Gestion de Clé de Groupe Adaptative avec Support de Mobilité

Saïd Gharout, Abdelmadjid Bouabdallah, Yacine Challal

Gestion de Clé de Groupe Adaptative avec Support de Mobilité

Saïd GHAROUT (sgharout@hds.utc.fr)*
Abdelmadjid BOUABDALLAH (bouabdal@hds.utc.fr)*
Yacine CHALLAL (ychallal@hds.utc.fr) *

Abstract: Nous assistons ces dernières années à l'émergence d'applications destinées à un groupe d'utilisateurs dans divers domaines. Ceci est rendu possible grâce à une technique de routage appelée le routage multicast. Cependant, le manque de sécurité empêche le déploiement à grande échelle de ces applications, pour lesquelles la demande ne cesse pas d'augmenter chez les fournisseurs de services Internet et les distributeurs du contenu multimédia. La nature dynamique des sessions multicast complique le service de confidentialité. Ce dernier doit assurer que uniquement les membres légitimes ont accès aux messages destinés au groupe. En effet, chaque changement d'adhésion induit une redistribution d'une nouvelle clé de chiffrement du trafic à tous les membres légitimes. Cette redistribution induit une charge supplémentaire connue sous le nom du phénomène 1-affecte-n. Pour faire face à ce coût, d'autres solutions organisent le groupe en sous-groupes mais elles présentent un nouveau défi qui est le besoin de la traduction du flux à chaque fois qu'il passe d'un sous-groupe à un autre. Ceci est considéré comme un inconvénient pour les applications qui exigent une transmission en temps réel telle que la vidéo-conférence. Dans ce papier, nous proposons une nouvelle approche adaptative pour la gestion de clé de groupe qui tient en compte l'aspect dynamique du groupe. Notre approche décentralisée, isole les zones dynamiques du reste du groupe pour utiliser des clés de trafic différentes. Nous proposons aussi un mécanisme de vérification des membres lorsqu'ils sont mobiles au sein du groupe pour éviter de re-générer la clé du groupe. Les résultats de simulation montrent de meilleures performances pour notre approche par rapport à d'autres solutions de gestion de clé.

Keywords: Sécurité, Gestion de clé, Communication de groupe, Dynamisme, Mobilité

1 Introduction

Le vaste déploiement de l'Internet a contribué au développement de nouvelles applications qui requièrent des services de communication multipoint (multicast ou communication de groupe). Une liste non exhaustive d'applications destinées au group d'utilisateurs comprend les conférences multimédia, le télé-enseignement, les jeux vidéo distribués, le travail coopératif, la vidéo à la demande, etc. Une des briques de base de ce mode de communication est le routage multicast (multicast)[Dee88] qui a rendu la communication multicast un moyen efficace pour envoyer des données aux membres d'un groupe. Le multicast permet la distribution des données à grande échelle en offrant un mécanisme de transport efficace

* Lab. Heudiasyc UMR 6599. Université de Technologie de Compiègne. Centre de Recherches de Royallieu. BP 20529.60205 COMPIEGNE cedex FRANCE
Tel : 33 (0) 3 44 23 46 45 - Fax : 33 (0) 3 44 23 44 77

pour les communications un-à-plusieurs et plusieurs-à-plusieurs. Au cours des dernières années, le multicast a été le sujet de beaucoup de travaux de recherche, d'ingénierie, et d'efforts de déploiement. L'ensemble des ces efforts ont pu et continue à transformer le multicast en une technologie utilisée par plusieurs applications et domaines [JA03].

Mais plusieurs types d'applications utilisent les communications multicast nécessitent un certain niveau de sécurité : authentification, intégrité, confidentialité et contrôle d'accès. Comme les techniques implémentant ces services dans les communications point-à-point ne peuvent être appliquées telles quelles aux communications de groupe, la sécurité des communications de groupe a fait l'objet de plusieurs travaux et continue à être un sujet d'actualité pour les chercheurs et les gouvernements.

La confidentialité des communications de groupe nécessite à ce qu'uniquement les membres valides peuvent avoir accès au contenu destiné au groupe. Généralement, une clé symétrique est utilisée pour chiffrer les donnés par la source du trafic et les déchiffrer aux niveau des récepteurs. Cette clé est appelée Clé de Chiffrement de Trafic (TEK, *Traffic Encryption Key*). De plus, dans les environnements mobiles lorsque l'utilisateur se déplace d'une zone (qui peut être un sous-réseau ou un autre point d'accès au réseau) à une autre, il devra pouvoir recevoir les données à partir de la nouvelle zone. Ainsi, pour sécuriser les communications dans le multicast mobile, le protocole doit traiter non seulement l'adhésion dynamique des membres au groupe mais aussi avec la localisation dynamique des membres [RKL+04]. Généralement, lorsque un membre se déplace d'une zone à une autre ceci est considéré comme un départ de l'ancienne zone et comme une adhésion dans la nouvelle zone. Ce qui implique deux mises-à-jour de clé (dans l'ancienne et dans la nouvelle zone).

Les besoins de confidentialité au sein du groupe dans un environnement mobile peuvent être résumés avec ces trois règles de gestion de clé de groupe:

- Confidentialité Passée (*Backward Secrecy*): Consiste à garantir qu'un intrus qui connaît un ensemble de clés ne peut déduire les clés antérieures de groupe. Cette propriété permet d'assurer qu'un nouveau membre du groupe ne peut déchiffrer les messages envoyés au groupe avant son adhésion.

- Confidentialité Future (*Forward Secrecy*): Consiste à garantir qu'un intrus qui connaît un ensemble d'anciennes clés de groupe, ne pourra pas déduire les nouvelles clés de groupe. Cette propriété permet d'assurer qu'un ancien membre ne pourra pas déchiffrer les messages destinés au groupe après l'avoir quitté.

- Mobilité des utilisateurs (*Host Mobility*): celle-ci augmente la complexité de la gestion de la clé au sein du groupe. Un membre mobile se déplaçant d'une zone à une autre peut compromettre la clé de chiffrement du trafic et donc causer la génération d'une nouvelle.

Pour satisfaire à ces besoins, une redistribution de clé doit être effectuée à chaque fois qu'un membre adhère ou quitte le groupe. Ceci consiste en la génération d'une nouvelle TEK et la distribuer uniquement aux membres valides du groupe en incluant le nouveau membre dans le cas d'une adhésion et en l'excluant dans le cas d'un départ ou expulsion. Ce processus permet d'assurer qu'un nouveau membre ne pourra pas déchiffrer les messages envoyés avant son adhésion et qu'un ancien membre ne pourra pas déchiffrer les messages envoyés après son départ. Le plus grand problème des mécanismes de redistribution de clé

est la scalabilité (mise à l'échelle): comme le processus de re-distribution de clé (*re-keying*) devrait être déclenché à chaque fois qu'il y ait un changement d'adhésion (*Join* ou *Leave*), le nombre de messages de mises à jour de TEK peut augmenter considérablement avec l'augmentation de la taille du groupe dans le cas des groupes dynamiques. Cette situation est connue sous le nom du phénomène *1-affecte-n* (*1-affects-n*) qui consiste en l'impact d'un changement d'adhésion sur les autres membres du groupe. Certaines solutions proposent d'organiser le groupe en sous-groupes ou zones avec des TEKs différentes pour chaque zone. Ceci réduit l'impact de la redistribution de nouvelles TEKs (et donc du phénomène *1-affecte-n*) du moment que ça va se faire uniquement dans la zone concernée, mais engendre un nouveau coût de traitement lié aux opérations de déchiffrement/re-chiffrement aux bordures des zones. En effet, à chaque fois qu'un message traverse d'une zone à une autre il doit être déchiffré et re-chiffrer avec la TEK de la zone destination.

Le reste du papier est organisé comme suit: dans la section suivante 2, nous présentons quelques solutions de gestion de clé de groupe existantes. Dans la section 3, nous présentons notre solution de gestion de clé AGKM. Ensuite dans la section 4 nous détaillons la stratégie de redistribution de clé utilisée dans notre solution . La section 5 concernera la génération et vérification des clés des membres. La section 6 est consacrée aux simulations. Nous terminons par une conclusion.

2 Travaux antérieurs

Les protocoles de gestion de clé dans le groupe sont très étudiés ces dernières années. Dans [JA03], [RH03], [SBB+03], [CS05] les auteurs ont des études comparatives sur quelques protocoles et tenté de les classifier selon différents critères: selon la façon de partage de la clé: approches à *clé commune* et approches à *clé par sous-groupe* ou sur la façon de la gestion de clé: approches centralisée, décentralisée ou distribuée.
Parmi les protocoles à clé commune centralisés les plus connus, nous trouvons GKMP (*Group Key Management Protocol*) proposé par Harney et Muckenhirn [HM97a, HM97b] qui utilise l'approche paires point-à-point. Dans GKMP, le serveur de clés génère un paquet de clé de groupe GKP (*Group Key Packet*) qui contient deux clés: une clé de trafic pour le groupe appelée GTEK (*Group Trafic Encryption Key*) et une clé de chiffrement pour le groupe appelée GKEK (*Group Key Encryption Key*). La GTEK est utilisée pour chiffrer le trafic tandis que la GKEK est utilisée pour sécuriser la distribution des nouveaux paquets GKP quand c'est nécessaire. Le serveur de clé partage secrètement avec chaque membre valide une clé secrète appelée KEK (*Key Encryption Key*). Quand un nouveau membre adhère à la session, le serveur de clé génère un nouveau GKP contenant une nouvelle GTEK pour assurer la confidentialité passée. Il envoie ensuite ce GKP, via un canal sécurisé, au nouveau membre chiffré avec la KEK qu'ils partagent secrètement. Il l'envoie aussi aux anciens membres chiffré avec l'ancienne GTEK. Le serveur de clé rafraîchit périodiquement le GKP et utilise la GKEK pour sa redistribution aux membres du groupe. Quand un membre quitte le groupe, le serveur de clés génère une nouveau GKP (GTEK et GKEK) et l'envoie à tous les membres restants chiffré avec la KEK qu'il partage avec chaque membre. Pour assurer la confidentialité future, GKMP envoie des messages de redistribution de clé d'une complexité $O(n)$ pour chaque départ de groupe.
Wong et al. [WGL98, WGL00], Wallner et al. [WHA99] ont proposé simultanément le protocole LKH (*Logical Key Hierarchy*). Dans LKH, le serveur de clé maintient un arbre

dont chaque nœud est une clé. Les nœuds internes représentent les KEKs (*Key Encryption Keys*) et les nœuds feuilles de l'arbre représentent les clés secrètes que partagent chaque membre avec le serveur. Chaque membre détient une copie de sa clé secrète se trouvant dans le nœud feuille et toutes les KEKs se trouvant dans le chemin reliant le nœud feuille vers la racine de l'arbre. Le nœud racine contient la clé de trafic (TEK). Pour un arbre binaire équilibré, chaque membre sauvegarde au maximum $1 + log_2(n)$ clés, où n est le nombre de membres dans le groupe.
Cette *hiérarchie de clés* permet de réduire le nombre de messages de redistribution de clé à l'ordre de $O(\log n)$ tandis que le nombre de message a une complexité de l'ordre de $O(n)$ dans GKMP.

Parmi les approches à TEK par sous-groupe nous trouvons protocole Iolus proposé par Mittra dans [Mit97] qui est une structure qui utilise une organisation hiérarchique en sous-groupes. Chaque sous-groupe est un groupe multicast à part entière avec sa propre adresse multicast et éventuellement son protocole de routage multicast. L'ensemble des sous-groupes forment un groupe multicast virtuel. Chaque sous-groupe est contrôlé par un agent GSA (*Group Security Agent*) qui est responsable de la gestion de clé au sein du sous-groupe. Un contrôleur principal appelé GSC (*Group Security Controller*), qui est la racine de la hiérarchie, contrôle tous les contrôleurs de sous-groupes (GSAs). Chaque sous-groupes utilise sa propre clé de trafic (TEK). Dans Iolus, lorsque un changement d'adhésion a lieu dans un sous-groupe, uniquement ce sous-groupe est impliqué dans le processus de redistribution de clé. De cette manière, Iolus s'étend au groupe de grande taille et réduit le phénomène *1-affecte-n*. Cependant, Iolus possède l'inconvénient d'affecter le chemin de données. En effet, les données devrait être traduites[1] à chaque fois qu'elles passent d'un sous-groupe à un autre. Ceci engendre un coût dû au déchiffrement / re-chiffrement, ce qui n'est pas toléré par les applications qui sont sensibles au délais de transmission telles que les applications temps réel. Dans [CGBB08], les auteurs proposent le protocole ASGK qui est une solution décentralisée où le groupe est organisé en une hiérarchie de zones et chaque zone est administrée par un agent appelé *Area Security Agent (ASA)*. L'idée de ASGK est de construire des clusters de zones pour utiliser la même TEK en comparant. Dans ASGK lorsqu'un cluster est créé, toutes les zones se trouvant en bas de ce cluster appartiendront automatiquement à ce nouveau cluster si elles faisaient partie du même ancien cluster. ASGK est une solution qui ne supporte pas la mobilité. Peu de protocoles de gestion de clé traitement la mobilités des noeuds au sein du groupe [CLW06, RL06]. Dans [PMJ02], Di Pietro et al. ont proposé une amélioration de LKH appelée LKH++ pour les réseaux mobiles sans fil. Dans [KPA03], Kamat et al. proposent une version amélioré de Iolus baptisée M-Iolus. Dans M-Iolus, les sous-groupes sont aussi partitionnés en micro sous-groupes pour réduire encore plus le phénomène *1-affecte-n*. Lorsqu'un membre se déplace d'un sous-groupe à un autre ou d'un micro sous-groupe à un autre micro sous-groupe la clé de trafic n'est pas changé ni dans l'ancien sous-groupe ni dans le nouveau. Pour éviter cela, M-Iolus effectue un échange de messages entre l'ancien et le nouveau GSC concernant le membre. Cette technique à un prix sur la confidentialité passée au sein du groupe. M-Iolus souffre également du plus grand nombre de zones de chiffrement puisque les micro-sous groupe utilisent une clé supplémentaire pour chiffrer le trafic.

[1] déchiffrées puis re-chiffrées

3 Présentation de AGKM

Dans AGKM, le groupe multicast est organisé en une hiérarchie (arborescence) de zones administratives. Une zone peut être un groupe multicast à part entière avec sa propre adresse multicast, un réseau d'entreprise (*corporate network*) ou un point d'accès sans fil. Pour chaque zone il existe un agent contrôleur, de cette zone, qu'on appellera ASA (*Area Security Agent*). La mission d'un ASA est la gestion de clé au sein de la zone dont il est responsable et l'échange de messages bien spécifiés avec les autres ASAs du groupe. Chaque ASA est membre dans deux groupes (zones), la zone dont il assure le contrôle et la zone parent dans la hiérarchie. Un ASA joue le rôle de *proxy* pour sa zone dans la zone parent. Quand un ASA reçoit un message multicast de la zone parent, il le fait suivre vers les membres de sa zone. Initialement, tous les ASAs utilisent la même *TEK*. Donc, lorsqu'un ASA reçoit des messages destiné au groupe de la part de la zone parent, il le fait suivre vers les membres de la zone sous son contrôle sans aucune transformation. Dans ce cas, on dit que l'agent ASA est dans un état *passif*. Dans une telle configuration, un seul changement d'adhésion (de composition) dans une zone va induire à la mise à jour de la TEK pour toutes les zones, puisque elles utilisent toutes une TEK commune. Par conséquent, quand une zone devient très dynamique, on ferait bien d'*isoler* cette zone des autres zones pour utiliser sa propre TEK pour ne pas les déranger avec les messages de redistribution de clé qu'elle cause, et par la suite réduire le phénomène *1-affecte-n*. Lorsqu'une zone est isolée pour utiliser sa propre TEK, son ASA doit traduire (déchiffrer puis re-chiffrer avec sa TEK) les messages reçus avant de les envoyer aux membres de la zone sous son contrôle. Dans cette situation, on dit que l'ASA est *actif*. Dans notre architecture nous proposons que, dans une situation de dynamisme fort d'une zone (dépassant un certains taux noté α fixé par l'utilisateur), que l'agent ASA prenne la décision d'utiliser une TEK indépendante pour sa zone. Ainsi, lorsque cet ASA reçoit les messages multicast de la zone parent (chiffrés avec la TEK de la zone parent), il doit les envoyer vers les membres de sa zone chiffrés avec sa TEK à lui. Pour ce faire, il les déchiffre en utilisant la TEK de la zone parent, et les re-chiffre ensuite avec la TEK de sa zone pour pouvoir les envoyer donc aux membres de sa zone. Dans cette situation, on dit que l'ASA est dans un état *actif*. Le fait qu'un ASA devienne actif engendre un coût de traitement dû aux opérations de déchiffrement/re-chiffrement des messages reçus. Le paramètre α est fixée selon la nature de l'application utilisée durant la session. Par exemple, on peut prévoir une grande valeur de α pour des applications stables telles que les visio-conférences.

Pour pouvoir isoler une zone dynamique a_i des autres zones, on pousse donc son ASA_i à devenir actif. Cela ne suffit pas pour l'isoler du reste du groupe car les zones d'en bas utiliseront la même clé de trafic que celle-ci. Pour faire un isolement total on pousse donc les zones d'en bas (*fils*) à devenir actives. La différence entre les décisions de devenir actif pour ASA_i et les ASAs d'en bas c'est que la première décision était indépendante et les autres étaient subies. Quand un agent ASA_i décide de s'isoler (devenir actif), on dit que la décision est locale et *indépendante*. Quand cet agent ASA_i devient actif il va s'isoler de la zone parent et des zones fils. C'est pour cela que toutes les zones fils dont les ASAs sont passifs basculent vers l'état actif. La figure 1, montre les différents composants de l'architecture AGKM: **A**daptive **G**roup **K**ey Management with **M**obility Support.

L'Algorithme ci-dessous résume le fonctionnement des ASAs à travers le temps.

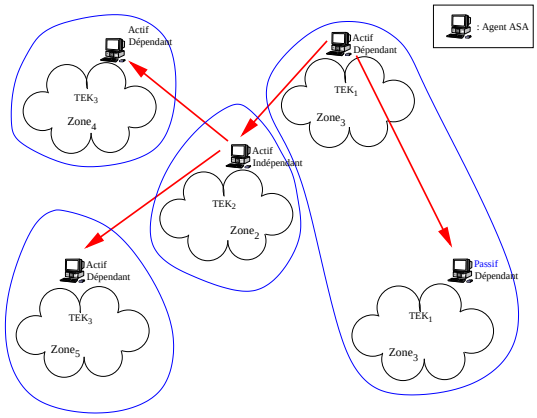

Fig. 1: Architecture AGKM: **A**daptive **G**roup **K**ey Management with **M**obility Support

Entrées: ASA_i, α
Sorties: Mise à jour de l'état de ASA_i
si la fréquence des changements d'adhésion dans la zone a_i est supérieure à α **alors**
ASA_i devient **actif**
ASA_i devient **indépendant** en décision
pour chaque ASA_j tel que ASA_i est père de ASA_j **faire**
si ASA_j est **dépendant** en décision **alors**
ASA_j devient **actif**
finsi
fin pour
sinon
// Dans ce cas la zone n'est pas dynamique
ASA_i devient **passif**
ASA_i devient **dépendant** en décision
pour chaque ASA_j tel que ASA_i est père de ASA_j **faire**
si ASA_j est **dépendant** en décision **alors**
ASA_j devient **passif**
finsi
fin pour
finsi

Il est à noter que la passage du statut *indépendant* vers le statut *dépendant* se fait localement pour chaque agent ASA. Un agent ASA devient *indépendant* uniquement si il décide de devenir *actif* (zone dynamique) et devient *dépendant* uniquement si il décide de devenir *passif* (zone stable). Un agent *indépendant* est un agent qui n'accepte pas que l'agent parent lui modifie son état (*actif* ou *passif*). Un agent *dépendant* est un agent capable de s'adapter aux exigences de l'agent parent. Nous supposons que initialement tous les statuts sont *dépendants*.

Contrairement à ASGK [CGBB08] où un nouveau cluster peut contenir plusieurs zones, dans AGKM un nouveau cluster créé contiendera uniquement la zone concernée. Cette

zone va se séparer de toutes les autres zones (parent et fils). Dans ASGK un nouveau cluster créé veut dire division d'un cluster en deux clusters, alors que dans AGKM on peut obtenir plusieurs clusters (≥ 2) comme résultat de l'opération de création d'un nouveau cluster du moment qu'il pousse aussi les zones fils à s'isoler.

4 Stratégies de mise à jour de clé

Dans le but d'assurer de parfaites confidentialités *passée et future*, à chaque fois qu'un changement d'adhésion a lieu dans une zone, une redistribution de clé aura lieu dans toutes les zones utilisant la même TEK. Rappelons qu'une qu'une zone avec un ASA *passif* utilise la même TEK que la zone parent. Donc, une nouvelle *TEK* doit être générée et redistribuée aux membres de toutes ces zones. Chaque ASA_i partage avec tous les membres de sa zone une clé de chiffrement de clé appelée KEK_i (*Key Encryption Key*). Nous supposons que lorsqu'il y a un changement d'adhésion dans une zone active, la TEK est générée par l'ASA de cette zone et lorsqu'il y a un changement d'adhésion dans une zone passive (dont l'ASA est passif), la TEK est générée par le premier ASA actif rencontré dans le chemin vers la racine. En effet, dans ce dernier cas l'ASA actif utilise la même TEK que la zone affectée. On dit que ces zones forment un *cluster* de zones dont la racine est un ASA actif et toutes les zones internes ont des ASAs passifs.

Nous supposons que les ASAs partagent entre eux une clé symétrique appelée clé de session (SessionKey, *Session Key*) générée par l'ASA auquel la source du trafic est connectée. Donc, donc ce qui suit nous supposons que la clé de session a déjà été générée par l'ASA source lors de l'établissement de la session et distribuée vers les autres ASAs.

Dans ce qui suit nous présentons la stratégie de redistribution de la TEK lors des adhésions (*joins*), départs (*leaves*) et la mobilité des membres dans le groupe.

4.1 Redistribution lors d'une adhésion

Quand un nouveau membre m_l localisé dans la zone a_i adhère à la session sécurisée, ASA_i responsable de la zone a_i établit une clé secrète MEK_l (*Member Encryption Key*) pour ce membre m_l (voir section 5). Pour garantir la *confidentialité passée*, tous les ASAs, appartenant au même *cluster* que celui de la zone où le membre a rejoint la session, doivent redistribuer la nouvelle TEK aux membres de leurs zones respectives. Pour distribuer la nouvelle TEK aux membres de la zone a_i où l'adhésion a eu lieu, ASA_i envoie un message contenant la nouvelle TEK et la KEK_i courante chiffrées avec MEK_l ($\{\text{new TEK}, KEK_i\}_{MEK_l}$) au nouveau membre m_l et il envoie cette nouvelle TEK par multicast aux autres membres dans sa zone, chiffrée avec sa KEK_i courante ($\{\text{new TEK}\}_{KEK_i}$). Pour distribuer la nouvelle TEK dans les autres zones a_t du même cluster, chaque ASA_t envoie par multicast à sa zone a_t la nouvelle TEK chiffrée avec sa KEK_t ($\{\text{new TEK}\}_{KEK_t}$). Donc, chaque adhésion à la session nécessite un message multicast dans chaque zone appartenant au cluster où l'adhésion a eu lieu, en plus d'un message unicast envoyé au nouveau membre.
Un intrus ne peut pas avoir accès à la nouvelle TEK, et donc ne peut pas connaître le contenu de la session. Ceci est assuré parce qu'il ne peut pas connaître ni l'ancienne TEK (qui n'est pas compromise dans le cas d'un *join*), ni la clé secrète MEK_l du nouveau membre et ni les KEKs des autres zones. Notons que la nouvelle TEK peut être aussi envoyée par multicast dans chaque zone en la chiffrant avec l'ancienne TEK. La confidentialité passée

est assurée du moment que l'intrus ne connaît pas l'ancienne TEK.

4.2 Redistribution lors de la mobilité d'un membre

Un membre mobile peut se déplacer d'une zone a_i vers une autre zone a_j.

Dans notre approche, nous utilisons un mécanisme de vérification du membre mobile dans la nouvelle zone afin d'éviter de renouveler la TEK dans les deux zones (l'ancienne et la nouvelle). Pour ce faire, les opérations suivantes sont réalisées lorsque le membre m_l se déplace de la zone a_i vers la zone a_j: m_l envoie à ASA_j une requête d'adhésion à a_j en spécifiant qu'il vient de a_i. Cette requête est chiffrée avec la clé MEK_l. A partir des informations concernant le membre m_l, ASA_j recalcule MEK_l déchiffre ensuite la requête de m_l. Si il y a succès (la MEK présentée est valide), il vérifie si les membre est autorisé. Si le membre est autorisé, ASA_j envoie à m_l la clé KEK_j et éventuellement TEK_j (si $TEK_i \neq TEK_j$) chiffrées avec MEK_l. Rappelons que cette dernière clé est personnelle au membre m_l. Cependant, n'importe quel ASA est capable de la recalculer(voir section 5). L'ancien ASA (ASA_i) garde une trace que m_l a été membre dans sa zone a_i. Nous concluons que lors de la mobilité d'un membre, la TEK n'a pas été renouvelée.

4.3 Redistribution lors d'un départ

Quand un membre m_l est exclu d'une zone a_i, tous les ASAs appartenant au même cluster doivent envoyer la nouvelle TEK aux membres de leurs zones respectives, pour garantir *la confidentialité future*. L'agent ASA_i génère une nouvelle KEK_i. Pour distribuer les nouvelles TEK et KEK_i dans la zone a_i, ASA_i envoie par multicast aux membres de sa zone un paquet contenant les nouvelles KEK_i et TEK, chiffrées avec les clés secrètes MEK_k qu'il partage avec chaque membre de sa zone a_i, excepté la clé MEK_l associé au membre expulsé m_l ({ {new TEK, new $KEK_i\}_{MEK_1}$, {new TEK, new $KEK_i\}_{MEK_2}$, ...{new TEK, new $KEK_i\}_{MEK_k}$, ...}, avec $k \neq l$). Par conséquent, chaque membre valide restant sera capable de déchiffrer et récupérer les nouvelles TEK et KEK_i en utilisant sa clé secrète MEK_k. Le membre expulsé ne pourra pas connaître ces deux clés (TEK et KEK_i) par ce que sa clé MEK_l n'a pas été utilisée pour chiffrer les nouvelles clés.

Pour distribuer la nouvelle TEK dans les autres zones a_t du même cluster, chaque ASA_t envoie par multicast à sa zone un message contenant la nouvelle TEK chiffrée avec la KEK_t qu'il partage secrètement avec les membres de sa zone a_t ({new TEK}$_{KEK_t}$). On déduit que chaque expulsion de la session induit seulement un seul message multicast dans les autres zones du même cluster où le départ a eu lieu.

Avec le même principe expliqué ci-dessus, toutes les TEKs de toutes les zones où m_l était membre (les zones déjà visitées) seront changées s'ils n'ont pas été changées entre temps par de départs. Un départ d'une zone induit toujours aux changement de sa KEK. Un changement de TEK ne protège pas la zone du moment que la KEK est restée la même.

Un intrus ne peut pas connaître la nouvelle TEK, donc il n'aura pas accès au contenu de la session. Ceci est rendu possible du fait qu'il ne connaît pas les autres clés secrètes MEK_k des autres membres restants de sa zone ni des zones visités, et aussi il ne connaît aucune clé de zone KEK_j des autres zones puisque'il ne les a jamais visité. Rappelons que dans le cas d'une expulsion, nous ne pouvons pas utiliser l'ancienne TEK pour chiffrer la nouvelle TEK parce qu'elle est compromise du moment qu'elle est connue par le membre qui a quitté la session.

5 Génération et Vérification de la MEK d'un membre

Rappelons que tous les ASAs partagent secrètement une clé symétrique $SessionKey$. Cette clé est générée au début de la session par l'ASA auquel la source du trafic est connectée.

Lorsque un nouveau membre m_l localisé dans la zone a_i adhère au groupe il envoie un message *join* signé avec sa clé privée à ASA_i. Ensuite, ASA_i génère la clé MEK_l en utilisant la formule 1 ci-dessous et l'envoie au membre m_l signée avec la clé publique de ce dernier. Pour générer la MEK du membre m_l, ASA_i utilise une fonction de dérivation de clé KDF (*Key Derivation Function*) comme suit:

$$MEK_l = KDF(SessionKey, data_l) \tag{1}$$

où,

- $SessionKey$ = La clé partagée par les ASAs.
- $data_l$ = Contient l'identité du membre m_l et des informations propres à lui telle que sa clé publique.

Comme dans [KF07], la KDF utilisée est une fonction pseudo aléatoire (*Pseudo Random Function*) qui est une combinaison de fonctions de hachage à sens unique (MD5, HMAC, SHA1, SHA256,..). Plusieurs fonctions pseudo aléatoires peuvent être envisagées dans la KDF. Cependant, pour avoir une fonction de dérivation de clé coordonnée, la même fonction pseudo aléatoire devrait être utilisée par tous les ASAs. Quelques fonctions pseudo aléatoires basées sur HMAC-SHA1, HMAC-MD5 , HMAC-SHA256 [KF07] peuvent être envisagées.

La clé $SessionKey$ joue un grand rôle dans la génération des MEKs, car c'est à partir de celle-ci que les ASAs peuvent re-générer les MEKs de tous les membres. C'est pour ce la qu'ils sont les seuls capables de recalculer les MEKs.

Notons que la durée de vie d'une MEK est liée à la durée de vie de la session. Ceci dit, un membre peut avoir plusieurs MEKs lui permettant de participer à plusieurs sessions au même temps.

6 Simulation et résultats

Dans nos simulations, nous utilisons la hiérarchie AGKM représentée dans la figure 1, qui a été utilisée pour illustrer l'architecture AGKM.

Pour la simulation, nous avons considéré une session multicast de trois heures. Les membres arrivent au groupe selon une loi de Poisson avec des inter-arrivées moyennes de 20 secondes, des inter-déplacement (*inter-moves*) moyens de 15 secondes et restent dans la session 30 secondes en moyenne. Après chaque période $\theta = 15min$, les ASAs compare leur fréquence de dynamisme par rapport à α et décident de devenir *actif* ou *passif* selon l'algorithme présenté en section 3. La fréquence de dynamisme ou de changement d'adhésion pour un agent est donnée par le nombre d'adhésion et départ sur une période de temps. De notre cas, elle est calculée chaque période $\theta = 15minutes = 15 * 60secondes$. Pour des raisons de simplicité et de précision cette fréquence est le rapport du nombre de changement d'adhésion durant θ unité de temps sur θ en secondes.

L'affectation des membres arrivés aux différentes zones se fait aléatoirement avec un pourcentage des arrivées pour chaque zone qui varient dans le temps. Dans la table 1, nous montrons la distribution des pourcentages des arrivées pour chaque zone a_i durant toute la session. Pour chaque heure de la session, nous spécifions le pourcentage des arrivées pour chaque zone pour mettre en évidence l'aspect dynamique de AGKM. Pour la mobilité des membres nous choisissons aléatoirement un membre et nous le faisons déplacer vers une autre zone différente de la zone courante choisie aléatoirement. Nous présentons deux variantes de AGKM dans les simulations: *AGKM with mobility* qui est la solution telle qu'elle a été présentée dans ce papier et *AGKM without mobility* où on considère un déplacement comme un départ de la première zone et une adhésion dans la nouvelle.

Période	a_1	a_2	a_3	a_4	a_5
0h-1h	20%	20%	40%	15%	5%
1h-2h	40%	20%	20%	15%	5%
2h-3h	15%	20%	5%	40%	20%

Tab. 1: Distribution des pourcentages d'arrivées aux différentes zones

Sans perte de généralité, la technique de traduction de clé que nous utilisons pour illustrer notre approche est le déchiffrement/re-chiffrement, pour pouvoir comparer notre protocole avec d'autres de la littérature (GKMP, Iolus, M-Iolus) sur les mêmes bases. Autrement dit, quand un agent ASA devient actif, il déchiffre les messages reçus en utilisant la TEK de la zone parent, qui est différente. Ensuite, il re-chiffre les messages en utilisant la TEK de sa zone. Après ceci, il envoie par multicast le message re-chiffré aux membres de sa zone. Pour des raisons de simplicité, nous supposons que tous les ASAs ont les mêmes caractéristiques matérielles et mêmes puissances de traitement. Ils utilisent un crypto-système symétrique tel que DES [Fed93], 3DES [Ame98] ou AES [Fed01]. Ceci nous permettra de les traiter de la même manière.

6.1 Impact du seuil de dynamisme α

Dans un premier temps, nous nous sommes intéressés à l'étude de l'impact du taux de dynamisme α dans les performances de notre protocole. Si le dynamisme d'une zone dépasse ce taux, la zone concernée est isolée pour utiliser sa propre TEK. Nous avons voulu trouver comment il permet d'adapter le comportement de notre protocole pour répondre aux besoins de l'application utilisée en terme de synchronisation et de tolérance aux délais.

Pour cela, nous avons fait varier la valeur α et nous avons exécuté d'intensives simulations pour chaque valeur de α. La figure 2 montre l'impact de α sur le *nombre de membres affectés par les changements d'adhésion*: *1-affecte-n*. Nous remarquons bien que lorsque la valeur α augmente, le phénomène *1-affecte-n* augmente aussi. Mais il augmente plus rapidement pour *AGKM without mobility* puisque'il est affecté par la mobilité des membres.

Comme nous pouvons voir dans la figure 3, il est évident que le nombre des opérations de déchiffrement/re-chiffrement (qui est égal au nombre d'ASAs actifs) diminue parce que les ASAs on tendance à devenir passif quand α augmente.

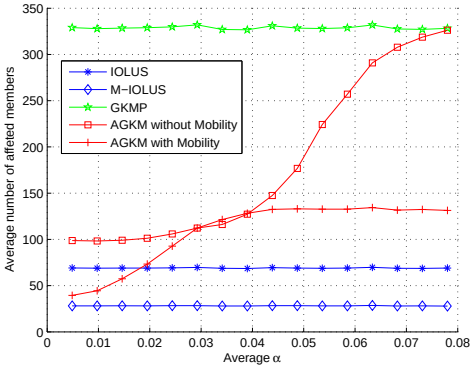

Fig. 2: Impact de la variation de α sur le phénomène *1-affecte-n*

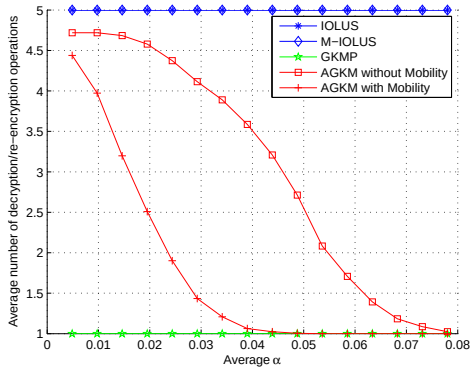

Fig. 3: Impact de la variation de α sur le déchiffrement/re-chiffrement

Nous concluons que le paramètre α donne un large spectre de flexibilité pour notre approche afin qu'elle s'adapte au besoin de l'application utilisée. Les petites valeurs de α pousse la solution à devenir une approche à clé par sous-groupe et les grandes valeurs la pousse à être une solution à clé commune. Faire varier α permet à MASGK de combiner les deux approches.

Pour la suite des simulations nous utilisons une petite valeur de $\alpha = 10^{-3}$ pour s'intéresser plus au groupes dynamiques.. Dans les prochaines simulations, nous ferons varier la moyenne des inter-arrivées, la moyenne de la durée de séjour et la moyenne des inter-déplacements et montrer l'impact de chaque paramètre sur les performances de notre protocole.

6.2 Impact de la taille du groupe

Dans un deuxième temps, nous nous sommes intéressés à la mise à l'échelle (*scalabilité*) de la taille du groupe dans AGKM par rapport aux autres approches Iolus, M-Iolus et GKMP. Nous avons tester les performances de ces protocoles par rapport aux phénomène *1-affecte-n* et à l'influence des opérations de déchiffrement/re-chiffrement.

Deux paramètres de notre modèle de simulation peuvent contrôler *la taille du groupe*: la *moyenne des inter-arrivées* des membres vers la session, et la *duré moyenne de séjour* des membres dans la session.

6.2.1 Impact des inter-arrivées

Dans un premier lieu, nous nous sommes intéressés à étudier l'impact de la moyenne des inter-arrivées des membres dans le groupe. Dans les figures 4 et 5 nous avons fait varier la moyenne des inter-arrivées de 1 seconde vers 65 secondes et nous avons mesuré les coûts dûs au phénomène *1-affecte-n* et au déchiffrement / re-chiffrement, respectivement.

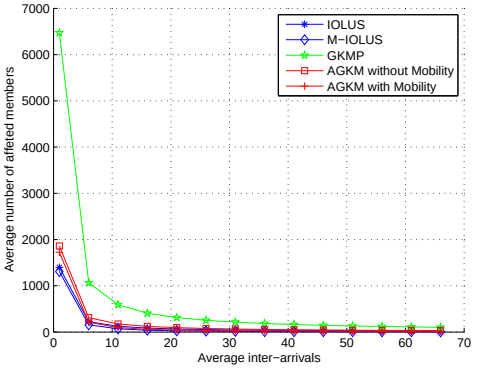

Fig. 4: Impact da la variation des inter-arrivées sur le coût *1-affecte-n*

Les plus petites valeurs des inter-arrivées correspondent aux tailles les plus larges du groupe. Avec GKMP, tous les membres sont affectés, et donc, le protocole souffre du phénomène *1-affecte-n*. Typiquement, nous constatons dans la figure 4 que pour l'inter-arrivées dans l'intervalle $[1s : 5s]$, le nombre de membres affectés pour GKMP est très grand. Pour le même intervalle des valeurs des inter-arrivées, Iolus et AGKM réduisent le nombre de membres affectés au minimum. Donc, notre approche et les deux variantes de Iolus s'étendent bien aux larges groupes. De plus, nous constatons dans la figure 5 que lorsque l'inter-arrivée augmente, la taille du groupe diminue, mais Iolus et M-Iolus possèdent toujours le même nombre des opérations de déchiffrement/re-chiffrement (5 opérations car 5 sous-groupes). Par contre, AGKM réduit cet *overhead* (charge) et maintient un *overhead* réduit du phénomène *1-affecte-n*. *AGKM without mobility* maintient un nombre d'opérations de déchiffrement semblable à celui de Iolus puisqu'il est est très affecté par la mobilité des membres.

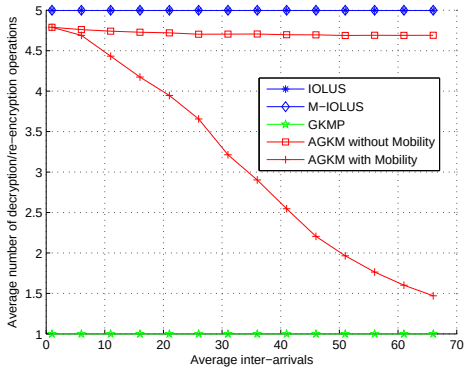

Fig. 5: Impact da la variation des inter-arrivées sur le coût déchiffrement/re-chiffrement

6.2.2 Impact de la durée de séjour

Dans un second lieu, nous nous sommes intéressé à étudier l'impact de la moyenne des durées de séjour des membres dans le groupe.

Nous pouvons constater le même phénomène en faisant varier la moyenne de la durée de séjour des membres dans la session. Dans la figure 6 nous remarquons que AGKM, Iolus et M-Iolus sont scalables aux groupes larges.

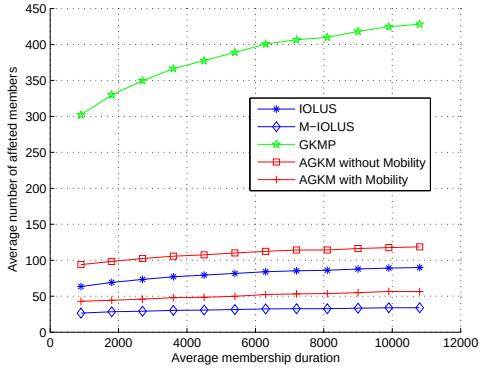

Fig. 6: Impact de la variation de la durée de séjour des membres sur le coût *1-affecte-n*

Dans la figure 7, nous remarquons également que AGKM (*with mobility*) possède l'avantage, par rapport à M-Iolus, de réduire l'*overhead* de déchiffrement/re-chiffrement sans augmenter de façon dramatique l'*overhead* du phénomène *1-affecte-n*. AGKM sans mobilité montre des même résultats en le comparant avec Iolus.

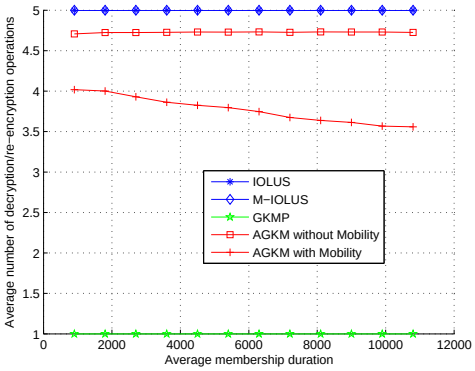

Fig. 7: Impact de la variation de la durée de séjour des membres sur le coût déchiffrement/re-chiffrement

6.3 Impact de la mobilité des membres

Finalement, nous nous sommes intéressés à étudier l'impact de la mobilité des membres dans le groupe. Dans les figures 8 et 9 nous avons fait varier la moyenne des inter-déplacements de 1 seconde vers 50 secondes et nous avons mesuré les coûts dûs au phénomène *1-affecte-n* et au déchiffrement / re-chiffrement, respectivement.

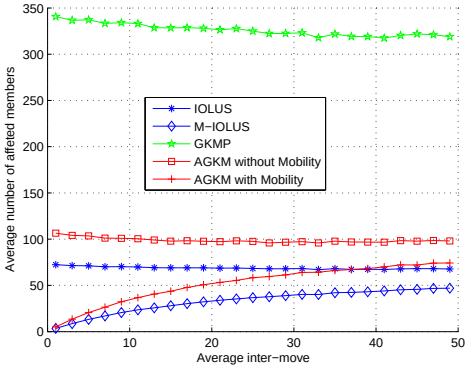

Fig. 8: Impact da la variation des inter-déplacements sur le phénomène *1-affecte-n*

Dans le figure 8 on remarque bien que pour de très petite valeur des inter-déplacements [1s:10s] (groupe très dynamique) AGKMP réduit considérablement le nombre de membre affectés. Ce nombre augmente en augmentant la valeur des inter-déplacements. Pour des valeurs très grandes des inter-déplacements l'impact de la mobilité est négligeable sur les performances des protocoles simulés. Ceci, dit en augmentant l'inter-déplacement les deux variantes de AGKM réduisent le nombre des opérations de déchiffrement/re-chiffrement

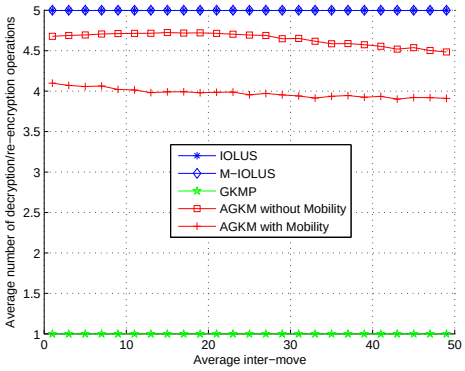

Fig. 9: Impact da la variation des inter-déplacements sur le coût déchiffrement/re-chiffrement

dans la figure 9 inférieurs à ceux de Iolus et M-Iolus.

7 Conclusion

Dans ce papier, nous avons proposé une approche adaptative pour la gestion de clé de groupe dans les environnements mobiles. Le but de notre architecture est d'isoler les zones caractérisées par un fort dynamisme, c'est-à-dire, les zones où les membres arrivent et quitte le group avec une grande fréquence. Le fait d'isoler les zones dynamique réduit considérablement le phénomènes *1-affecte-n* au sein du groupe. En effet, le phénomène *1-affecte-n* va se limiter à la zone concernée. Les zones stables de faible dynamisme utilisent des TEKs communes puisqu'il n y a pas de phénomène *1-affecte-n*. Le support de mobilité utilisé dans notre solution permet de vérifier les membres sans avoir à mettre à jour la clé de trafic ni dans l'ancienne zone ni dans la nouvelle.

A travers les simulations, nous avons montré que notre solution produit de meilleures performances par rapport à d'autres solutions de la littérature. Notre solution convient parfaitement aux environnements de forte mobilité où le membres peuvent se déplacer au sein du réseau et changer de point d'attachement.

References

[Ame98] American National Standards Institute. *Triple Data Encryption Algorithm Modes of Operation*, 1998. ANSI X9.52-1998.

[CGBB08] 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.

[CLW06] 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.

[CS05] Y. Challal and H. Seba. Group Key Management Protocols: A Novel Taxonomy. *Enformatika, International Journal of Information technology*, 2(1), 2005.

[Dee88] S. E. Deering. Multicast Routing in Internetworks and Extended LANs. *ACM SIGCOMM*, August 1988.

[Fed93] Federal Information Processing Standards Publication. *Data Encryption Standard (DES)*, December 1993. FIPS PUB 46.

[Fed01] Federal Information Processing Standards Publication. *Advanced Encryption Standard (AES)*, November 2001. FIPS PUB 197.

[HM97a] H. Harney and C. Muckenhirn. RFC2093: Group Key Management Protocol (GKMP) Architecture. *IETF RFC 2093*, July 1997.

[HM97b] H. Harney and C. Muckenhirn. RFC2094: Group Key Management Protocol (GKMP) Specification. *IETF RFC 2094*, July 1997.

[JA03] P. Judge and M. Ammar. Security Issues and Solutions in Multicast Content Distribution: A Survey. *IEEE Network*, 17(1):30–36, January/February 2003.

[KF07] S. Kelly and S. Frankel. RFC4868: Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec . *IETF RFC 4868*, May 2007.

[KPA03] 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.

[Mit97] 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.

[PMJ02] 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.

[RH03] S. Rafaeli and D. Hutchison. A Survey of Key Management for Secure Group Communication. *ACM Computing Surveys*, 35(3):309–329, September 2003.

[RKL+04] 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.

[RL06] 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.

[SBB+03] H. Seba, A. Bouabdallah, N. Badache, H. Bettahar, and D. Tandjaoui. Key Management and Multicast Security: a Survey. *Annales des Telecommunications (Annals of telecommunications)*, 58(7-8):1090–1129, 2003.

[WGL98] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *ACM SIGCOMM*, pages 68–79, 1998.

[WGL00] 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.

[WHA99] D. Wallner, E. Harder, and R. Agee. RFC2627: Key Management for Multicast : Issues and Architecture. *IETF RFC 2627*, June 1999.