The phenomenal growth of the Internet in the last few years and the increase of bandwidth in today networks have provided both inspiration and motivation for the development of new *group-oriented* applications and services, combining voice, video and text "over IP". Nowadays, group-oriented applications are increasingly deployed over the Internet such as video conferencing, interactive group games, video on demand (VoD), IP-TV, e-learning, software updates, database replication and broadcasting stock quotes. Unfortunately, the strengths of group-oriented applications, implemented either by IP Multicast [Savola 2008], overlay Multicast [Yeo et al. 2004] or other means is their lack of security. Indeed, the open and anonymous membership [Haberman and Martin 2008] and the distributed nature of multicasting are serious threats to the security of this communication model. For this purpose, much efforts have been conducted to address the many issues relating to securing group communication, such as: access control, *confidentiality*, authentication and watermarking.

Group communication confidentiality requires that only group members could read data even if the data is broadcasted into the entire network [Judge and Ammar 2003, Baugher et al. 2005]. Typically, the distribution of data with commercial value (video streaming, ...)or State top-secret content requires the use of appropriate mechanisms to prevent non-legitimate recipients from having access to the content. To ensure confidentiality in group communication, only the customers authorized for the service would have access to the content for only the duration corresponding to their authorization. A straightforward solution is to encrypt the group-intended data by the sender with a group key, called *Traffic Encryption Key* (TEK), common to all authorized recipients. However, when the authorized duration for a recipient expires (in video streaming or audio conferencing for example), it is necessary to change the common group (traffic encryption) key into a new key in order to prevent the leaving customer from having access to the content beyond the limit of his authorized duration. Therefore, the sender has to share the new TEK with all legitimate recipients except the leaving one. This phase is called *re-keying*, and should be performed each time a customer *joins* the secure session to prevent him from having access to old content (what is called *Backward Secrecy*) or *leaves* the session to prevent him from having access to future content (what is called *Forward Secrecy*). The role of a group key management protocol is to generate, update and distribute TEKs to legitimate group members. To ensure perfect backward and forward secrecy, a re-keying must be done each time there is membership changes (*join* or *leave*) in the group. The impact of this re-keying process on group members, commonly called *1-affects-n* phenomenon, measures the number of affected members by a re-keying process. This *1-affects-n* phenomenon is a challenging issue in designing group key management protocols. If the group size keeps increasing, such a phenomenon will significantly degrade the system performance. In the last few years, a lot of group key management protocols have been conducted in the literature to address the confidentiality issue in group communication [Rafaeli and Hutchison 2003]. Even though a multitude of data confidentiality mechanisms currently exist for the fixed Internet, this security service remains a challenging problem in terms of scalability, efficiency, and performance. A critical problem with any re-key technique is scalability: as the re-key process should be triggered after each membership change, the number of TEK update messages may be important in case of frequent join and leave operations. Some solutions propose to organize the secure group into subgroups with different local TEKs. This reduces the impact of the key updating process, but needs decryption and re-encryption operations at the borders of subgroups. These operations may decrease the communication quality and causes computational overheads.

While the problem of developing efficient group key management protocols is difficult, the problem becomes more difficult and complex when we consider member mobility [Romdhani et al. 2004]. Indeed, in the mobile environment [Perkins 2002, Johnson et al. 2004, Koodli 2009] where members can move inside the group the key management protocol must deal not only with dynamic group membership (*join* and *leave*) but also with dynamic member *location* [Romdhani et al. 2004]. In mobile environments [Perkins 2002, Johnson et al. 2004, Koodli 2009],when a member moves from an access area to another, it changes its access point to the traffic. So, the complexity of key management increases since the member is not known in the new area even though a fast handover mechanism exists. This can re-route the traffic to the new access point but the security issue must also be considered. This can induce the generation of new TEKs in both old and new areas. In this case the member is considered as a leaving member in old area and as joining member in the new area even it is a valid member in the system.

In this paper, we propose a decentralized architecture for group key management in mobile environments where the group is organized into clusters of areas, and areas of the same clusters use a common TEK. We call it *KMGM* for **K**ey **M**anagement to secure **G**roup Communications in **M**obile Environments.

The rest of this paper is organized as follows: In section 2, we overview group key management protocols in the literature. In section 3 and section 4, we present our protocol for group key management in mobile environments and our re-keying strategy, respectively. Section 5 details how members are verified in the system when they move. We present our simulation model and results in section 6. Finally, we conclude the paper in section 7.

## 2 Related works

Group key management has been extensively studied in the literature [Rafaeli and Hutchison 2003, Zhu and Jajodia 2004, Challal et Seba 2005]. Many classifications of group key management protocols have been proposed in the literature: *centralized*, *decentralized*, and *distributed* protocols. In *centralized* protocols, the key distribution function is assured by a *single entity* which is responsible for generating and distributing the traffic encryption key (TEK) whenever required. In *decentralized* protocols, a hierarchy of key *managers* share the labor of distributing the TEK to group members in order to avoid bottlenecks and single point of failure. In *distributed* key-agreement protocols, the group *members* cooperate to establish a group key. Another classification is done according to the manner that the TEK is shared between group members. Thus, we distinguish *common TEK* approaches and *per subgroup TEK* approaches. In the common TEK approach, a single TEK is used for the whole group, whereas in the per subgroup TEK approach, the group is organized into subgroups (areas) and a different TEK is used for each subgroup. Most of the centralized protocols proposed in the literature use a common *Traffic Encryption Key (TEK)* for group members [Challal et Seba 2005]. A typical solution that fits into this category is the *Group Key Management Protocol* (GKMP) [Harney and Muckenhirn 1997]. The drawback of this scheme is its high number of update messages (in the order of $O(n)$ with $n$ being the number of valid group members) to transmit the new TEK after membership changes. In the *Logical Key Hierarchy (LKH)* protocol proposed at same time by Wong et al. in [Wong et al. 2000] and Wallner et al. in [Wallner et al. 1999], a *hierarchy of keys* is used to reduce the required number of *TEK update messages* induced by re-keying after membership changes to the order of $\log(n)$. The drawback of these centralized protocols is the fact that they require that all group members commit to a new *common TEK*, whenever a membership change occurs in the group, in order to ensure *perfect backward and forward secrecy*. This makes these solutions suffering from the *1-affects-n* phenomenon. In order to mitigate the *1-affects-n* phenomenon, another decentralized approach have been proposed and which consists in organizing group members into subgroups, where each subgroup uses its own independent TEK. Indeed, in this scheme, when a membership change occurs in a subgroup, it affects only the members of this subgroup. Mittra proposed in [Mittra 1997] the *Iolus* architecture which is a framework of a hierarchy of multicast subgroups. Each subgroup is managed by a Group Security Agent (GSA) which is responsible for key management inside the subgroup. A main controller called the Group Security Controller (GSC) manages the GSAs. When a membership change occurs in a subgroup, only that subgroup is involved in a re-key process. This way, Iolus scales to large groups and mitigates *1-affects-n* phenomenon. However, Iolus has the drawback of affecting the *data path* since data is translated (decrypted and re-encrypted) each time it passes from a subgroup to another. Even though it is difficult to design efficient and scalable group key management protocols, the problem becomes more difficult while we consider node mobility. Indeed, in a wireless environment a node participating in a group session can move from a subnet or area to another. In traditional solutions, this mobility can induce a re-keying process two times since this can be considered as a *leave* in old area and a *join* in new area.

In the literature, few group key management protocols have been designed for mobile environments [Cao et al. 2006, Roh and Lee 2006]. In [Roh and Lee 2006], Roh et al. propose two key management schemes (KTMM and WSMM) suited to the Mobile IP environment. The first scheme matches the key management tree to the mobile multicast network topology. The second scheme separately manages each the wired and wireless area. These solutions suffer from a single point of failure which is the group manager. In [Di Pietro et al. 2002], Di Pietro et al. propose an improved version of LKH [Wong et al. 2000, Wallner et al. 1999] called LKH++ for mobile wireless networks but it does not treats the mobility issue of members. In M-Iolus [Kamat et al. 2003] which is a version of Iolus that supports members mobility, subgroups are divided into micro-subgroups to reduce the *1-affects-n* phenomenon. This has a price since it increases the number of encryption areas. When a member moves from a micro-subgroup to another micro-subgroup the TEK is not updated. M-Iolus proposes a null rekeying cost on member mobility, at the expense of raising a backward secrecy violation.

In [Hernandez-Serrano et al. 2005], authors propose a group key management protocol for wireless networks with a slightly mobile set using clustered areas. They propose to use LKH for intra-area rekeying but the use an inter-area rekeying when the mobile node moves from area to another the same in [Zhang et al. 2002]. In [Zhang et al. 2002], where the group is divided into areas [Hardjono et al. 2000] using a common TEK authors propose several techniques for inter-area rekeying when members move: Static Rekeying (SR), Baseline Rekeying (BR), Immediate Rekeying (IR) and FEDRP (First Entry Delayed Rekeying) when mobile node moves from an area to another.

In [Mat Kiah and Martin 2007, Mat Kiah and Martin 2008], Kiah and Martin propose a key management protocol where they address the mobility issue of members. They consider a DKM (*Domain key Manager*) as the main key manager of domain $i$ and AKM (*Area Key Manager*) as the key manager of the area $j$ inside a domain. A list called $MobList$ is used by the DKM and AKMs to keep trace of mobile members. Thus, each time a member moves from one area to another, the protocol stores in $MobList$ IDs of the moving member, multicast group joined by the member, area that a member is moving from, ID of the visited area that a member is moving to. When a member $M_i$ moves from area $i$ to area $j$ it notifies the new and previous AKMs. The two messages are encrypted with two different keys ( $A_iM_i\_Key$ shared between $AKM_i$ and $M_i$ and $Sm\_Key_{ji}$ between $AKM_j$ and $M_i$). The drawback of this protocol is the number of used keys and signaling messages. The protocol did not explain how leave re-keying is done since there is no re-keying while moving. In [Gharout et al. 2010], we proposed a host mobility management for group key management protocols.

### 2.1 Discussion and motivation

We notice that both existing approaches suffer from great concerns depending on group dynamism: the *common TEK approach* suffers from the *1-affects-n* phenomenon, where a single group membership change (join or leave) results in a re-keying process that disturbs all group members to update the TEK. Moreover, centralized protocols are not scalable, and distributed ones bring new challenges such as synchronization and conflict resolution. On the other hand, the *TEK per subgroup approach* reduces the *1-affects-n* problem. This is advantageous for highly dynamic groups. However, this approach requires translation of sent messages whenever they pass from a sub-group to another, and this may not be tolerated by applications that are sensitive to packet delivery delay variations. Besides, this approach would not be worthy with relatively static groups because the multiple translations would induce avoidable delays and useless computation overheads. These shortcomings are due to the lack of *dynamism awareness* in existing group key management schemes.

## 3 KMGM protocol: Mobility support to ASGK

### 3.1 Overview

In KMGM, we organize the group into a *hierarchy of administrative areas* where each area is managed by an *Area Key Distributor* (AKD). This hierarchy of areas is partitioned into clusters of areas (see figure 1). If there is only one cluster, our architecture is then similar to a *common TEK* approach, and if the number of clusters is equal to the number of areas our architecture becomes a *TEK per subgroup* approach.

In our architecture, an area represents an autonomous system which can be a corporate network, a multicast domain (e.g. PIM domain), or a wireless area. Figure 1 illustrates the different components of the KMGM architecture.

Each *AKD* is responsible for the key management process inside the area under its control and is considered as a member in its area and in its parent area. Thus, an *AKD* plays the role of a *proxy* for its area in the parent area. When an *AKD* receives a message from the parent area, it sends it to members of the area under its control. Areas of the same cluster use the same TEK which is

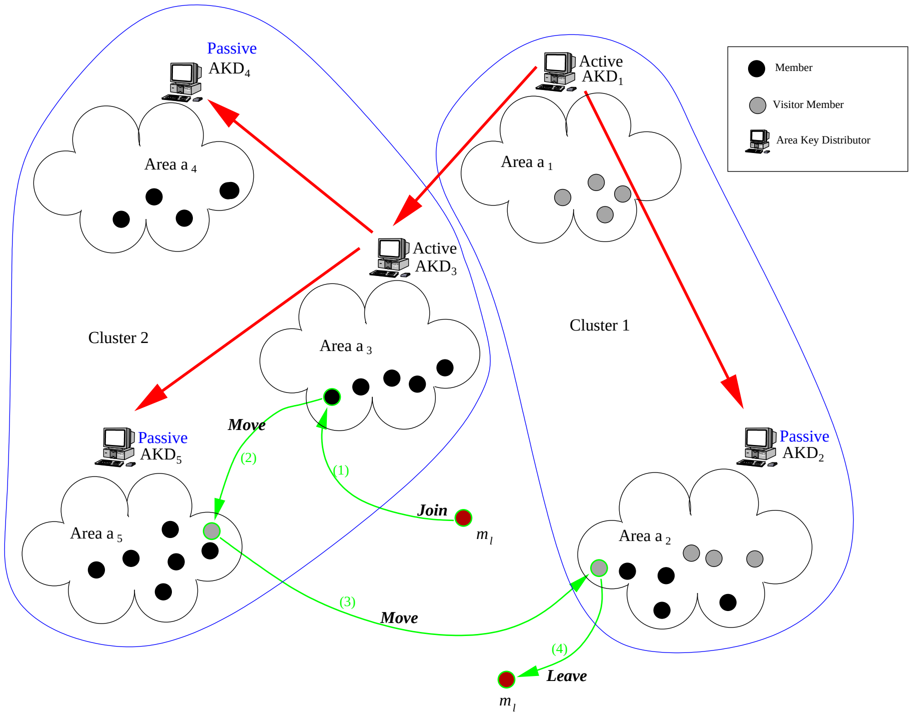


**Figure 1:** The KMGM Architecture and mobilty scenarios.

different from TEKs of other clusters. The task of internal AKDs in a cluster is to forward received messages sent by the parent area. When messages pass from a cluster to another cluster, they must be translated (decrypted and re-encrypted) since different TEKs are used in each cluster. Therefore, upon receiving messages from the parent area (encrypted using the parent's TEK), the root $AKD$ of the cluster , which is the entry point of the cluster, will decrypt them and re-encrypt them using the TEK used within its area before forwarding them downward to its area members. The root cluster AKD is said to be in *active* state because it decrypts messages received from parent area which belongs to a different cluster and re-encrypt them with its TEK before sending them to its area members. The internal AKDs of a cluster are considered *passive* since they forward received messages to their area members without any cryptographic transformation. The construction of clusters can be done using different heuristics presented in our previous works [Challal et al. 2008, Gharout et al. 2008, Challal et al. 2004].

When there is a membership change (*join* or *leave*) in any area, a new TEK is distributed to the overall areas of the same cluster because of using a *common* TEK.

In our solution, the mobility issue of members is treated without re-keying process since the moving member is still valid in the session. For this purpose, we propose a verification mechanism that we present in detail in subsequent sections. In conclusion, our solution deals with two main functionalities:

- Partitioning the group into clusters of areas while reducing the impact of *1-affects-n* phenomenon. Areas in the same cluster uses a common TEK.
- Mobility management in term of re-keying. We handle the mobility of members between areas.

In [Challal et al. 2008, Gharout et al. 2008, Challal et al. 2004] we show how the partitioning of the group into clusters of areas is done while considering different heuristics relating to dynamism inside the group. In [Gharout et al. 2010], we presented a mobility support for group key management protocols. The solution we present in this paper is partially based on the protocol we proposed in [Gharout et al. 2010].

### 3.2 Definitions and assumptions

Let us denote a group member by $m_l$, an area by $a_i$ and a cluster by $C_k$. In table 1 we summarize the nomenclature used through the paper.

Each member $m_l$ has a personal secrete key called Member Encryption Key ($MEK_l$). The $MEK$ is used to communicate secretly with any $AKD$ in the group. We assume that each member has a certified pair of public and private keys. All the $AKD$s share a symmetric key called Session Encryption Key ($SEK$) which is used to generate and verify Members' Keys ($MEK$s) (see section 5). This key must be known only by AKDS. Let us recall that all the areas of the same cluster $C_k$ share the same *TEK*. This is the first key that each $AKD_i$ should maintain and share with all the members of its area $a_i$. Each $AKD_i$ shares a secrete Key Encryption Key ($KEK_i$), used to encrypt new TEKs, with all its area members. We introduced KEK as a common key at the area level in order to optimize the number of re-keying messages whenever distributing a new TEK is required and the old one can not be used to encrypt the new one. In this case, the KEK will be used to encrypt and send the new TEK through a single multicast message to the corresponding area, instead of sending the new TEK encrypted with the individual keys of area members.

We assume that each $AKD_i$ maintains two lists : a *list of members* (denoted by $ListM_i$) which contains identities of current area members; and a *list of old members* (denoted $ListO_i$) which contains identities of members which were members of area $a_i$ and moved to other areas without leaving the secured group session.

We discuss in the following section how to use these keys for re-keying in cases of *join*, *leave* and *member mobility* inside the group.

| Symbol | Signification | Role |
|---|---|---|
| $TEK$ | *Traffic Encryption Key* | Used to encrypt and decrypt group messages. $newTEK$ and $oldTEK$ refers to the newly generated TEK and currently used TEK, respectively. This is a symmetric key. |
| $AKD$ | Area Key Distributor | This entity is responsible of key management inside an area. |
| $SEK$ | Session Encryption Key | This key is known only by AKDs. This is a symmetric key. |
| $KEK$ | Key Encryption Key | Used to encrypt other keys. This is a symmetric key. |
| $a_i$ | Area $i$ | a set of group members using the same $TEK$. |
| $C_k$ | Cluster of areas | A set of areas using the same $TEK$. |
| $MEK$ | Member Encryption Key | Used to encrypt messages between AKD and member. Each member $m_l$ owns its $MEK$ identified by $MEK_l$. This is a symmetric key. |
| $k_s$ | Private key | $k_{s\,l}$ is the private key of member $m_l$. This is an asymmetric key. |
| $k_p$ | Public key | $k_{p\,l}$ is the public key of member $m_l$. This is an asymmetric key. |
| $ListM_i$ | *List of members* | Contains the list of current members inside area $a_i$. |
| $ListO_i$ | *List of old members* | Contains the list of old members of area $a_i$ which moved to other areas. |
| $X_i$ | $X$ of area $a_i$ | X refers to AKD, TEK, KEK, ListM or ListO |

**Table 1:** Nomenclature.

## 4 Key distribution protocol

In our protocol, a common TEK is used by areas inside the same cluster. In order to guarantee *perfect backward and forward secrecy*, each time a membership change (*join* or *leave*) occurs in an area, our protocol KMGM generates a new TEK and distributes it in all the areas of the concerned cluster.

We distinguish four cases of re-keying:

- *join re-keying*: when a new member joins the group for the first time, a new TEK is generated and distributed to this new member and the current group members. This scenario is illustrated by action (1) of figure 1.
- *leave re-keying*: when a member leaves the group, a new TEK is generated and distributed only to the remaining members. This scenario is illustrated by action (4) of figure 1.
- *move re-keying*: when a member moves from an area to another, there is no TEK generation but the member is verified in new area. This scenario is

illustrated by actions (2) and (3) of figure 1.

- *split re-keying*: when an AKD becomes active, a new TEKs are generated for the two resulting clusters.
- *merge re-keying*: when an AKD becomes passive, a new TEK is generated for the resulting cluster.
- *periodic re-keying*: this one can be triggered by the protocol itself to refresh the TEK and empty all lists. A new TEK is generated.

In what follows, we detail the re-keying protocol for each case.

### 4.1 Join re-reying

In KMGM, when a member $m_l$ located in area $a_i$ wants to join the group, it sends a *Registration Request* (RR) message with *join* option signed with its private key $k_{sl}$ to $AKD_i$ (action (1) in figure 1). Upon receiving the RR request, $AKD_i$ verifies if $m_l$ is authorized to join the group session. If so, $AKD_i$ generates a symmetric key $MEK_l$ (section 5) and sends it to $m_l$ encrypted with the public key of $m_l$ ($k_{p_l}$). $AKD_i$ adds $m_l$ to its list of area members $ListM_i$.

To guarantee *backward secrecy*, all the AKDs of the cluster where the new member has joined should commit to a new TEK and distribute it to the members in their areas. The commitment of new TEK is discussed in section 4.5. $AKD_i$ generates a new $KEK_i$. To distribute the new TEK and KEK to the members in area $a_i$, where the new member joins the session, $AKD_i$ sends a message containing the new TEK and the new $KEK_i$ encrypted with $MEK_l$ ($\{new\_TEK, new\_KEK_i\}_{MEK_l}$) to the new member $m_l$ and multicasts the new TEK to the other members in its area, encrypted with the old TEK ($\{new\_TEK, new\_KEK_i\}_{oldTEK}$). To distribute the new TEK in the other areas $a_t$ of the cluster, each $AKD_t$ multicasts to its area $a_t$ the new TEK encrypted with the old one ($\{new\_TEK\}_{oldTEK}$).

Thereby, each join to the session induces a multicast message in each area of the cluster where the join occurred, in addition to a single unicast message to the new member.

An intruder cannot have access to the new TEK, and hence to the content, because it cannot know neither the old TEK (which is not compromised in case of a join), nor the secret key $MEK_l$ of the new member.

### 4.2 Mobility re-keying

A member can move from an area $a_i$ to another visiting area $a_j$. We distinguish two kinds of member mobility :

1. Intra-cluster mobility: the node moves from area $a_i$ to area $a_j$ which is located in the same cluster (action (2) in figure 1).

2. Inter-cluster mobility: the node moves from area $a_i$ to area $a_j$ which is located in a different cluster (action (3) in figure 1).

When area $a_i$ and area $a_j$ belongs to the same cluster (action (2) in figure 1), area $a_i$ and area $a_j$ use the same $TEK$. In our solution, $AKD_j$ has just to verify if the member $m_l$ is a valid and really comes from area $a_i$. When area $a_i$ and area $a_j$ are located in different clusters, area $a_i$ and area $a_j$ use different TEKs (action (3) in figure 1). In this case, besides verification $AKD_j$ sends to $m_l$ the TEK used in its area.

When $m_l$ moves from area $a_i$ to area $a_j$ the following operations are then executed:

– The member $m_l$ sends to $AKD_j$ a *Registration Request* (RR) with *move* option signed with $MEK_l$;

– $AKD_j$ verifies that $m_l$ is a valid member and really comes from area $a_i$ (see how in section 5);

– if the verification succeeds, $AKD_j$ adds the *visitor* member $m_l$ to its *list of members* $ListM_j$;

– $AKD_i$ removes $m_l$ form its list of members $ListM_i$ and adds it to its *list of old members* $ListO_i$.

We notice that when there is a move in the same cluster the TEK is not renewed since the same $TEK$ is used in old and new areas, and the member is still valid. Even though the member changed a cluster, no new $TEK$ was renewed because the member is still valid in the group session. Old AKD ($AKD_i$) has just to keep trace that $m_l$ was a member in its area $a_i$ so that $AKD_i$ updates the TEK whenever $m_l$ leaves the group definitively.

The keys that $m_l$ receives in the new area $a_j$ are:

1. Intra-cluster mobility ($TEK_i = TEK_j$): $AKD_j$ sends $KEK_j$ to $m_l$ encrypted with $MEK_l$ .

2. Inter-cluster mobility ($TEK_i \neq TEK_j$): $AKD_j$ sends $KEK_j$ and $TEK_j$ to $m_l$ encrypted with $MEK_l$ .

We conclude that in our solution the TEK is not renewed when a member moves form an area to another even though the two areas are located in different clusters.

### 4.3 Leave re-keying

A valid member $m_l$ located in area $a_i$ can be expelled from the group by $AKD_i$ or can intentionally leave the group by sending a *leave* message to the $AKD_i$. In this case, all the $AKD$s of the same cluster where the *leave* occurs should commit to a new TEK and distribute it to the members in their areas, in order to guarantee *forward secrecy*.

The departing member might visit other areas inside the same cluster or other clusters. So, $m_l$ knows all $KEK$s and $TEK$s used in previously visited areas. Thereby, the TEK must be updated in each cluster $C_k$ where $m_l$ was member since it can know $TEK$s and $KEK$s used in previously visited areas of these clusters.

**Lemma 1.** *Let $TEK_k$ be the TEK used in cluster $C_k$ and $m_l$ an expelled member from area $a_i$. We have*

$$\forall C_k, \exists a_j \subset C_k / m_l \in ListO_j \Rightarrow TEK_k \textit{ is compromised} \tag{1}$$

*Proof*

When $m_l$ moves from an old area to a new area, it receives the key material (TEK, KEK) used in the new area but the TEK used in old area is not changed (section 4.2). Thus, $m_l$ knows all $TEK$s and $KEK$s of areas it has already visited. Suppose $E = \{a_1, a_2, \ldots, a_j\}$ the set of areas that $m_l$ has already visited. Now, when $m_l$ is expelled from $a_i$ it can be able to decrypt message destined to $E$ since it knows $KEK$s and $TEK$s of areas in $E$.
■

When $m_l$ is expelled from $a_i$, other $AKD$s are informed that $m_l$ has been expelled from the group (section 4.5) and thereby new $TEK$s are generated in clusters where equation 1 is verified. To distribute the new TEK in area $a_i$, $AKD_i$ sends to each member $m_n$ $(n \neq l)$ in its area the new TEK and the new $KEK_i$ encrypted with the secret key $MEK_n$ of member $m_n$ $(AKD_i \rightarrow m_n\ (n \neq l) : \{new\_TEK, new\_KEK_i\}_{MEK_n})$. To distribute the new TEK in the other areas $a_t$ (not visited by the expelled member)of the cluster where the leave occurs, each $AKD_t$ multicasts to its area $a_t$ a message containing the new TEK encrypted with $KEK_t$ $(\{new\_TEK\}_{KEK_t})$. $AKD_i$ removes $m_l$ from $ListM_i$ **and empties** $ListO_i$.

Therefore, each remaining valid member $m_n$ in $a_i$ would be able to decrypt the new TEK and the new $KEK$ using its secret key $MEK_n$, and the expelled member $m_l$ will be prevented from having access to these new TEK and $KEK$, because its associated key $MEK_l$ would have not been used to encrypt them.

For each cluster $C_k$ where equation 1 is verified, to distribute the new TEK' in each area $a_p \subset C_k$ where $m_l \in ListO_p$, $AKD_p$ sends to each member $m_n$ $(n \neq l)$

in its area the new TEK' and the new $KEK_p$ encrypted with the secret key $MEK_p$ of member $m_n$ ($AKD_p \rightarrow m_n$ $(n \neq l) : \{new\_TEK', new\_KEK_p\}_{MEK_n}$). To distribute the new TEK in the other areas $a_q$ in the cluster $C_k$, each $AKD_q$ multicasts to its area $a_q$ a message containing the new TEK encrypted with $KEK_q$ ($\{new\_TEK'\}_{KEK_q}$). Each $AKD_p$ **empties** $ListO_p$.

We notice, that when there is a *leave* re-keying, *lists of old members (ListO)* of area $a_i$ and each area $a_p$ ($a_p/m_j \in ListO_p$) are emptied. Indeed, there is no need to treat further departures of these old members since their *Member Encryption Keys* ($MEK$s) are not used in the re-key message. Thereby, they will not be able to know new KEK and TEK of area $a_i$ and each area $a_p$.

An intruder cannot have access to the new TEK, and hence to the content, because it cannot know any secret key $MEK_n$ of a remaining legitimate member, and cannot know any $KEK$ of the other areas it had not visited. Note that, in case of an expulsion, we cannot use the old TEK to encrypt the new one because it is considered compromised since it is known by the expelled member. Since $m_l$ do not know $KEK$s of other areas it had not visited, we use in these areas $KEK$s to send the new TEK. This optimizes number of re-keying messages.

### 4.4 Periodic re-keying

In our solution, a batch re-keying can be envisaged to refresh $TEK$s. This can be done periodically or triggered by an $AKD$. An $AKD_i$ can trigger a Re-Keying if the size of its *list of old members* ($ListO_i$) is full enough. The cluster root can also have the role to schedule batch re-keying.

When a batch re-keying is done in a cluster $C_k$, the new TEK is sent as in *leave* case since the members inside the *list of old members (ListO)* know the current TEK and KEK of concerned areas even though they are not members of these areas. We distinguish two kinds of areas: areas where *list of old members (ListO)* is empty and areas where this list (ListO) is not empty. To distribute the new TEK in each area $a_i$ where $ListO_i$ is not empty, $AKD_i$ sends to each area member $m_n$ ($m_n \notin ListO_i$) in its area the new TEK and the new $KEK_i$ encrypted with the secret key $MEK_n$ of member $m_n$. Each $AKD_i$ empties $ListO_i$. To distribute the new TEK in the other areas $a_t$ of the cluster $C_k$ where $ListO_t$ is empty, each $AKD_t$ multicasts to its area $a_t$ a message containing the new TEK encrypted with $KEK_t$ ($\{new\_TEK\}_{KEK_t}$).

The advantage of batch re-keying is that it allows to increase the robustness of $TEK$s by refreshing them and to empty *lists of old members* in the cluster. This decreases the number of key updates (see section 4.3) in further group membership changes.

### 4.5 TEK generation

When an $AKD_i$ needs to renew the TEK inside its area due to membership change or a triggered re-keying, it sends a *re-key request* to the cluster root asking it to generate a new TEK for the whole cluster. If the membership change type is a *leave*, $AKD_i$ must include the identity of the expelled member $m_l$ in the request. The cluster root generates a new TEK and sends it to its area members. If the type of the membership change is a *leave* the cluster root must include the identity of the expelled member in the re-key message. The cluster root $AKD$ informs other AKDs of the system that $m_l$ has been expelled ($\{Expelled(m_l), \ldots\}_{SEK}$). Upon receiving, the message $Expelled(m_l)$ each $AKD_j$ checks if $m_l \in ListO_j$. If it is the case, it sends a key request to its cluster root $AKD$ with *leave* option but it informs it that $m_l$ is an external member (message 2 below). This last indication is important to avoid that the cluster root $AKD$ informs other $AKD$s about this leave. A cluster root AKD informs other $AKD$s about a member expulsion or leave only if this expulsion occurs in its cluster.

The following is the format of the re-key request message sent by an $AKD_i$, which can be signed with its private key to prove authentication, to the cluster root $AKD_r$:

$$AKD_i \rightarrow AKD_r : \{option, status, member\_id, ...\} \tag{2}$$

In formula 2, the field *option* can be *join*, *leave* or triggered. The field *status* can have two status: *intern* which means that the membership change occurs in current area; or *extern* which means that the membership change occurs in different area but compromises key materials in current area when the expelled member is present in the *list of old members* of area $a_i$ ($ListO_i$).

If the cluster root $AKD$ receives other re-key requests with *leave* option and *extern* status caused by a member $m_l$, it verifies if it has already treated a similar request caused by this member. If it is the case it ignores the request. The cluster root AKD sends the new TEK encrypted with the old one. But when a member leaves this key can not be used to encrypt the new one. In this case, $AKD_r$ can use either the MEK of each AKD since each AKD is considered as a member or the SEK of the current session.

### 4.6 Split re-keying

When $AKD_i$ takes the decision to become *active* (a split operation), it creates a new cluster. Therefore, $AKD_i$ generates a new TEK and distributes it downward encrypted with $KEK_i$ ($\{newTEK\}_{KEK_i}$). When the child agents in the so formed cluster receive this message they forward it downward, encrypted with their respective KEK, until it hits *active* agents. Since the old TEK becomes obsolete, the root of the old cluster will also re-distribute a new TEK to ensure forward and backward secrecy in the same way. Afterward, $AKD_i$ assures the translation of received upward messages into the TEK used in its new cluster.

### 4.7 Merge re-keying

When $AKD_i$ takes the decision to become *passive* (a merge operation), it will use the same TEK of its parent $AKD_j$. Since it is a member in its parent area, it knows the TEK used in its parent cluster, and thereby distributes it downward encrypted with the old TEK previously used in its cluster ($\{parentTEK\}_{oldTEK}$). When the downward child agents receive this message they forward it downward until it hits *active* agents. Afterward, the role of $AKD_i$ will be limited to forward received upward messages.

## 5 Mobility management

Recall that all $AKD$s share symmetric key $SEK$ which is known only by $AKD$s. We assume that this key is generated by the AKD where the traffic source is connected at session setup, and distributed securely to other AKDs. Thus, we will have different $SEK$s for each group session.

When a new member $m_l$ located in area $a_i$ joins the group it sends a *Registration Request* ($RR$) message to $AKD_i$ encrypted with its private key. Thus, $AKD_i$ generates a *member encryption key* $MEK_l$ using formula 3 and sends it to $m_l$ encrypted with the public key of $m_l$ ($k_{p_l}$). To generate the $MEK$ of member $m_l$ $AKD_i$ uses a *Key Derivation Function* (KDF) as follows:

$$MEK_l = KDF(SEK, data_l) \tag{3}$$

where,

- $SEK$ = The Session Key shared which is only known by $AKD$s.
- $data_l = label_l + optional\_data$ ("+" denotes concatenation).
- $label_l$ = Contains member $m_l$ identity, the public key of $m_l$ and eventually other information concerning this member.
- $optional\_data$ = This field can be empty or contains some information about the concerned member or the group session.

Note that the MEK is valid only in current session since $SEK$ depends on session information. A member will have different MEKs if it participates to many secured group sessions. Thereby, if a member is expelled from a session it can receive messages in other sessions.

As in [Kelly and Frankel 2007], the KDF used is a PRF (*Pseudo Random Function*) which is a combination of one way hash functions (MD5, HMAC, SHA1, SHA256,..). Several PRFs can be used in the *Key Derivation Function*. However, in order to have a coordinated key derivation function the same PRF function must be used by all the $AKD$s. Some popular PRFs based on HMAC-SHA1, HMAC-MD5 , HMAC-SHA256 used in [Kelly and Frankel 2007] are recommended.

The $SEK$ key which is known only by $AKD$s plays a key role in $MEK$s generation. Thus, each $AKD$ is able to verify the $MEK$ presented by a user as follows: When $m_l$ moves from area $a_i$ to area $a_j$ it sends a *Registration Request* with *move* option to $AKD_j$ and presents its credentials such as a certified public key. According to formula 3, $AKD_j$ calculates $MEK'_l$ with $m_l$ credentials. If $(MEK'_l = MEK_l)$ then $m_l$ is accepted in area $a_j$ and receives the $TEK$ in area $a_j$.

## 6 Simulation and results

In this section, we present our simulation model and some results of the carried out simulations in which we compared KMGM with other protocols from literature: Iolus [Mittra 1997] which is a decentralized approach with *independent TEK per subgroup*,the *Group Key Management Protocol* (GKMP) [Harney and Muckenhirn 1997] which is a centralized solution with *common TEK* for the whole group and M-IOLUS which is a new version of Iolus for mobile environments. We also used in simulation our protocol ASGK which is a native version of KMGM but it does not support mobility. In our simulation a re-keying is done each time there a membership change (*join* or *leave*) in the group. In the protocols which not support mobility (GKMP, Iolus and ASGK) we consider a member *move* as a *leave* in the old area and a *join* in the new area. We study the *1 affects n* behavior of each simulated protocol, the number of decryption / re-encryption operations required for the communication and the number of re-keying messages. For the number of re-keying messages we don't make a difference between a unicast message and a multicast message. We focus our evaluation on the number of these messages.

### 6.1 Simulation model

In our simulations, we use a KMGM hierarchy composed of 5 subgroups presented in figure 1. Almeroth et al. showed in [Almeroth and Ammar 1996, Almeroth and Ammar 1997] that the dynamism of some multicast sessions over the MBone (Multicast Backbone) can be modeled as follows: the users arrive in a multicast group according to a *Poisson* process with rate $\lambda$ (arrivals/time unit), and the membership duration of a member in the group follows an exponential distribution with a mean duration $\frac{1}{\mu}$ time units. In our case, we apply this model to each area. Unless specified otherwise, we considered a session of 3 hours, where members arrive following a *Poisson* law with an average *inter-arrival* equal to 20 seconds, an *inter-move* equal to 15 seconds and remain in the session 30 minutes in average. After each $\theta = 15min$, the area key distributors reconsider the dynamism information and decide to become *active* or *passive* using the heuristic described in [Challal et al. 2008]. The affectation of arriving members to the different subgroups is random and follows a uniform distribution with percentages varying over time. In table 2, we illustrate the percentage distribution of arrivals to the different subgroups during the whole session.

For mobility scenario (figure 1), we first choose the leaving area using a uniform distribution, so that, each area has the same probability of selection. After that, a destination area, unlike leaving one, is chosen using the same law. Finally, the mobile member is chosen from the leaving area. All the current members have the same selection probability independently from their joining date. The selection probabilities of areas and members during the mobility are then equiprobable.

| *Period* | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 0h-1h | 20% | 20% | 40% | 15% | 5% |
| 1h-2h | 40% | 20% | 20% | 15% | 5% |
| 2h-3h | 15% | 20% | 5% | 40% | 20% |

**Table 2:** Percentage distribution of arrivals to the different subgroups.

## 6.2 Simulation results

### 6.2.1 Impact of the group size

In a first stage, we were interested in the size scalability of KMGM compared to the other approaches in what relates to *1-affects-n* and decryption /re-encryption overheads which is a computation overhead. Two parameters of our simulation model control the size of the group: the *average inter-arrival* of members into the session, and the *average membership duration* of the members in the session. In figures 2 and 3 we varied the average inter-arrival from 1 second to 65 seconds and we measured the *1-affects-n*, the decryption / re-encryption overheads and the number of re-keying messages, respectively.

The smallest values of the inter-arrival correspond to the largest sizes of the group. With the GKMP protocol, all the members are affected and hence

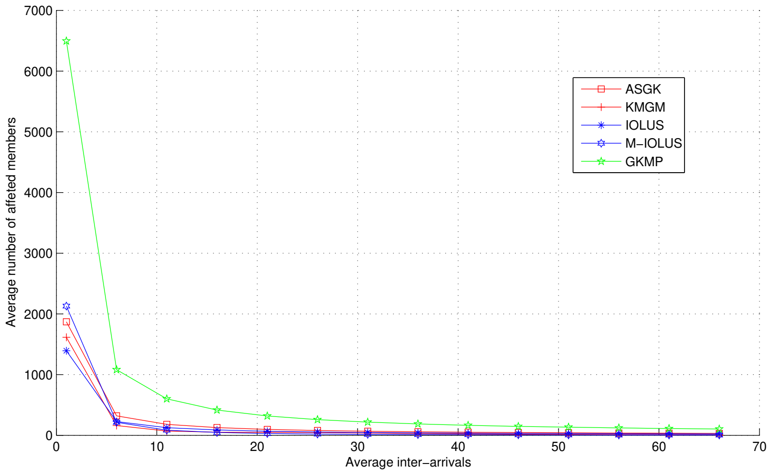


**Figure 2:** Impact of inter-arrival variation on *1-affects-n* overhead.

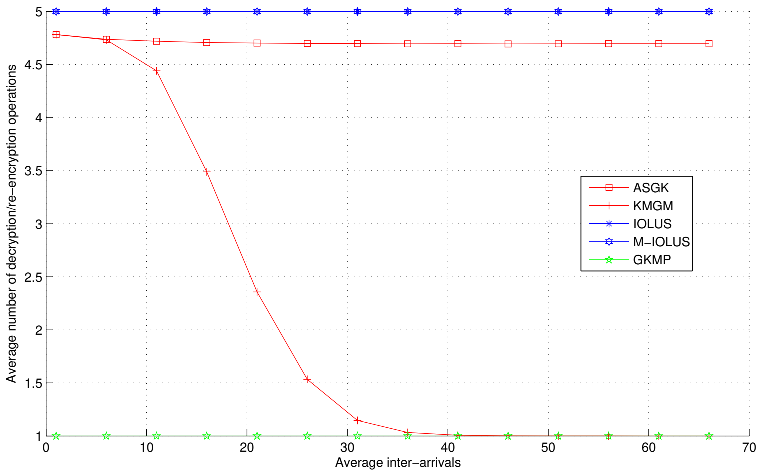


Figure 3: Impact of inter-arrival variation on decryption / re-encryption overhead (same behaviour for IOLUS and M-IOLUS).

suffers from the *1-affects-n* phenomenon. Typically, we notice in figure 2 that for the inter-arrival in $[1s : 5s]$, the number of affected members for the GKMP

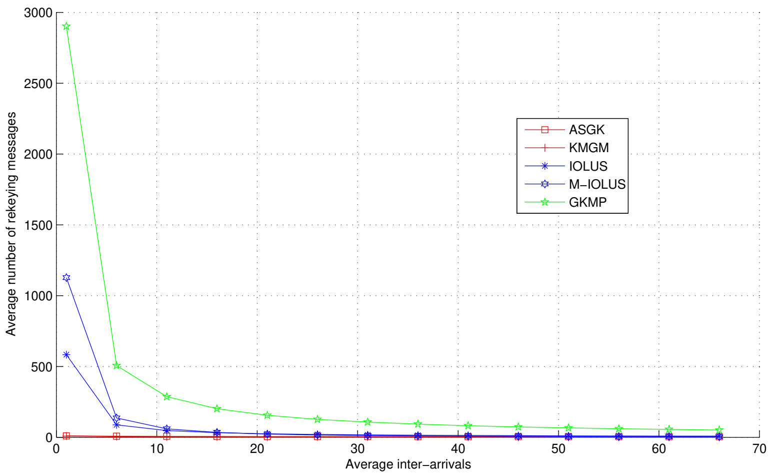


**Figure 4:** Impact of inter-arrival variation on re-keying overhead.

protocol is bigger. For the same range of inter-arrival values, Iolus, M-IOLUS and KMGM reduce the number of affected members to the minimum. Therefore, our approaches and M-Iolus scale better to large and dynamic groups. Moreover, notice in figure 3 that when the inter-arrival increases, and hence the group size decreases, Iolus and M-Iolus performs always the same number of decryption / re-encryption operations (5 operations). We refer by operations to the number of encryption areas. Indeed, for Iolus and M-Iolus this number is always equals to 5 the number of areas in the group. However, KMGM reduces this overhead while maintaining low *1-affects-n* overhead. For the smallest groups, KMGM reaches the same performance of the GKMP protocol in performing a single encryption at the source and decryption at receivers. In figure 4, we can easily see that the average number of re-keying messages in KMGM is smaller in other solutions. This is due to the fact when a member leaves the group the list of old members are emptied in all areas that the expelled member has previously visited. Thereby, in further departures, these areas will not be affected since their TEKs have been updated.

We can notice the same phenomenon by varying the average membership duration of the members in the session. In figure 5, we remark that KMGM and M-Iolus scale better to large groups. We remark also that KMGM scales better than the GKMP protocol.

In figure 6, we remark again that KMGM has the advantage over KMGM and M-Iolus of decreasing the decryption / re-encryption without dramatically

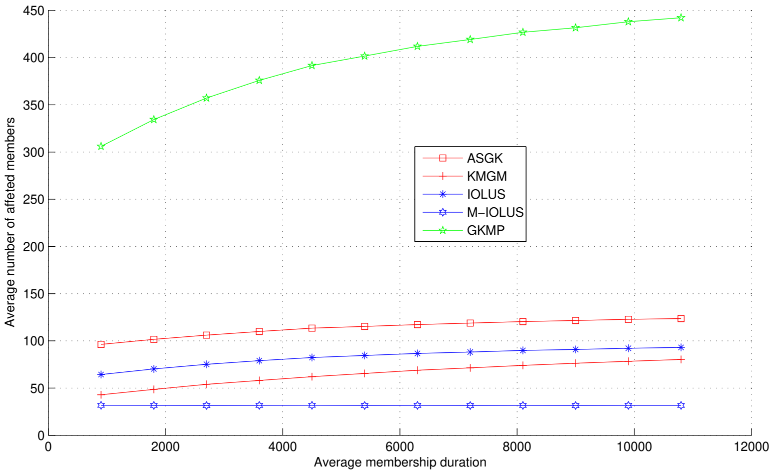


**Figure 5:** Impact of membership duration variation on *1-affects-n* overhead.

increasing the *1-affects-n* overhead. When membership duration increases, $SM$s have tendency to be passive, and thus the number of decryption/re-encryption operations decreases.

The same behavior of KMGM can be seen in figure 7 where the average number of re-keying messages is smaller comparing to other solutions.

#### 6.2.2 Impact of the mobility

In a second stage, we were interested by the impact of mobility members inside the group. We have varied inter-moves from 1 second to 15 seconds in order to have a high mobility scenario. In figure 8, we can see that for smallest values of inter-moves ([1s,5s]), which corresponds to a high mobility, the average number of affected members in KMGM and M-Iolus is smaller than other protocols because they are not affected by mobility of members between areas. This number is few greater in KMGM because the size of affected areas is greater in KMGM. Indeed, for the same interval we see in figure 9 that the number of encryption areas is smaller in KMGM comparing to M-Iolus. This means that areas are merged to use the same TEK. However, in figure 10, we see that the average number of re-keying messages in KMGM is very smaller compared to M-Iolus and other protocols. This is due to the fact that list of old members are emptied when there is a leave. So, the corresponding members will not induce re-keying in future membership changes. We know that the great number or re-keying

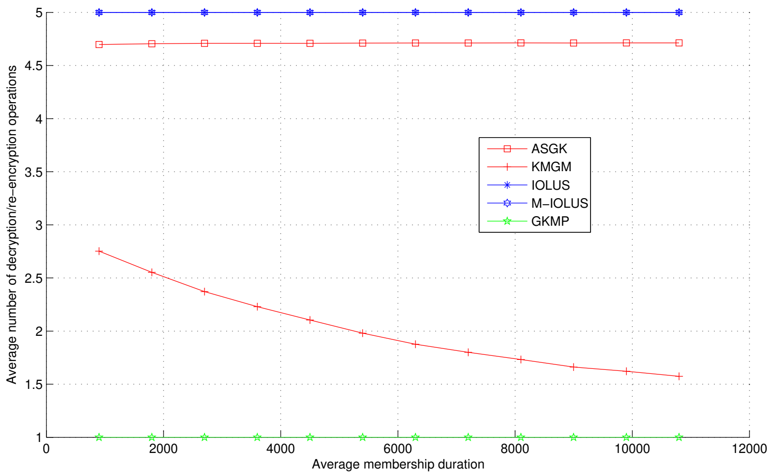


Figure 6: Impact of membership duration on decryption / re-encryption overhead (same behaviour for IOLUS and M-IOLUS).

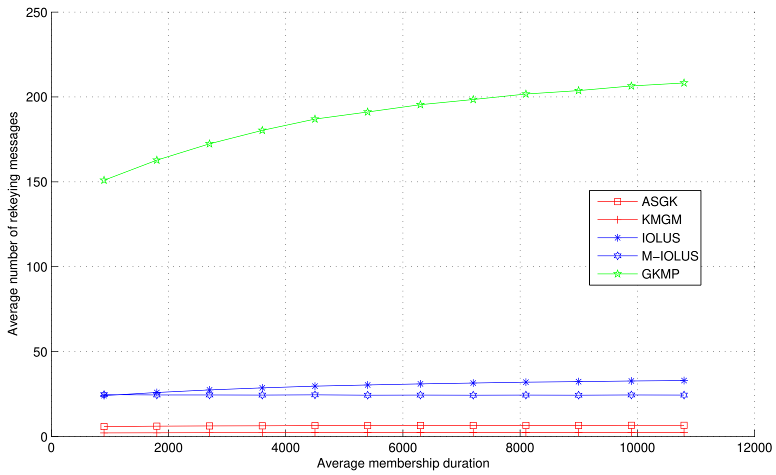


**Figure 7:** Impact of membership duration on re-keying overhead.

messages is generated in case of leave (cf. section 4.3).

We conclude that KMGM deals better with members mobility thanks to

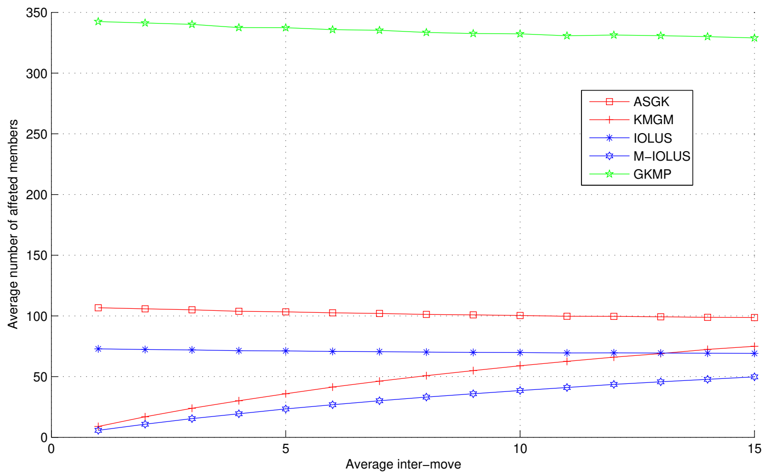


**Figure 8:** Impact of inter-move variation on *1-affects-n* overhead.

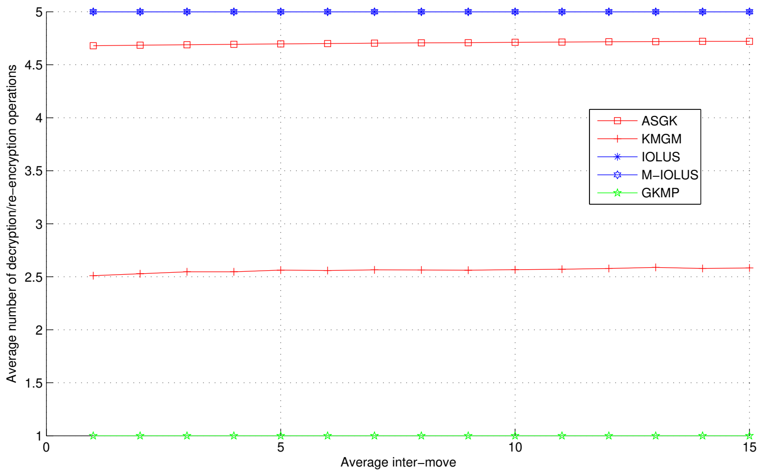


Figure 9: Impact of inter-move variation on decryption / re-encryption overhead (same behaviour for IOLUS and M-IOLUS).

reducing both the number of affected members and the number of re-keying messages. Our protocols is suitable to dynamic groups and wireless mobile en-

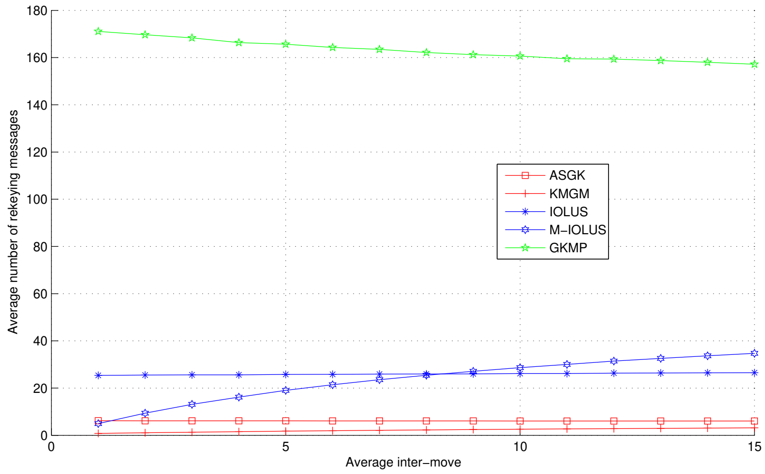


Figure 10: Impact of inter-move variation on decryption / re-encryption overhead.

vironments where there is a high members' mobility.

## 7 Conclusion

Several group key management protocols have been proposed to minimize the overall overhead in a scalable and secure manner. However, all the approaches have some shortcomings in handling group dynamism. The centralized schemes suffer from the *1-affects-n* phenomenon especially if the group membership is highly dynamic. The distributed nature of group-oriented services has a huge impact on security efficiency. On one hand, a group distribution tree can span large networks where members may be tremendously distant from each other. On the other hand, the efficiency of group security mechanisms can be severely affected by some phenomena that depend on their occurrence location in the network. Besides, in mobile environments the mobility of members between areas and subnets complicates key management and authentication in the group. Given this group communication feature, we proposed in this paper a group key management protocol to secure group communication in mobile environments where we treat members dynamism and mobility with minimizing re-keying messages. Indeed, we proposed a mechanism that avoids to renew the TEK when the member moves from an area to another. Simulation results showed that our protocols achieves a better performance trade-offs comparing to other protocols.

## References

[Almeroth and Ammar 1996] K. Almeroth and M. Ammar. Collecting and Modeling the Join/Leave Behavior of Multicast Group Members in the MBone. In *HPDC '96: Proceedings of the High Performance Distributed Computing (HPDC '96)*, pages 209–216, Washington, DC, USA, 1996. IEEE Computer Society.

[Almeroth and Ammar 1997] K. Almeroth and M. Ammar. Multicast group behaviour in the internet's multicast backbone (Mbone). *IEEE Communications*, 35(6):124–129, 1997.

[Baugher et al. 2005] M. Baugher, R. Canetti L. Dondeti, and F. Lindholm. RFC4046: Multicast Security (MSEC) Group Key Management Architecture. *IETF RFC 4046*, April 2005.

[Challal et al. 2004] Y. Challal, H. Bettahar, and A. Bouabdallah. SAKM: a scalable and adaptive key management approach for multicast communications. *ACM SIGCOMM Computer Communications Review*, 34(2):55–70, 2004.

[Challal et al. 2008] Y. Challal, S. Gharout, A. Bouabdallah, and H. Bettahar. Adaptive clustering for Scalable Key Management in Dynamic Group Communications. *Inderscience International Journal of Security and Networks (IJSN)*, 3(2), 2008.

[Cao et al. 2006] J. Cao, L. Liao, and G. Wang. Scalable key management for secure multicast communication in the mobile environment. *Pervasive and Mobile Computing*, 2(2), April 2006.

[Challal et Seba 2005] Y. Challal and H. Seba. Group Key Management Protocols: A Novel Taxonomy. *Enformatika, International Journal of Information technology*, 2(1), 2005.

[Gharout et al. 2010] S. Gharout, A. Bouabdallah, M. Kellil, and Y. Challal. Key management with host mobility in dynamic groups. In *Proceedings of the 3rd international conference on Security of information and networks*, SIN '10, pages 186–194, New York, NY, USA, 2010. ACM.

[Gharout et al. 2008] S. Gharout, Y. Challal, and A. Bouabdallah. Scalable Delay-constrained Multicast Group Key Management. *International Journal of Network Security (IJNS)*, 7(2):153–167, September 2008.

[Hardjono et al. 2000] T. Hardjono, B. Cain, and I. Monga. Intra-domain Group Key Management for Multicast Security. *IETF Internet draft*, September 2000.

[Harney and Muckenhirn 1997] H. Harney and C. Muckenhirn. RFC2093: Group Key Management Protocol (GKMP) Architecture. *IETF RFC 2093*, July 1997.

[Haberman and Martin 2008] B. Haberman and J. Martin. RFC5186: Internet Group Management Protocol Version 3 (IGMPv3) / Multicast Listener Discovery Version 2 (MLDv2) and Multicast Routing Protocol Interaction . *IETF RFC 5186*, May 2008.

[Hernandez-Serrano et al. 2005] J. Hernandez-Serrano, J. Pegueroles, and M. Soriano. GKM over large MANET. *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks (SNPD/SAWN'05)*, 2005.

[Judge and Ammar 2003] P. Judge and M. Ammar. Security Issues and Solutions in Multicast Content Distribution: A Survey. *IEEE Network*, 17(1):30–36, January/February 2003.

[Johnson et al. 2004] D. Johnson, C. Perkins, and J. Arkko. RFC3775: Mobility Support in IPv6. *IETF RFC. Status: Proposed Standard.*, June 2004.

[Kelly and Frankel 2007] S. Kelly and S. Frankel. RFC4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec . *IETF RFC 4868*, May 2007.

[Mat Kiah and Martin 2007] M. L. Mat Kiah and K. M. Martin. Host Mobility Protocol for Secure Group Communication in Wireless Mobile Environments. In *FGCN '07: Proceedings of the Future Generation Communication and Networking*, pages 100–107, Washington, DC, USA, 2007. IEEE Computer Society.

[Mat Kiah and Martin 2008] M. L. Mat Kiah and K. M. Martin. Host Mobility Protocol for Secure Group Communication in Wireless Mobile Environments. *International Journal of Security and its Applications*, 2(1):39–52, January 2008.

[Koodli 2009] R. Koodli. RFC5568: Fast Handovers for Mobile IPv6 . *IETF RFC. Status: Proposed Standard.*, July 2009.

[Kamat et al. 2003] S. Kamat, S. Parimi, and D. P. Agrawal. Reduction in Control Overhead for a Secure, Scalable Framework for Mobile Multicast. *IEEE International Conference on Communications*, 1:98 – 103, May 2003.

[Mittra 1997] S. Mittra. Iolus: a framework for scalable secure multicasting. In *SIGCOMM '97: Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 277 – 288, New York, NY, USA, 1997. ACM Press.

[Perkins 2002] C. Perkins. RFC3344: IP Mobility Support for IPv4. *IETF RFC. Status: Proposed Standard.*, August 2002.

[Di Pietro et al. 2002] R. Di Pietro, L. V. Mancini, and S. Jajodia. Efficient and secure keys management for wireless mobile communications. In *POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing*, pages 66–73, New York, NY, USA, 2002. ACM.

[Rafaeli and Hutchison 2003] S. Rafaeli and D. Hutchison. A Survey of Key Management for Secure Group Communication. *ACM Computing Surveys*, 35(3):309–329, September 2003.

[Romdhani et al. 2004] I. Romdhani, M. Kellil, H-Y. Lach, A. Bouabdallah, and H. Bettahar. IP Mobile Multicast: Challenges and Solutions. *IEEE Communications Surveys & Tutorials*, 6(1), 2004.

[Roh and Lee 2006] J-H. Roh and K-H. Lee. Key management scheme for providing the confidentiality in mobile multicast. *The 8th International Conference Advanced Communication Technology. ICACT.*, 2:5, Februray 2006.

[Savola 2008] P. Savola. RFC5110: Overview of the Internet Multicast Routing Architecture. *IETF RFC 5110*, January 2008.

[Wong et al. 2000] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM Transactions on Networking*, 8(1):16–30, February 2000.

[Wallner et al. 1999] D. Wallner, E. Harder, and R. Agee. RFC2627: Key Management for Multicast : Issues and Architecture. *IETF RFC 2627*, June 1999.

[Yeo et al. 2004] C.K. Yeo, B.S. Lee, and M.H. Er. A survey of application level multicast techniques. *Computer Communications*, 27(15):1547–1568, September 2004.

[Zhang et al. 2002] C. Zhang, B. DeCleene, J. Kurose, and D. Towsley. Comparison of inter-area rekeying algorithms for secure wireless group communications. *Perform. Eval.*, 49(1-4):1–20, 2002.

[Zhu and Jajodia 2004] S. Zhu and S. Jajodia. Scalable group rekeying for secure multicast: A survey. *In Proc. 5th International Workshop on Distributed Computing*, 2918:1–10, 2004.