Sécurité des communications de groupe dans les réseaux ad hoc

Mohamed Salah Bouassida

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**
UFR STMIA

Sécurité des communications de groupe dans les réseaux ad hoc

THÈSE

présentée et soutenue publiquement le 05 Décembre 2006

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Mohamed Salah BOUASSIDA

Composition du jury

Président :	René SCHOTT	Professeur, UHP Nancy
Rapporteurs :	Abdelmadjid BOUABDALLAH	Professeur, UTC Compiègne
	Maryline MAKNAVICIUS-LAURENT	Professeur, INT Evry
Examinateurs :	Isabelle CHRISMENT	Professeur, ESIAL Nancy
	Olivier FESTOR	Directeur de recherche INRIA Lorraine
	Pierre de SAQUI-SANNES	Enseignant Chercheur, HDR, ENSICA Toulouse

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

Remerciements

Je tiens à adresser mes plus chaleureux remerciements à Isabelle Chrisment et lui exprimer toute ma reconnaissance pour son encadrement, ses conseils, son soutien constant ainsi que pour sa confiance, sa générosité et sa patience. J'ai eu l'honneur et le plaisir de travailler sous sa direction pendant mon projet de fin d'études, mon stage de DEA et ma thèse et j'espère pouvoir continuer à travailler avec elle dans le futur.

Je remercie tout autant Olivier Festor, mon directeur de thèse, pour l'ensemble des conseils qu'il a su me donner au quotidien. Je tiens à lui exprimer toute ma gratitude pour ses remarques pertinentes et ses contributions considérables tout au long de mes trois années de thèse.

Je remercie vivement tous les membres de notre équipe de recherche MADYNES, pour l'ambiance chaleureuse et la bonne humeur qu'ils ont pu toujours maintenir et qui ont été un facteur important du bon déroulement de mon travail. Je pense tout particulièrement à Josiane Reffort, notre secrétaire d'équipe, à qui je tiens à témoigner ma reconnaissance et ma sympathie pour sa gentillese, son sourire et sa patience.

Mes plus sincères remerciements vont également à Maryline Maknavicius-Laurent et Abdelmadjid Bouabdallah qui ont accepté d'être les rapporteurs de cette thèse et de participer au jury. Ils ont contribué par leurs nombreuses remarques et suggestions à améliorer la qualité de ce mémoire et je leur en suis très reconnaissant. Je remercie profondément Pierre de Saqui-Sannes et René Shott qui m'ont fait l'honneur de participer au jury de ma soutenance.

Que tous trouvent ici l'expression de mes sentiments les plus respectueux.

À mes très chers parents, nulle dédicace n'est susceptible de vous exprimer ma profonde affection, mon immense gratitude pour tous les sacrifices que vous avez consacrés pour moi

À ma petite soeur Marwa que j'aime plus que tout au monde

À la femme de ma vie Manel

À tous ceux qui m'aiment

Table des matières

Table des figures

Introduction

Introduction générale

1 Cadre scientifique

Ce manuscrit présente la synthèse de trois années de recherche effectuée dans le cadre de ma thèse, au sein de l'équipe de recherche MADYNES[1] au LORIA[2] à Nancy. Le LORIA est une unité mixte de recherche (UMR 7503), commune à plusieurs établissements qui sont le CNRS[3], l'INPL[4], l'INRIA[5], l'Université Henry Poincaré[6] et l'Université Nancy 2[7]. Les missions du LORIA concernent la recherche fondamentale et appliquée au niveau international dans le domaine des Sciences et Technologies de l'Information et de la Communication (STIC), la formation par la recherche en partenariat avec les universités lorraines et le transfert technologique par le biais de partenariats industriels et par l'aide à la création d'entreprises.

L'équipe MADYNES, dirigée par Olivier Festor, est organisée autour de deux axes de recherche : la gestion des réseaux et services et la sécurité des réseaux. Dans ce cadre, plusieurs thématiques particulières font l'objet d'études et de propositions, notamment, la sécurité du plan de gestion, la performance des infrastructures de supervision, la supervision des réseaux pair à pair et enfin la sécurité des communications de groupe dans les réseaux ad hoc. Ce dernier thème constitue le sujet de thèse sur lequel j'ai travaillé sous la direction de Isabelle Chrisment et Olivier Festor. Je présente dans ce manuscrit les travaux effectués ainsi que les résultats obtenus.

2 Contexte et problématique

L'évolution importante d'Internet au cours de la dernière décennie a suscité le développement des technologies vers des réseaux haut-débit et des puissances de calcul de plus en plus rapides et efficaces. Cette croissance a favorisé la conception de nouvelles applications telles que l'audio et la vidéo conférence, la télévision à la demande, les jeux de groupes interactifs, le télé-enseignement, ...Pour ces applications coopératives impliquant un groupe d'utilisateurs, les communications multipoint (*multicast*) sont non seulement appropriées mais indispensables pour un meilleur acheminement des données à de multiples destinataires, pouvant varier dans le temps et dans l'espace. En effet, les transmissions IP multipoint [Dee91] apportent une économie en termes de bande passante et de ressources des routeurs, ainsi qu'une optimisation

[1] Management of Dynamic Networks and Services - http ://madynes.loria.fr
[2] Laboratoire Lorrain de Recherche en Informatique et Automatique - http ://www.loria.fr
[3] Centre National de Recherche Scientifique - http ://www.cnrs.fr
[4] Institut National Polytechnique de Lorraine - http ://www.inpl-nancy.fr
[5] Institut National de Recherche en Informatique et en Automatique - http ://www.inria.fr
[6] http ://www.uhp-nancy.fr
[7] http ://www.univ-nancy2.fr

en temps d'acheminement des flux de données.
Cependant, l'absence de sécurité dans le modèle des communications multicast est l'un des facteurs qui a freiné leur déploiement dans des réseaux à grande échelle, plus particulièrement dans le cadre des applications de diffusion de données à but commercial. Cette limitation a été une motivation majeure pour plusieurs travaux de recherche dont le but principal est d'établir une architecture sécurisée de communications de groupe, évitant toute attaque malveillante. Les services de sécurité offerts par une telle architecture peuvent varier selon la politique de sécurité adoptée dans le cadre de l'application concernée. Les principaux services sont : la confidentialité et l'intégrité des données, l'authentification des données, l'authentification et le contrôle d'accès des membres, la disponibilité de la source, ...

Parallèlement au développement des communications de groupe dans l'Internet, on assiste depuis quelques années à un déploiement exponentiel des réseaux spontanés grâce à l'émergence de nouvelles technologies sans fil et des standards associés (e.g. les réseaux 802.11[8], Wimax[9], ...) et aussi grâce à la disponibilité croissante de terminaux évolués et autonomes (téléphones, PDAs, ...). Les réseaux spontanés permettent à un ensemble de machines hôtes d'être connectées facilement et rapidement entre elles, sans aucune infrastructure préalable. Les réseaux ad hoc sont une illustration de ce concept de spontanéité où chaque nœud contribue activement à la vie du réseau soit en collaborant pour acheminer les données à destination, soit en acceptant d'être à la fois client et fournisseur de contenu. Les réseaux ad hoc sont dynamiques dans l'espace et dans le temps et offrent une grande flexibilité. Cependant, cette flexibilité associée à la vulnérabilité des connexions sans fil nécessite davantage de sécurité pour les données et les utilisateurs.

Plusieurs applications telles que les communications de groupes militaires ou la coordination de forces civiles se déploient aujourd'hui dans des réseaux ad hoc et requièrent des transmissions de données en multicast. Cette nouvelle intégration engendre de nouveaux challenges à relever pour assurer des communications de groupe sécurisées dans un réseau ad hoc, offrant les services de sécurité requis pour de telles communications, tout en s'adaptant à la nature des réseaux ad hoc et à leurs caractéristiques (lien sans fil, mobilité et dynamicité des nœuds,...).

Nous présentons dans un premier temps le modèle IP multicast et les besoins de sécurité à instaurer pour ce type de communications. Ensuite nous présentons les réseaux ad hoc, ainsi que les challenges de sécurité qu'ils impliquent. Et finalement, nous identifions notre problématique : sécuriser des communications de groupe dans les réseaux ad hoc.

2.1 Le modèle IP multicast

La transmission en multicast est le mécanisme le plus efficace et approprié pour les applications orientées vers un groupe d'utilisateurs, telles que l'audio et la vidéo conférence, le télé-enseignement, ... Le modèle IP multicast défini par DEERING [Dee91] est une extension du modèle IP. Il définit la notion de groupe, les types d'adressage et le protocole d'adhésion au groupe. Un *groupe multicast* est un ensemble de stations dont le nombre varie de zéro à l'infini. La composition du groupe est dynamique ; une station peut joindre (Join) ou quitter (Leave) le groupe à tout moment (cf. Figure 1). Un groupe est ouvert ; une station peut émettre un paquet à un groupe sans en faire partie.

[8] http ://grouper.ieee.org/groups/802/11
[9] http ://www.wimaxxed.com/

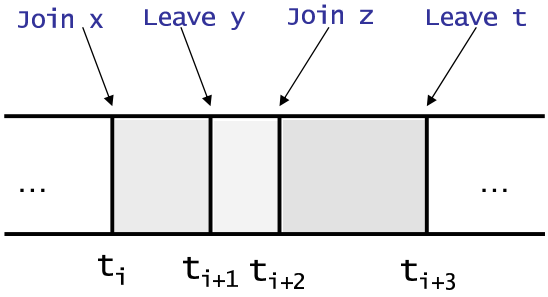


FIG. 1 – Évolution de la vie d'un groupe multicast

Les adresses de groupe forment un sous ensemble des adresses IP, de classe D dans IPv4 et de préfixe FF00 : :/8 dans IPv6. Il existe des groupes permanents d'adresses bien connues et le nombre de membres peut être occasionnellement nul. D'autres groupes sont transitoires et possèdent des adresses allouées dynamiquement.

Les protocoles d'adhésion au groupe IGMP (*Internet Group Management Protocol*) dans IPv4 et MLD (*Multicast Listener Discovery*) dans IPv6 [Dee91] opèrent entre les machines et leur routeur multicast, auquel elles sont reliées directement. Ils permettent à une machine d'informer son routeur qu'elle désire recevoir les transmissions destinées à un groupe multicast spécifique. Ainsi, le routeur interroge périodiquement son réseau local, afin de déterminer s'il y a encore des membres appartenant à des groupes multicast. En se basant sur les informations acquises par le protocole d'adhésion (IGMP ou MLD), un routeur est capable de déterminer quel trafic multicast nécessite d'être diffusé dans son réseau. Ces routeurs multicast utilisent ces informations, conjointement avec les protocoles de routage multicast (e.g. MOSPF [Moy94], PIM [DEF94] dans les réseaux filaires et MOLSR [LJM+03], MAODV [RP00] dans les réseaux ad hoc), afin de supporter le multicast sur Internet. La figure 2 présente les composants de base du modèle IP multicast.

2.2 La sécurité dans les communications de groupe

Les services de sécurité que doit fournir une architecture de sécurisation de communications de groupe sont liés aux données multicast émises par la source, mais aussi aux identités des participants au groupe multicast. Nous distinguons cinq principaux services :

- **Confidentialité des données.** Cette propriété signifie que seuls les membres adhérents au groupe peuvent accéder aux données émises par la source, même si ces données sont diffusées dans tout le réseau. Pour assurer cette confidentialité, une clé symétrique est utilisée par la source pour chiffrer les données et par les membres pour les déchiffrer. Cette clé est appelée "clé de chiffrement de données" (TEK : *Traffic Encryption Key*).
- **Secrets futur et passé.** Un membre ayant quitté le groupe multicast, ne doit plus être capable de déchiffrer le flux multicast après son départ (*Forward Secrecy*). De même, une entité qui adhère au groupe ne doit pas être capable d'accéder au flux de données émis avant son arrivée (*Backward Secrecy*). Il est donc nécessaire de déclencher un processus de renouvellement de la clé de chiffrement de données (TEK) après chaque événement d'ajout ou de retrait d'entités dans le groupe. La nouvelle TEK est ainsi renouvelée et

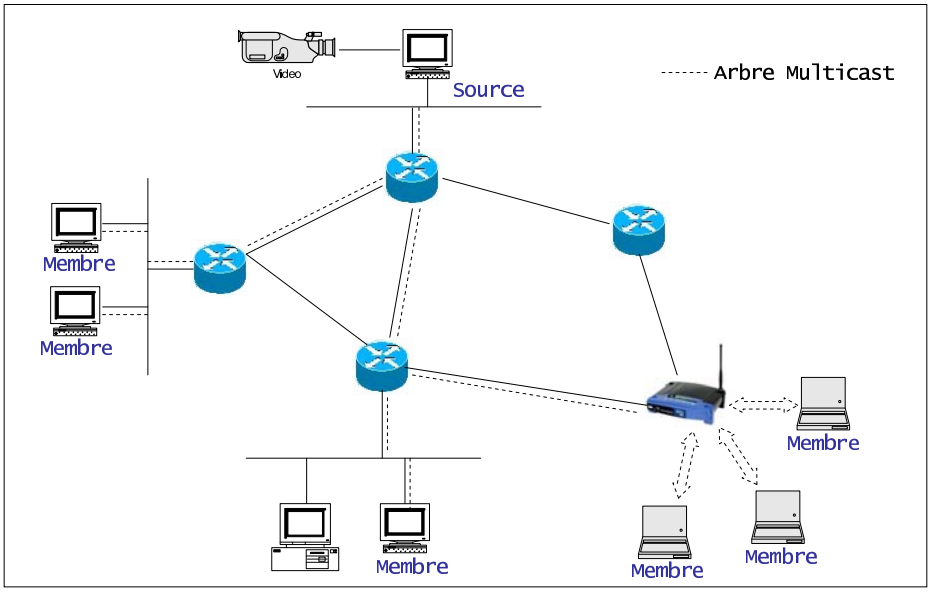


FIG. 2 – Le modèle IP multicast

distribuée à tous les membres du groupe y compris le nouveau en cas d'arrivée et excepté le membre quittant en cas de départ. L'envoi de la clé TEK doit être sécurisé grâce à des clés de chiffrement de clés, appelées KEKs (*Key Encryption Keys*).
Il est à noter que les services de secrets futur et passé sont réalisés selon la politique de sécurité adoptée par l'application en question. En effet, c'est la source du groupe qui est responsable de déclencher un processus de renouvellement des clés de son groupe, dépendant du niveau de sécurité souhaité et de la sensibilité des données transmises.
Le renouvellement de la clé du groupe doit faire face aux phénomènes "*1 affecte n*" et "*1 n'est pas égal à n*". Le phénomène "*1 affecte n*" consiste à affecter tous les membres du groupe par le renouvellement de la clé, déclenché suite à un événement d'ajout ou de retrait d'un seul membre du groupe. Le phénomène "*1 n'est pas égal à n*" consiste à traiter les membres du groupe séparément en leur envoyant des messages en unicast lors d'un renouvellement de la clé du groupe.

- **Contrôle d'accès des membres du groupe**. Ce service de sécurité garantit que l'adhésion au groupe est assurée via une liste de contrôle d'accès ACL (*Access Control List*), contenant toutes les entités autorisées à joindre le groupe.
- **Authentification de la source.** Cette propriété de sécurité exige que les membres d'un groupe multicast s'assurent de l'identité de la source à chaque fois qu'ils reçoivent le flux émis en multicast. Ce service représente la clé de voûte pour la confidentialité et l'intégrité des données.
- **Authentification du groupe.** Cette propriété de sécurité requiert que les membres d'un groupe multicast s'assurent que la source du flux multicast est bien adhérente au groupe.

Le modèle IP multicast est attractif, efficace et adapté au passage à l'échelle. Cependant, ses caractéristiques présentent quelques vulnérabilités auxquelles doivent faire face les services de sécurité à instaurer pour assurer des communications de groupe sécurisées. En effet, dans le modèle IP multicast, aucune identification des hôtes participants au groupe n'est effectuée. Les adresses des groupes multicast sont publiquement connues ; toute entité du réseau peut ainsi joindre le groupe ou le quitter sans y avoir été invitée et accéder au flux de données multicast envoyé par la source. Une entité malveillante peut également envoyer des données multicast aux membres du groupe, sans en être adhérente et sans aucune autorisation ou contrôle d'accès. Ces actions peuvent engendrer des attaques de dénis de services (DoS) et d'atteinte à la confidentialité et à la disponibilité des données. De plus, les données multicast s'acheminent dans le réseau via plusieurs chemins, construisant l'arbre multicast du groupe. Ceci augmente les opportunités d'écoutes clandestines et d'attaques malicieuses.

2.3 Les réseaux ad hoc

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calculs portables, poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but ultime des réseaux : "offrir l'accès à l'information n'importe où et n'importe quand". Le concept des réseaux mobiles ad hoc essaie d'étendre les notions de mobilité à toutes les composantes de l'environnement. Aucune administration centralisée n'est disponible, ce sont les hôtes mobiles eux mêmes qui forment d'une manière ad hoc l'infrastructure du réseau.

Définition : Un réseau mobile ad hoc, communément appelé MANET (*Mobile Ad hoc Network*), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée. La topologie du réseau peut changer à tout moment ; elle est donc dynamique et imprévisible et implique que la déconnexion des unités est potentiellement très fréquente. Le mot ad hoc en latin signifie "qui va vers ce vers quoi il doit aller", c'est-à-dire "formé dans un but précis", telle qu'une commission ad hoc, formée pour régler un problème particulier.

Un réseau ad hoc peut être modélisé par un graphe $G_t = (V_t, E_t)$, où V_t représente l'ensemble des nœuds et E_t modélise l'ensemble des connexions qui existent entre ces nœuds. Si u et v appartiennent à V_t et $e = (u, v)$ appartient à E_t, cela veut dire que les nœuds u et v sont en mesure de communiquer ensemble directement à l'instant t.

Le modèle de communication, basé sur le médium radio [Die04], peut être caractérisé par un modèle half-duplex 1-port réception Δ-port émission. Half-duplex signifie qu'une interface radio ne peut pas émettre et recevoir simultanément ; 1-port réception signifie qu'elle ne peut recevoir d'information que de la part d'un correspondant à la fois ; et Δ-port émission signifie qu'elle peut émettre un même paquet vers un nombre indéterminé de correspondants grâce à la nature diffusante du lien radio.

Les réseaux ad hoc sont caractérisés par :

- une topologie dynamique : les unités mobiles du réseau se déplacent de façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer à tout instant de manière rapide et aléatoire.
- une bande passante limitée : due à l'utilisation d'un médium de communication partagé ; ce partage fait que la bande passante réservée à un hôte est modeste.

- des contraintes d'énergie : les hôtes mobiles sont alimentés par des sources d'énergie autonomes et limitées comme les batteries. Ce paramètre d'énergie doit être pris en compte au sein de toute architecture de contrôle du système.
- une capacité de mémoire et une puissance de calcul limitées : les équipements utilisés dans les réseaux ad hoc ont des capacités de stockage faibles et des puissances de calcul limitées. Leur sécurité physique est également faible.
- l'absence d'infrastructure : les réseaux ad hoc se distinguent des autres réseaux mobiles par l'absence d'infrastructure préexistante et le non support d'une administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau de manière continue.

Applications des réseaux ad hoc Les applications ayant recours aux réseaux ad hoc [NBCF95, MSK+93, Tri03] couvrent un large spectre, incluant les applications militaires et tactiques, l'enseignement à distance, les systèmes de fichiers répartis, la simulation distribuée interactive, les applications de calcul distribués ou méta-computing, ... D'une façon générale, les réseaux ad hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est trop contraignant, soit parce que difficile à mettre en place, soit parce que la durée d'installation du réseau ne justifie pas de câblage à demeure. La figure 3 montre des exemples d'illustration des réseaux ad hoc.

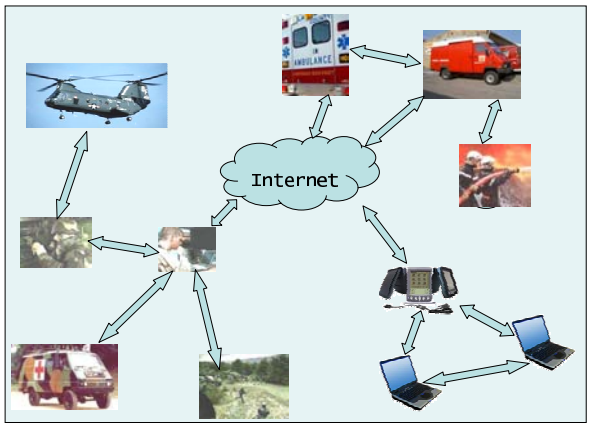


FIG. 3 – Exemples d'illustration des réseaux ad hoc

2.4 Les challenges de sécurité dans les réseaux ad hoc

L'utilisation des liens sans fil rend un réseau ad hoc exposé à des attaques passives comme les écoutes clandestines et des attaques actives. Les écoutes passives permettent à un adversaire d'avoir accès à des informations secrètes ; les attaques actives, quant à elles, peuvent détruire des messages, injecter des messages erronés, modifier des messages et usurper l'identité d'un nœud et par conséquent violer la disponibilité, l'intégrité, l'authentification et la non répudiation, qui sont des éléments de base de la sécurité des réseaux. Les réseaux ad hoc exhibent également de nouvelles vulnérabilités telles que les attaques par brouillage radio, les

attaques par consommation de batteries, la perturbation du routage ad hoc en modifiant les informations de routage.
Les nœuds d'un réseau ad hoc ont une protection physique relativement pauvre et donc ils ont une probabilité non négligeable d'être compromis. Il s'avère ainsi qu'il ne faut pas considérer seulement les attaques malicieuses venant de l'extérieur du réseau ad hoc, mais aussi de l'intérieur par les nœuds compromis. L'utilisation d'une entité centrale dans une solution de sécurité pour un réseau ad hoc pourrait ainsi mener à un état vulnérable du moment où cette entité centrale se trouve compromise.
Un réseau ad hoc est dynamique en raison des changements fréquents de sa topologie et de ses membres. Il est ainsi indispensable que les mécanismes de sécurité puissent s'adapter rapidement à ces changements.
Finalement, un réseau ad hoc peut se composer de centaines, voire de milliers de nœuds. Il est donc important que la solution de sécurité proposée permette le passage à l'échelle.

2.5 Sécuriser les communications multicast dans les réseaux ad hoc

Le déploiement des communications de groupe dans un réseau ad hoc accentue les challenges envers la sécurité du modèle IP multicast, auxquelles s'ajoutent les vulnérabilités des MANETs présentées dans la section 2.4 et de nouvelles vulnérabilités dues à la combinaison du multicast avec l'ad hoc, que nous présentons dans ce qui suit. Telle est la problématique de cette thèse, schématisée dans la figure 4.

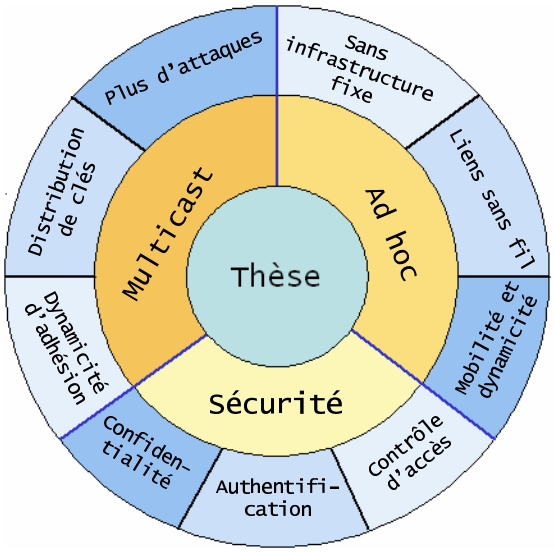


FIG. 4 – Contexte de la thèse

– L'absence d'infrastructure fixe est l'une des principales caractéristiques des réseaux ad hoc. Cette caractéristique élimine toute possibilité de pouvoir établir une référence centralisée afin de concentrer les accès au groupe multicast en un seul point unique capable d'administrer les différents services indispensables pour le bon fonctionnement du groupe. De cette absence d'infrastructure, il découle qu'un modèle centralisé de sécu-

rité, comme celui utilisé dans les infrastructures à clés publiques (PKI [HFPS99]), est difficilement applicable.
- La taille et la dynamicité des groupes multicast peuvent être très importantes dans les réseaux ad hoc. En effet, on ne peut pas contrôler le nombre de membres ni la fréquence d'adhésion au groupe. Ainsi le mécanisme de sécurité doit faire face à la dynamicité et au passage à l'échelle des groupes multicast dans les réseaux ad hoc.
- La mobilité des nœuds ad hoc doit aussi être prise en compte pour assurer des communications sécurisées dans le cadre des MANETs. En effet, quand un nœud se déplace dans le réseau, il peut perdre sa connectivité avec son groupe sans vouloir le quitter. Il ne doit pas être obligé à chaque fois de se ré-authentifier auprès de la source du groupe auquel il appartient. De plus, ce mécanisme de ré-authentification doit être léger et efficace et nécessiter le moins de messages transmis possibles.

La solution la plus appropriée pour assurer les services de sécurité requis au sein d'une architecture de communications de groupe dans les MANETs est l'établissement d'un protocole de gestion de clé de groupe (GKMP : *Group Key Management Protocol*). Ce protocole doit assurer la distribution de la clé de chiffrement de données (TEK) à la source pour chiffrer les données et aux récepteurs pour les déchiffrer. Il est également responsable de sécuriser la distribution de la TEK grâce au déploiement des clés de chiffrement de clés (KEKs) et du renouvellement de clés selon la politique de sécurité adoptée par l'application en question, tout en s'adaptant aux caractéristiques des réseaux ad hoc.

3 Organisation du manuscrit

Afin de présenter notre travail, nous organisons ce manuscrit en trois parties. La première présente l'état de l'art de la sécurité dans les réseaux ad hoc et particulièrement des protocoles de gestion de clé de groupe dans les réseaux ad hoc. La deuxième partie regroupe l'ensemble des contributions que nous avons proposées pour aboutir à une architecture de gestion de clé de groupe dans les MANETS. La dernière partie valide ces propositions à travers une vérification formelle, une implantation dans le cadre d'une application réelle et des simulations avec l'outil de simulation de réseaux NS2.

3.1 Partie I : État de l'art

Cette partie, composée de deux chapitres, a pour but de présenter et de synthétiser les travaux de recherche dans le domaine de la sécurité dans les réseaux ad hoc.

Le chapitre 1, intitulé "La sécurité dans les réseaux ad hoc", définit dans un premier temps les risques et menaces de sécurité dans les réseaux ad hoc, liés à ses caractéristiques et ses vulnérabilités. Nous présentons dans un second temps la problématique d'établissement de la confiance dans ce type de réseaux, considérée comme la clé de voûte de toute architecture de sécurité dans les MANETs.

Dans le chapitre 2, intitulé "Gestion de clé de groupe dans les réseaux ad hoc", nous présentons une taxonomie des protocoles de gestion de clé de groupe dans les réseaux filaires. Nous présentons ensuite notre taxonomie des protocoles de gestion de clé de groupe dans les MANETs, tenant compte des caractéristiques et spécificités de ces réseaux (support de la mobilité, optimisation de l'énergie et conscience du caractère multi-sauts des communications). Nous comparons ensuite les protocoles présentés, selon des métriques que nous avons spécifiées.

3.2 Partie II : Contributions

Dans cette partie, nous décrivons les contributions que nous avons proposées afin d'assurer des communications de groupe sécurisées adaptées au contexte des réseaux ad hoc. Nos contributions sont structurées autour de quatre chapitres.

Protocole de gestion de clé de groupe dans les MANETs - orienté récepteurs

Le chapitre 3 présente un protocole de gestion de clé de groupe dédié à opérer dans les réseaux ad hoc. Le but de ce protocole, dénommé Enhanced BAAL, n'est pas de réaliser une nouvelle architecture de sécurisation des communications de groupe dans les réseaux ad hoc, mais d'améliorer et d'adapter le protocole de gestion de clé de groupe BAAL [CCS00], testé et validé dans les réseaux filaires, au contexte des MANETs. Enhanced BAAL assure une gestion de clés efficace, atténuant le phénomène "1 affecte n" en intégrant un mécanisme de clusterisation dynamique des membres du groupe multicast, au prix d'un surcoût engendré par des opérations intermédiaires de chiffrement et de déchiffrement du flux multicast. La dynamique des récepteurs membres du groupe est également prise en compte dans ce protocole stipulant ainsi qu'à tout événement d'ajout ou de retrait d'une entité dans le groupe, une opération de renouvellement des clés est déclenchée afin de garantir les propriétés de secrets futur et passé du flux de données émis par la source.
La confidentialité des données, l'authentification et le contrôle d'accès des membres adhérents au groupe ainsi que la génération sécurisée des clés du groupe sont les principales fonctionnalités de Enhanced BAAL.

Protocole de gestion de clé de groupe dans les MANETs - orienté sources

Dans le chapitre 4, nous présentons BALADE : un protocole de gestion de clé de groupe, dédié aux MANETs. BALADE assure une distribution de clé efficace et fait face aux limitations que nous avons rencontrées avec le protocole Enhanced BAAL. De par son architecture à une seule clé de chiffrement de données, BALADE est une approche orientée sources du groupe, afin de sécuriser des sessions de groupe dans un modèle de communications multi-sources séquentielles.

Le chapitre 4 est organisé comme suit. La première section présente les motivations et les choix de conception opérés dans ce protocole. Sont présentées ensuite les différentes opérations de gestion et de sécurité réalisées par BALADE. Finalement, nous décrivons le fonctionnement du protocole au cours d'une session de communication de groupe (initialisation, ajout et retrait d'entités, authentification et contrôle d'accès et renouvellement des clés).

Algorithme de clusterisation dynamique

Le chapitre 5 aborde un élément important de la conception de protocoles de gestion de clé de groupe dans les MANETs : la clusterisation dynamique des groupes. Après avoir mis en évidence les limites de la procédure de clusterisation dynamique utilisée dans les deux protocoles de gestion de clés présentés, nous avons élaboré un algorithme de clusterisation dynamique du groupe multicast, dénommé OMCT (*Optimized Multicast Clustering Tree*). Cet algorithme assure un service de distribution de clés prenant en compte la mobilité et la localisation des nœuds, tout en optimisant la consommation de l'énergie et de la bande passante.

Nous décrivons dans la première partie du chapitre l'algorithme OMCT et nous le validons à travers des analyses et des discussions. Ensuite, nous traitons de l'intégration de OMCT dans BALADE et décrivons l'impact de ce couplage sur le processus de gestion et de distribution de clés. Dans la dernière section du chapitre, nous étudions la possibilité d'intégrer la technique de relais multipoints dans la conception d'OMCT et nous discutons les apports d'une telle intégration.

Spécification de BALADE avec OMCT

Avant d'entamer la partie validation et implantation du protocole BALADE couplé avec OMCT, nous procédons à une phase indispensable qui est sa spécification semi-formelle. Cette phase permet d'enlever les ambiguïtés éventuelles dues à la description du protocole en langage naturel. La spécification du couplage BALADE / OMCT consiste à la spécification des entités qui participent au fonctionnement du protocole, les messages échangés par ces entités et les connaissances statiques et/ou dynamiques que peuvent avoir ces entités. Le chapitre 6 est consacré à cette spécification.

3.3 Partie III : Validations et expérimentations

Cette partie est consacrée à la validation de nos principales contributions. Leur applicabilité est évaluée selon trois approches complémentaires.

Spécification et validation formelle

Dans le chapitre 7, nous traduisons la spécification de notre protocole BALADE combiné avec OMCT présentée dans le chapitre 6, en HLPSL [CCC+04] (*High Level Protocol Specification Langage*) : un langage de spécification formelle dédié aux protocoles de sécurité. La validation de notre protocole, dont le but est de détecter les éventuelles failles de sécurité qu'une attaque malicieuse peut cibler, est ensuite présentée. Cette validation a été réalisée avec l'outil de validation des protocoles de sécurité AVISPA [Tea05] (*Automated Validation of Internet Security Protocols and Applications*).

Implantation

La mise en œuvre effective de BALADE est montrée via son implantation dans le cadre d'une application de Jukebox coopérative, appelée JDukebox. Le chapitre 8 comprend la présentation de cette application réalisée au sein de notre équipe de recherche. Il comprend notamment la description du module BALADE et de son intégration avec l'application DJukebox.

Évaluation de performances

Le chapitre 9 présente l'évaluation de performances de BALADE avec l'outil de simulation des réseaux NS2. Pour cela, nous procédons comme suit. La première section présente l'évaluation des performances de l'algorithme OMCT. Ensuite, nous évaluons l'impact du modèle de mobilité sur le couplage de BALADE avec OMCT. La section suivante s'intéresse à la validation de l'intégration de la technique des relais multipoints dans OMCT. Finalement, une

étude comparative de BALADE avec d'autres protocoles de gestion de clé de groupe dans les MANETs est présentée.

Nous terminons ce manuscrit par une synthèse des contributions que nous avons apportées, ainsi qu'une description des travaux futurs.

Première partie

État de l'art

Chapitre 1

La sécurité dans les réseaux ad hoc

Sommaire

1.1 Introduction

L'essor des technologies sans fil offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente des moyens de la communication sans fil a permis le traitement de l'information à travers des unités de calcul portables qui ont des caractéristiques particulières et accèdent au réseau à travers une interface de communications sans fil. Comparé à un environnement statique, ce nouvel environnement mobile permet aux unités de calcul une libre mobilité et ne pose aucune restriction sur la localisation des usagers. Ces environnements mobiles offrent également une grande flexibilité d'emploi. En particulier, ils permettent la mise en réseau de sites dont le câblage serait trop onéreux à réaliser, voire même impossible. Un réseau ad hoc est une illustration de ces réseaux mobiles. Il est défini comme une collection d'entités mobiles, interconnectées par une technologie sans fil et formant un réseau temporaire sans l'aide d'aucune administration ou de support fixe. Aucune supposition ou limitation n'est faite sur la taille du réseau, ni sur la mobilité de ses nœuds.

La sécurité constitue actuellement l'un des principaux obstacles à un large déploiement des réseaux ad hoc. Sécuriser un réseau ad hoc revient à instaurer les différents services de sécurité dans ce réseau, tout en prenant en compte ses différentes caractéristiques.

Dans ce chapitre, nous présentons dans un premier temps les menaces de sécurité que court un réseau ad hoc dû à ses caractéristiques. Nous abordons ensuite la problématique d'établissement de confiance dans les réseaux ad hoc et nous présentons les différentes approches assurant l'authentification : la clé de voûte de toute architecture de sécurité au sein des MANETs.

1.2 Menaces et attaques dans les réseaux ad hoc

Gayraud et al. [GND+03] ont mené une étude d'analyse de risque dans les réseaux ad hoc. Cette étude est une corrélation entre l'analyse des besoins et exigences des réseaux ad hoc et les risques issus de leurs vulnérabilités. Une liste des attaques les plus probables est ainsi dressée dans cette section.

Les attaques de dénis de service (DoS)

Les attaques DoS apparaissent comme les attaques les plus faciles à réaliser par un attaquant, mettant en péril la disponibilité des membres du groupe et plus particulièrement des entités qui jouent le rôle de serveurs ou de contrôleurs au sein du réseau ad hoc. Les attaques DoS les plus connues dans un MANET sont les suivantes :

- Brouillage du canal radio pour empêcher toute communication.
- Tentative de débordement des tables de routage des nœuds servant de relais.
- Non-coopération d'un nœud au bon fonctionnement du réseau, dans le but de préserver son énergie. Cette attaque est connue sous le nom de "*Selfishness*" et peut être détectée grâce à des mécanismes de réputation et de détection de comportements égoïstes (Nuglets [BH01], Confidant [BB02], Core [MM02], ...).
- Tentative de gaspillage de l'énergie des nœuds ayant une autonomie de batterie faible. Cette attaque est connue sous le nom de "*Sleep Deprivation Torture*" [Fra02] et consiste à faire en sorte que le nœud cible soit obligé de rester en état d'activité et ainsi de lui faire consommer toute son énergie.

- Dispersion et suppression du trafic en attaquant les mécanismes de routage. L'attaque *Wormhole* [MM03] fait partie de ce type d'attaques et est particulièrement difficile à détecter ou à prévenir. Elle consiste à ce qu'un nœud malicieux achemine tous les paquets du réseau via un tunnel privé partagé avec un autre attaquant, offrant à une source un meilleur chemin (érroné) pour atteindre sa destination et éliminant ainsi la possibilité de découverte des chemins fiables dans le réseau.
- Attaques des mécanismes de sécurité eux même.

Les attaques passives d'écoute et d'analyse de trafic

Ces attaques, appelées aussi "*Sniffing*", consistent à écouter le réseau dans lequel transitent des paquets de données et récupérer à la volée et illégalement des données qui peuvent être confidentielles. Les attaques d'écoute et d'analyse de trafic sont d'autant plus dangereuses dans les réseaux sans fil tels que les MANETs. En effet, les ondes radio-électriques ont intrinsèquement une grande capacité à se propager dans toutes les directions avec une portée relativement grande, facilitant ainsi à une personne non autorisée d'écouter le réseau et d'analyser le trafic. Ces attaques constituent une menace certaine pour la confidentialité des données, ainsi que pour l'anonymat des utilisateurs.

L'usurpation de l'identité d'un nœud

L'usurpation d'identité, appelée également mystification, consiste à se faire passer pour une entité connue et fiable auprès des autres nœuds, afin de récupérer illégalement des informations confidentielles ou d'injecter des messages dans le réseau. Ce type d'attaques met en péril l'authentification et le contrôle d'accès des membres des réseaux ad hoc.

Les attaques physiques d'un élément valide du réseau

Ces attaques compromettent des nœuds valides du réseau (destruction, altération ou changement physique d'un composant) et s'avèrent être des attaques particulièrement dangereuses dans les réseaux ad hoc.

1.3 Établissement de confiance dans les réseaux ad hoc

L'authentification permet à un nœud de s'assurer de l'identité des nœuds avec lesquels il communique. Sans authentification, un adversaire peut communiquer avec des nœuds du réseau et ainsi bénéficier de ressources auxquelles il n'est pas autorisé à accéder.

Pour que deux nœuds d'un réseau ad hoc puissent communiquer ensemble, la confiance mutuelle se révèle très importante ; en effet, la confiance doit être un pré requis nécessaire à l'établissement de la communication entre deux nœuds du réseau ad hoc pour assurer en conséquence les services de confidentialité et de contrôle d'accès. Afin de garantir l'échange, le service de confiance s'organise selon deux phases distinctes : l'établissement de la confiance, phase initiale pour définir les conditions de l'échange et la gestion de la confiance en assurant son maintien. Pour accomplir l'établissement de la confiance, trois étapes doivent se succéder : la distribution du secret commun, l'établissement d'un canal sûr obtenu grâce au secret commun précédent et l'échange de clés de chiffrement destinées à la confidentialité de la session, si nécessaire.

Le service d'authentification se révèle ainsi indispensable pour pouvoir faire communiquer deux entités appartenant à un réseau ad hoc, sans infrastructure fixe. Dans cette partie, nous présentons les principales approches d'authentification dans les MANETs.

1.3.1 La technique de cryptographie à seuil

Cette approche présentée dans [ZH99] a pour but de remédier au problème d'absence d'infrastructure des réseaux ad hoc et des réseaux de capteurs sans fil, qui rend l'utilisation d'une infrastructure à clé publique PKI (*Public Key Infrastructure*) très difficile.

Pour établir des communications sécurisées entre les membres d'un réseau filaire muni d'une PKI, chaque nœud détient une clé publique et une clé privée que l'autorité de certification CA (*Certification Authority*) lui délivre. Le CA, à son tour, possède une paire de clés, publique et privée (K, k). Le CA est toujours disponible dans le réseau puisque les clés privées et publiques des nœuds doivent être mises à jour périodiquement pour diminuer le risque d'attaques malicieuses du réseau. Le CA doit aussi pouvoir remettre en question la clé publique d'un nœud s'il ne lui fait plus confiance.

Dans un réseau ad hoc, avoir un seul CA présente un point de vulnérabilité. S'il n'est plus disponible, les nœuds ne pourront plus prouver l'authenticité des clés publiques des autres nœuds et ne pourront plus mettre en place de communications sécurisées entre eux. Les attaquants peuvent aussi utiliser ce point de vulnérabilité pour compromettre le bon fonctionnement de tout le réseau.
Une première solution à ce problème consiste à répliquer le CA dans le réseau ad hoc, cette solution apporte plus de disponibilité dans le sens où on est sûr qu'au minimum un CA est en ligne à un instant donné. Cependant cette même solution présente plus de vulnérabilités en terme de sécurité parce qu'elle réplique aussi la possibilité de compromettre un CA et par la suite tout le réseau ad hoc.

La cryptographie à seuil [ZH99] se propose de remédier à ce problème : le nouveau service de gestion de clés ayant la configuration $(n, t+1)$ consiste à avoir n nœuds spéciaux qu'on appelle serveurs, présents dans le réseau ad hoc et qui partagent la capacité de générer des certificats pour les autres membres du réseau. Chaque serveur a sa propre paire de clés, publique et privée et enregistre les clés publiques de tous les nœuds du réseau, en particulier, celles des autres serveurs. Ceci permet aux nœuds serveurs d'établir des liens sécurisés entre eux.

Yi et al. proposent dans [YK02] de distribuer la confiance à des nœuds qui ont relativement une bonne sécurité physique et une bonne puissance de calcul, surtout dans un environnement hétérogène constitué de nœuds de différentes caractéristiques. Ces nœuds sont appelés MOCAs (MObile Certificate Authority). Dans le cas de notre configuration $(n, t+1)$, les n serveurs partagent la capacité de signer les certificats pour les autres nœuds du réseau. La clé privée k de tout le service est divisée en n secrets partagés ($s1$, $s2$, ..., sn), un secret étant connu d'un seul serveur. La figure 1.1 illustre cette configuration.

Chaque serveur génère une signature partielle du certificat du nœud et l'envoie à un combinateur, qui a besoin de rassembler au moins $t+1$ signatures partielles pour générer la signature complète. Le nombre maximum de serveurs compromis dans n'importe quelle période de temps est supposé égal à t ; avec t serveurs compromis, le combinateur est capable de générer une signature valide. Zhou et al. [ZH99] supposent que $n >= 3t + 1$.
Le combinateur peut aussi vérifier la validité d'une signature partielle envoyée par un serveur et quand il s'aperçoit qu'une signature est erronée, il la rejette et continue sa collecte de $t+1$

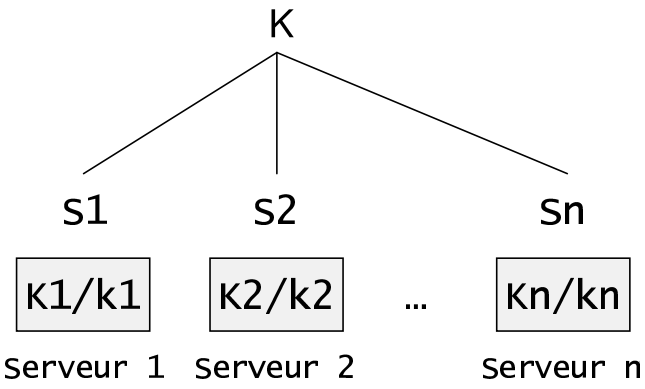


FIG. 1.1 – Configuration du service de gestion de clés

signatures valides. La figure 1.2 illustre cette opération de construction de signature : étant donnée une configuration (3,2), le serveur 2 a été compromis et n'a pas pu envoyer sa signature partielle (SP) et le combinateur a pu générer la signature du certificat du membre m.

La polémique du choix de t est détaillée dans [YK02]. Plus t est grand, plus grande est la sécurité contre d'éventuelles attaques mais en même temps plus important est le surcoût en terme de communication.

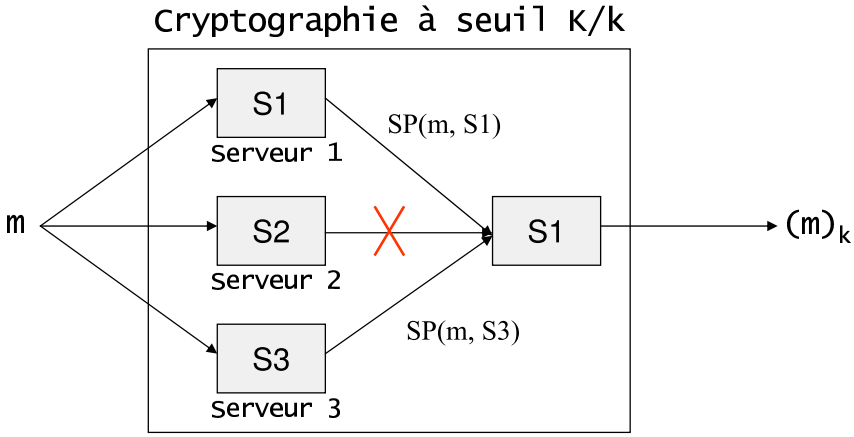


FIG. 1.2 – La technique de cryptographie à seuil avec le paramétrage (3,2)

Le combinateur, indispensable pour la génération des signatures des nœuds du réseau, peut à son tour être attaqué et par conséquent devenir un point de vulnérabilité de tout le système de sécurité du réseau. LEGRAND ET AL. proposent ainsi une réplication du combinateur en plusieurs CA : on obtient ainsi une architecture coopérative où des combinateurs locaux peuvent se former à la volée autour du nœud en question et lui générer sa signature [Leg03].

YI ET AL. présentent un protocole [YK02] de certification appelé MP (MOCA Certification Protocol). Selon ce protocole, les clients envoient des messages *Send Request* (SREQ) par inon-

dation, chaque MOCA recevant ce message, répond par un message Certif Response (CREP) (protocole analogue au protocole de routage AODV), contenant une signature partielle. Quand le client collecte t CREP valides, il peut calculer sa signature. Ce protocole n'utilise pas de combinateur, offrant ainsi un meilleur niveau de sécurité. Pour remédier au problème d'inondation des SREQ (tous les MOCAs reçoivent des copies de SREQ et répondent tous par des CREP tandis que le client n'a besoin que de t CREP), YI ET AL. proposent la technique B-Unicast. Cette solution permet à un nœud d'envoyer des requêtes en unicast à t MOCAs s'il détient dans sa table de routage assez d'informations concernant leurs routes. Sinon, le nœud se voit obligé de retourner à la solution plus contraignante d'inondation complète du réseau.

1.3.2 L'infrastructure à clé publique auto-organisée

HUBAUX ET AL. proposent dans [HBC01] une infrastructure à clés publiques auto-organisée et auto-gérée au sein du réseau ad hoc. Dans cette architecture, chaque nœud établit des certificats pour les nœuds en qui il a fait confiance. Lorsque deux éléments d'un réseau ad hoc veulent communiquer sans connaissance au préalable l'un de l'autre, ils s'échangent leur liste de certificats et essayent de créer une chaîne de confiance entre eux. Par exemple, si les nœuds A et B veulent communiquer ensemble et font tous les deux confiance au nœud C, une chaîne de confiance entre A et B peut être établie via C (comme dans le cas de PGP [Zim95] qui stipule que "les amis de mes amis sont mes amis").

Des mécanismes de construction des bases de données locales contenant les certificats sont utilisés dans [HBC01], de sorte que deux nœuds du réseau peuvent établir une chaîne de confiance entre eux avec une grande probabilité, même si la taille des bases de données locales est petite comparée au nombre total de nœuds dans le réseau. Le modèle relationnel de confiance entre les utilisateurs est représenté par un graphe G(V,E). V et E sont l'ensemble des sommets (utilisateurs) et l'ensemble des arcs (certificats) du graphe respectivement. Ainsi, si entre deux sommets i et j existe un arc dans le graphe de confiance, c'est que le nœud i a généré un certificat pour le nœud j. L'existence d'une chaîne de confiance entre deux nœuds du réseau est ainsi représentée par un chemin direct entre les deux sommets du graphe correspondants à ces deux nœuds. La figure 1.3 illustre ce processus d'établissement d'une chaîne de confiance entre deux nœuds u et v.

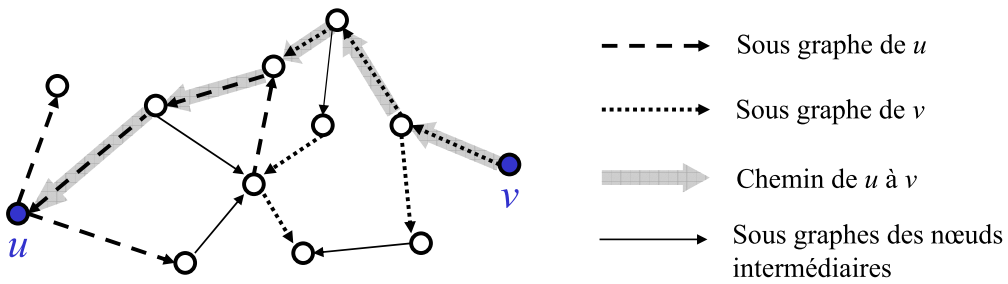


FIG. 1.3 – Graphe de confiance dans [HBC01]

Cette technique d'authentification distribuée reste à garantie probabiliste, du fait que

l'existence d'un chemin direct entre deux nœuds voulant communiquer ensemble n'est pas assurée. De plus, le stockage distribué des certificats des nœuds du réseau engendre un surcoût important, qui met en question l'applicabilité à large échelle de l'approche. De plus, des membres malicieux peuvent générer de faux certificats et les intégrer dans le graphe de confiance. Pour remédier à ce problème, HUBAUX ET AL. proposent l'utilisation de métriques d'authentification, qui permettent d'évaluer l'authenticité d'un certificat et la chaîne de confiance à laquelle il appartient. Le nombre de chaînes de certificats disjointes entre deux membres du graphe de confiance peut être un exemple de métriques d'authentification [HBC01]. Il est à noter cependant que les approches basées sur PGP ("les amis de mes amis sont mes amis") sont plus appropriées pour de petites communautés [MM03], car l'authenticité des certificats et des clés peut être assurée avec un plus haut degré de confiance.

L'approche proposée par LUO ET AL. dans [LL00] est basée également sur le principe de PGP et consiste à générer le certificat d'un membre du réseau ad hoc par ses voisins de façon coopérative et selon son comportement. Les services de certification, tels que la génération, le renouvellement ou la révocation sont distribués à tous les nœuds du réseau. Ainsi, similairement à la technique de cryptographie à seuil, la clé privée de l'autorité de certification est partagée entre un ensemble de membres du groupe, responsables de délivrer les certificats aux nœuds "honnêtes" et d'élargir le graphe de confiance dans le réseau ad hoc. Les membres voisins ayant des relations de confiance entre eux s'entraident pour acheminer leurs paquets et détecter d'éventuelles attaques dans le réseau. Les nœuds n'ayant pas encore reçu de certificat dans le réseau sont considérés comme des intrus potentiels.

1.3.3 L'accord de clé (*Key Agreement*) dans les MANETs

Le contexte de cette approche est un petit groupe de personnes dans une conférence, présentes ensemble dans une salle pour une réunion ad hoc et faisant partie d'un système asymétrique de chiffrement ; ces personnes veulent s'échanger des données secrètement durant la durée de la réunion.

Le principe de ce protocole consiste, sachant que tous les nœuds présents ont confiance les uns dans les autres, à partager un mot de passe faible à partir duquel un autre mot de passe sera généré et constituera la clé de session du groupe. Ce protocole tel qu'il est présenté par [AG00] doit avoir les propriétés suivantes :

- Secret : seules les personnes qui connaissent le premier mot de passe doivent pouvoir déduire la clé de session résultante.
- Accord contribuant : la clé de session générée doit être composée des contributions des différents participants à la session.
- La tolérance aux attaques d'insertion de messages dans le réseau.

ASOKAN ET AL. présentent un protocole générique d'authentification EKE (Encrypted Key Exchange). Les intervenants dans EKE sont deux nœuds A et B dans un réseau ad hoc qui veulent se mettre d'accord sur une clé forte de session K, de sorte qu'un attaquant ne puisse pas connaître K et ne puisse pas non plus attaquer le mot de passe faible de départ P (attaque par dictionnaire). Le nœud A détient une paire de clés publique et privée (E_A, D_A). Il génère en plus deux valeurs aléatoires $Challenge_A$ et S_A. Le nœud B génère trois valeurs aléatoires R, $Challenge_B$ et S_B. La clé de session entre A et B est K $= f(S_A,\ S_B)$; f étant une fonction à sens unique et h est une fonction publique. Les échanges de clés entre A et B sont les suivants :

(1) A –> B : A, P(E_A)

Le nœud A envoie à B son identité et sa clé publique chiffrée avec le mot de passe P.

(2) B –> A : P(E_A(R))
B envoie à A la valeur R chiffrée avec la clé publique de A, le tout chiffré avec le mot de passe P.

(3) A –> B : R($Challenge_A$, S_A)
A délivre à B sa contribution S_A, ainsi que la valeur $Challenge_A$, chiffrées avec R.

(4) B –> A : R($h(Challenge_A)$, $Challenge_B$, S_B)
À cette étape, le nœud B envoie sa contribution S_B au nœud A, en plus des valeurs ($h(Challenge_A)$ et $Challenge_B$. Les deux entités A et B peuvent ainsi calculer la clé K = $f(S_A, S_B)$.

(5) A –> B : R($h(Challenge_B)$)
Ce message permet à B de s'assurer que A détenait bien le mot de passe faible P qu'il a utilisé pour déchiffrer R et par suite $Challenge_B$.

Ensuite, une extension du protocole EKE est proposée dans [AG00] pour qu'il soit un protocole multi-parties, la seule contrainte est qu'un leader doit déclencher les opérations d'authentification et d'échanges de messages ; ce leader constitue ainsi un point de vulnérabilité du réseau ad hoc. Ce nouveau protocole tel qu'il a été présenté ne satisfait pas la contrainte d'accord contribuant, car c'est le leader qui calcule la clé de session globale et la diffuse à tous les autres nœuds. ASOKAN ET AL. ont apporté des modifications au protocole de base EKE [AG00] pour avoir un protocole multi-parties qui permet à tous les intervenants de contribuer à la génération de la clé de session K.

Le protocole de DIFFIE-HELMANN pour l'échange de clés [DH76] peut être adapté pour assurer un secret partagé parfait entre les différents participants à un réseau ad hoc, connaissant un mot de passe faible P. Les échanges de messages entre les membres $M_1, \ldots, M_n$ pour établir la clé secrète $K = g^{S_1 S_2 \ldots S_n}$ sont les suivants (S_i est la contribution du membre M_i) :

(1) M_i –> M_{i+1} : $g^{S_1 S_2 \ldots S_i}$, i=1,...,n-2, en séquence
Cette étape nécessite n-2 envois de messages ; à la fin de cette étape, M_{n-1} aura reçu $S_1, S_2, \ldots,$ S_{n-1} et pourra ainsi calculer $\pi = g^{S_1 S_2 \ldots S_{n-1}}$.

(2) M_{n-1} –> Tous : $\pi = g^{S_1 S_2 \ldots S_{n-1}}$, en diffusion

(3) M_i –> M_n : P(C_i), $i = 1, ..., n-1$, en parallèle, avec $C_i = \pi^{\hat{S}_i / S_i}$ et $\hat{S}_i$ est un facteur d'aveuglement choisi aléatoirement par M_i
Chaque M_i enlève de π sa contribution S_i et ajoute un facteur d'aveuglement $\hat{S}_i$.

(4) M_n –> M_i : $(C_i)^{S_n}$, $i = 1, ..., n-1$, en parallèle
M_n décrypte (C_i), lui ajoute son S_n et l'envoie au M_i correspondant. À ce stage, tous les participants pourront calculer $K = g^{S_1 S_2 \ldots S_n}$.

(5) M_i –> Tous : M_i, K(M_i, H(M_1, M_2,...,M_n)), pour i donné, en diffusion
Ce message est envoyé pour une simple vérification. H(M_1, M_2,...,M_n) est un terme bien spécifié, connu par tous les membres du groupe et qui leur permet de vérifier s'ils ont la même

vue sur la composition du groupe.

Une amélioration de Diffie-Helmann concernant le nombre de messages de communication est présentée dans [AG00], en arrangeant tous les participants dans un hypercube. L'idée de base de ce protocole est illustrée dans la figure 1.4 ; avec quatre participants A, B, C et D, voulant se mettre d'accord sur une clé de session. Chaque participant i a une adresse de deux bits et génère une contribution S_i. Dans une première étape, les nœuds A et B exécutent Diffie-Helmann pour deux participants, ils parviennent ainsi à calculer $S_{AB} = g^{S_A S_B}$; en même temps, C et D calculent $S_{CD} = g^{S_C S_D}$.
La deuxième étape consiste à exécuter Diffie-Helmann entre A et C et B et D, tout en utilisant comme contributions les clés calculées à l'issue de la première étape. Ainsi, à la fin de la deuxième étape, les quatre participants détiennent la même clé de session $S_{ABCD} = g^{S_{AB} S_{CD}}$.

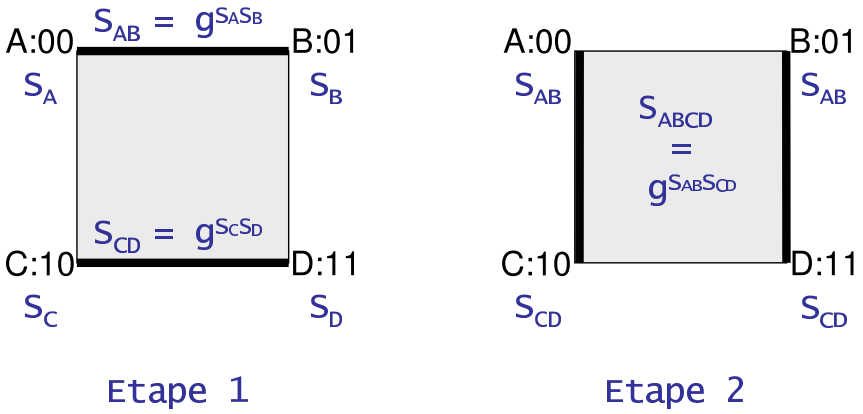


FIG. 1.4 – Échanges de clés Diffie-Helman dans un D-Cube

Si le nombre de participants est évalué à $n = 2^d$ participants, chaque participant se voit attribué un sommet dans un hyper-cube de dimension d. Le protocole procède en d étapes d'échanges de clés avec le même principe présenté précédemment. Après les d étapes, tous les participants auront la même clé de session.

Tous ces protocoles résolvent le problème d'authentification sans avoir besoin d'infrastructures additionnelles ou de canaux physiques de communication sécurisée, ce qui les rend adaptés à la nature des réseaux ad hoc.

1.3.4 Les identificateurs cryptographiques

Les identificateurs cryptographiques [MC02] sont générés et détenus par les nœuds d'un réseau spontané, afin de prouver leur identité auprès des nœuds avec lesquels ils communiquent, dans un environnement sans aucun tiers de confiance. Ces identificateurs sont statistiquement uniques et cryptographiquement vérifiables, ce qui signifie que de par leur nature, il est très peu probable que deux entités aient le même identifiant et qu'il est possible de vérifier la validité de l'identifiant présenté par une entité grâce à des techniques cryptographiques. L'identité cryptographique, notée CBID (*Crypto Based Identifier*), est définie par :

$$CBID = hmac\_sha1\_128(sha1(imprint), sha1(PK))$$

Où :
- PK est la clé publique du créateur de l'identifiant,
- *imprint* est une valeur aléatoire de 64 bits,
- hmac et sha1 sont deux fonctions de hachage.

L'idée de base de ces identificateurs est d'avoir une forte relation cryptographique avec leurs composants (clés privée et publique). Un nœud affirme son identité aux autres membres du réseau en prouvant qu'il détient la clé privée associée à sa clé publique, utilisée lors de la génération de son CBID. Comme exemple, pour prouver son identité, un nœud A envoie à un nœud B le message suivant :

$$A \rightarrow B \ : \ Clé\_pub_A,\ imprint,\ \{CBIDA\}_{clé\_priv_A}$$

Ce message contient la clé publique de A et l'entier *imprint* utilisés pour la création du CBID de A, ainsi que ce CBID chiffré avec la clé privée de A. Pour s'assurer de l'identité de son correspondant A, le nœud B, calcule le CBID du nœud A en utilisant sa clé publique et l'entier *imprint*, ensuite déchiffre le CBID de A envoyé dans le message avec la clé publique. B affirme que le nœud A détient la clé privée correspondant à sa clé publique si les deux CBID sont conformes.

Cette technique d'authentification ne doit pas compter sur une tierce partie, à savoir une PKI globale ou un serveur central de distribution de clés. Cette contrainte implique que deux entités qui ne se connaissent pas ne peuvent pas établir une confiance mutuelle entre eux et par conséquent, ne peuvent pas communiquer ensemble de façon sécurisée. L'authentification d'un nouveau nœud n'est donc pas possible. Seuls les nœuds qui se connaissent au préalable peuvent s'identifier et communiquer.

CASTELLUCIA ET AL. proposent d'intégrer la propriété de CBID dans les adresses IPv6 [MC02]. Cette proposition profite de l'avantage qu'aucune information supplémentaire n'est requise pour assurer l'acheminement des paquets aux destinataires. Ces adresses, de 128 bits, sont construites en combinant :
- les premiers 64 bits du préfixe du réseau,
- les derniers 64 bits correspondent au CBID de l'hôte, noté CBA (*Crypto Based Address*). Le CBA remplace l'identificateur de l'interface de l'adresse IPv6. Le CBA est ainsi généré de la façon suivante :

$$CBA = hmac\_sha1\_64(sha1(imprint), sha1(PK))$$

L'identificateur de 128 bits, ainsi généré, est routable évitant ainsi toute autre information de routage dans les paquets. Quand un nœud se déplace, il est amené à changer de préfixe de réseau, tandis que la partie CBA de son identificateur reste inchangée.

1.3.5 La technique de Resurrecting Duckling

Cette technique [SA99] est basée sur une métaphore inspirée de la biologie, qui décrit le comportement d'un canard sortant de son œuf et reconnaissant comme sa mère le premier objet mobile qui émet un son. Ce phénomène est appelé imprinting. De façon similaire, une entité reconnaît comme son propriétaire (son contrôleur) la première entité qui lui envoie une clé secrète (tout au long de la session). L'envoi de la clé secrète entre un équipement et son propriétaire est effectué directement (via un contact électrique), évitant ainsi toute opération cryptographique ou ambiguïté concernant les identités des intervenants, mais aussi

rendant la technique de resurrecting duckling restreinte à un type bien spécifique d'applications et inadaptée à un large déploiement d'un réseau ad hoc. Le contrôleur d'un équipement lui envoie, de façon sécurisée via la clé secrète, toute information qui détermine son comportement avec les autres membres du réseau (politique de sécurité, liste de contrôle d'accès, ...). Un équipement peut ainsi communiquer avec les autres entités du réseau, mais ne peut pas être contrôlé par eux.

L'application cible de cette technique, détaillée dans [SA99], est une application médicale dans laquelle les équipements sont par exemple des thermomètres détenus par les patients et les contrôleurs sont les PDAs des médecins.

1.4 Conclusion

Un réseau ad hoc est un environnement hostile, qui apporte plusieurs défis de sécurité, dus à ses caractéristiques et à ses spécificités (liens sans fil, capacités limitées, ...). Dans ce chapitre, nous avons présenté les attaques de sécurité fortement probables dans les réseaux ad hoc. Ensuite, nous avons étudié la problématique d'établissement de la confiance dans les MANETs, ainsi que les différentes approches d'authentification dans ces réseaux.

Le déploiement des communications de groupe sécurisées dans un réseau ad hoc induit également de nouvelles épreuves à prendre en compte. En effet, aux contraintes de sécurité des réseaux ad hoc, s'ajoutent les vulnérabilités du modèle IP multicast qui, de par sa nature, élimine toute possibilité d'identification des membres du groupe ou de confidentialité de données. Dans le chapitre suivant, nous présentons un état de l'art des protocoles de sécurisation des communications de groupe dans les réseaux ad hoc et nous les discutons selon des métriques bien spécifiques.

Chapitre 2

Gestion de clé de groupe dans les réseaux ad hoc

Sommaire

2.1 Introduction

La gestion de clé de groupe est la fonctionnalité fondamentale de toute architecture de sécurisation des communications de groupe. En effet, elle permet la distribution de la clé du groupe à la source pour chiffrer le flux multicast et aux récepteurs pour le déchiffrer, assurant ainsi la confidentialité des données. L'authentification et le contrôle d'accès des membres du groupe sont également assurées, du fait que seuls les membres autorisés qui détiennent la clé du groupe sont capables d'accéder aux données multicast émises par la source.

Plusieurs protocoles de gestion de clé de groupe ont été élaborés, la majorité étant dédiée aux réseaux filaires. Quelques approches de gestion de clés de groupe ont ciblé les environnements ad hoc au cours des dernières années. Nous les présentons dans ce chapitre et nous discutons leurs avantages et leurs limites. Ce chapitre est organisé de la façon suivante. Nous décrivons en premier lieu les protocoles de gestion de clé de groupe dans les réseaux filaires. Ensuite, nous présentons une taxonomie des protocoles de gestion de clé de groupe dans les réseaux ad hoc, adaptée aux caractéristiques et spécificités d'un tel environnement. Finalement, nous discutons les protocoles proposés dans les MANETs, selon des métriques et des propriétés que nous avons définies.

2.2 Gestion de clé de groupe dans les réseaux filaires

Plusieurs taxonomies des protocoles de gestion de clés dans les réseaux filaires ont été publiées ([CS05, BCF05a], ...) ; elles les divisent en général en trois approches : centralisée, décentralisée et distribuée (cf. Figure 2.1).

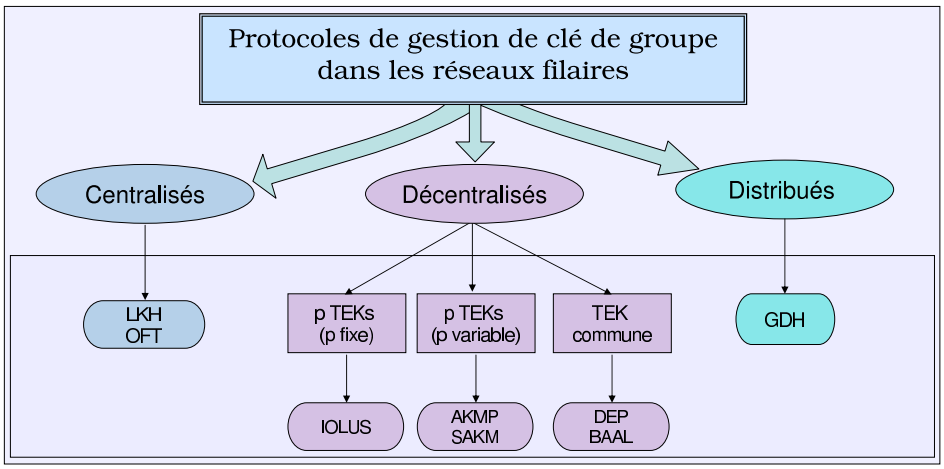


FIG. 2.1 – Taxonomie des protocoles de gestion de clés dans les réseaux filaires

2.2.1 Approche centralisée

Les architectures centralisées (LKH [WGL98], OFT [MS98], ...) font recours à un unique serveur de clés. Cette entité serveur est responsable de la génération, de la distribution et du renouvellement de la clé du groupe. Cette approche ne résiste pas au passage à l'échelle, parce

que le renouvellement de la clé du groupe suite à un événement d'addition ou de retrait d'une entité affecte tous les autres adhérents au groupe multicast (phénomène "1 affecte n"). De plus, un seul serveur de clés représente un goulot d'étranglement cible d'attaques potentielles.

Le protocole LKH (*Logical Key Hierarchy*) [WGL98] a opté pour cette architecture centralisée autour d'un serveur de clés. Un groupe multicast est défini dans LKH par un triplet (U, K, R), correspondant à un graphe orienté et acyclique (arbre de distribution de clés). U définit l'ensemble des utilisateurs membres du groupe, K est constitué par l'ensemble des clés du groupe et R définit les relations entre U et K (ensemble de clés détenues par chaque membre). La racine de l'arbre LKH correspond à la clé du groupe. Les feuilles de l'arbre LKH correspondent aux membres du groupe. Les nœuds intermédiaires sont constitués par des clés logiques. Une illustration d'un arbre de distribution de clés de LKH est présentée dans la figure 2.2.a. Dans cet exemple, il y a 4 membres U_1, U_2, U_3 et U_4. La clé du groupe est notée k_{1234} et est la racine de l'arbre.

Un membre connaît toutes les clés de son chemin vers la racine de l'arbre. Après un événement d'ajout (ou de retrait) d'entité dans le groupe, un processus de renouvellement de clés est déclenché et consiste à renouveler toutes les clés allant du nœud à joindre (respectivement à supprimer) jusqu'à la racine de l'arbre (clé du groupe). Plusieurs schémas de distribution de ces nouvelles clés peuvent être utilisés (orienté utilisateur, orienté clé, orienté groupe), mais souffrent tous du phénomène "1 affecte n". La figure 2.2.b illustre le renouvellement des clés LKH suite à l'ajout de l'entité U_5 dans le groupe multicast.

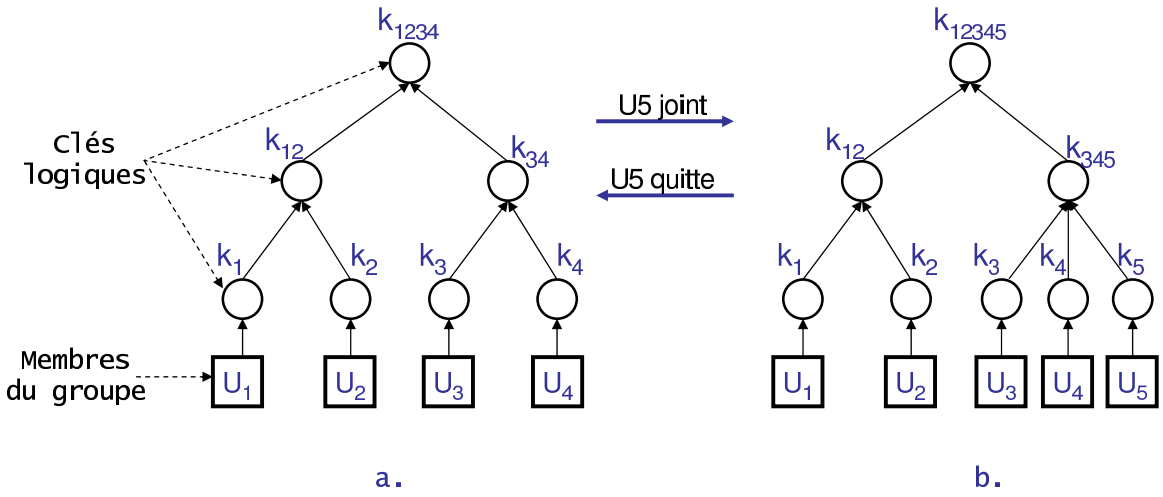


FIG. 2.2 – Arbre de distribution de clés dans LKH

2.2.2 Approche décentralisée

Cette approche divise le groupe multicast en des sous-groupes ou clusters. Les membres d'un cluster partagent une clé de session locale, gérée par un contrôleur local, atténuant ainsi le phénomène "1 affecte n". Quand un membre rejoint ou quitte le groupe, le renouvellement de la clé de session n'affecte que le cluster local et non plus tous les membres du groupe. Nous avons divisé les protocoles décentralisés en trois familles, selon le nombre de clés de chiffrement de données (TEKs) qu'ils utilisent.

– **Protocoles à p TEKs (p fixe)** : dans cette famille de protocoles, un cluster du groupe multicast partage une clé locale de chiffrement de données (TEK), ce qui requiert des opérations de déchiffrement et de rechiffrement du flux multicast, en passant d'un cluster à un autre. Ces protocoles à clusterisation statique ne sont pas flexibles et ne peuvent pas s'adapter à la dynamicité du groupe multicast dans Internet.
IOLUS [Mit97] appartient à cet ensemble de protocoles. Il divise le groupe multicast hiérarchiquement en des sous-groupes, géré chacun par un agent GSA (*Group Security Agent*), responsable de la gestion de la clé de chiffrement locale de son sous-groupe. L'agent du plus haut niveau est noté GSC (*Group Security Controller*) et est responsable de la gestion de la sécurité de tout le groupe multicast, ainsi que du contrôle des agents GSA (cf. Figure 2.3). Un événement d'ajout ou de retrait d'un membre dans le groupe ne nécessite que le renouvellement de la clé du sous-groupe concerné. Ainsi, la dynamicité d'un sous-groupe n'affecte pas les autres sous-groupes.

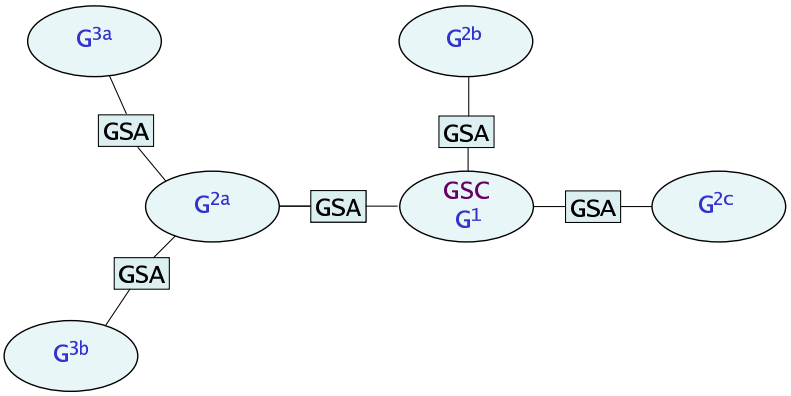


FIG. 2.3 – Clusterisation dans IOLUS

– **Protocoles à p TEKs (p variable)** : pour atténuer le problème rencontré par les protocoles à clusters fixes, d'autres protocoles comme AKMP [BBC02] et SAKM [CBB04b] divisent dynamiquement le groupe en p sous-groupes, p variant selon la dynamicité du groupe multicast. Ils réduisent ainsi le surcoût dû aux opérations de déchiffrement et de rechiffrement de données, tout en atténuant le phénomène "1 affecte n".

– **Protocoles à TEK commune** : ces protocoles utilisent une seule TEK pour tous les clusters du groupe multicast. Ils divisent hiérarchiquement le groupe multicast en des sous-groupes, géré chacun par un contrôleur. DEP [DMS99] appartient à cette famille de protocoles.
DEP consiste à subdiviser hiérarchiquement et statiquement le groupe multicast en des sous-groupes, géré chacun par un SGM (*Sub Group Manager*). DEP ne fait pas confiance aux nœuds non membres du groupe (*Trusting Third Party Problem*) et par conséquent les SGMs qui ne sont pas membres ne doivent pas être capables de déchiffrer le flux multicast émis par la source. Pour cela, trois types de clés de chiffrement de clés sont utilisés :
  – KEK_{i1} : partagée par un SGM_i et ses membres locaux.
  – KEK_{i2} : partagée par le contrôleur global (GC) et les membres locaux du sous-groupe

i, à l'exception du SGM_i (s'il n'est pas membre du groupe).
– KEK_{i3} : partagée par le GC et le SGM_i.

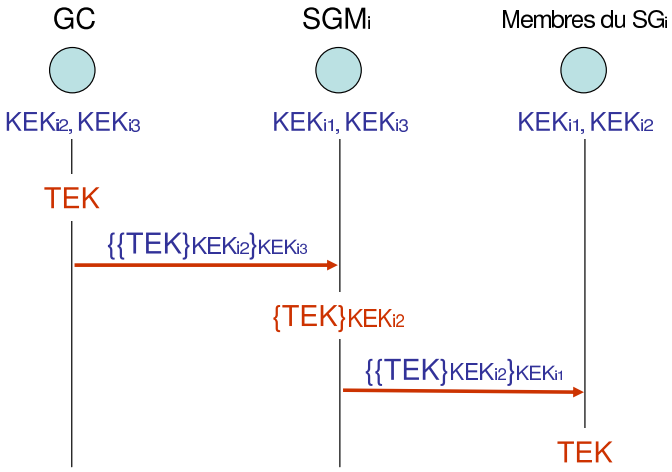


FIG. 2.4 – Distribution de la TEK dans DEP

Pour acheminer la TEK aux membres, le GC transmet un paquet contenant la TEK chiffrée avec KEK_{i2}, puis avec KEK_{i3}. En recevant ce paquet, les SGM_i le déchiffrent en utilisant KEK_{i3}, le rechiffre de nouveau avec KEK_{i1} et le distribuent à tous leurs membres locaux respectifs. Chaque membre du groupe recevant ce paquet, le déchiffre avec KEK_{i1} puis KEK_{i2} et obtient la clé de chiffrement du trafic TEK (cf. Figure 2.4). À chaque événement de *Join* ou de *Leave* au sein d'un sous-groupe, le SGM_i renouvelle la clé KEK_{i1} et ainsi tous les membres ne seront capables d'accéder aux futures TEK que s'ils obtiennent la nouvelle KEK_{i1}. Cependant, les secrets futur et passé ne seront plus assurés. En effet, tout nœud ayant quitté le groupe pourrait continuer à déchiffrer le flux multicast jusqu'au prochain renouvellement périodique de la TEK. Pareillement, un nœud peut accéder au flux multicast antérieur chiffré avec la clé TEK qu'il reçoit quand il rejoint le groupe.
Le protocole DEP propose ainsi une solution au problème de chiffrement / déchiffrement des données, en contre partie, il a recours à un double chiffrement de la clé TEK et donc utilise plus de KEKs. La clusterisation du groupe multicast se fait également de manière statique, ce qui ne peut pas être adapté à la dynamicité des groupes multicast.
Le protocole BAAL [CCS00] appartient également à l'approche décentralisée à TEK commune. Il consiste à diviser le groupe multicast en des sous-groupes, gérés chacun par un contrôleur local délégué par le contrôleur global du groupe. BAAL assure la confidentialité des données, les secrets futur et passé après tout événement d'ajout ou de retrait d'entités dans le groupe ainsi que l'authentification et le contrôle d'accès des adhérents au groupe.

2.2.3 Approche distribuée

Dans cette approche, aussi appelée approche d'accord de clés (*Key Agreement Protocols*), tous les membres du groupe coopèrent et génèrent une TEK, pour établir des communications

sécurisées entre eux. L'approche d'accord de clés élimine les goulots d'étranglement dans le réseau, comparée à une approche centralisée, mais ne permet pas le passage à l'échelle car la TEK est composée des contributions de tous les membres du groupe et nécessite plus de traitements et de calculs.

Le protocole GDH (*Group Diffie Hellman*) est un protocole distribué, de gestion de clé de groupe. Son principe est de généraliser le protocole d'accord de clé Diffie-Hellman au contexte des communications de groupe entre n participants $M_1, M_2, ..., M_n$. n étapes sont nécessaires pour la génération de la clé du groupe. Les premières $n-1$ étapes correspondent à la collection des contributions des membres du groupe de la part du dernier nœud M_n. Dans la dernière étape, M_n diffuse les valeurs intermédiaires aux membres du groupe, leur permettant de calculer la clé du groupe. Dans la figure 2.5, 4 membres M_1, M_2, M_3 et M_4 sont schématisés.

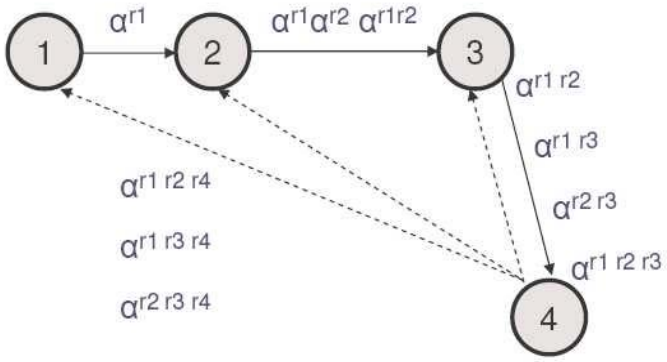


FIG. 2.5 – Exemple de génération de la clé de groupe dans GDH (avec 4 participants)

La génération de la clé de groupe s'effectue de la façon suivante :
Étape 1 - $M_1 \rightarrow M_2$: α^{r1} mod p
Étape 2 - $M_2 \rightarrow M_3$: α^{r1}, α^{r2}, $\alpha^{r1\ r2}$ mod p
Étape 3 - $M_3 \rightarrow M_4$: $\alpha^{r1\ r2}$, $\alpha^{r1\ r3}$, $\alpha^{r2\ r3}$, $\alpha^{r1\ r2\ r3}$ mod p
Étape 4 - $M_4 \rightarrow$Tous : $\alpha^{r1\ r2\ r4}$, $\alpha^{r1\ r3\ r4}$, $\alpha^{r2\ r3\ r4}$ mod p
RAFAELI ET AL. montrent dans [RH03] que le protocole GDH est vulnérable dès que le nombre de membres participant au protocole dépasse quatre.

2.3 Taxonomie des protocoles de gestion de clés dans les réseaux MANETs

La taxonomie des protocoles de gestion de clés dans les MANETs (présentée dans la figure 2.6) que nous proposons, étend et raffine la taxonomie classique utilisée dans les réseaux filaires, tout en intégrant les spécificités et caractéristiques des réseaux ad hoc (le support de mobilité, l'optimisation de l'énergie et la conscience des communications multi-sauts) [BCF06a].

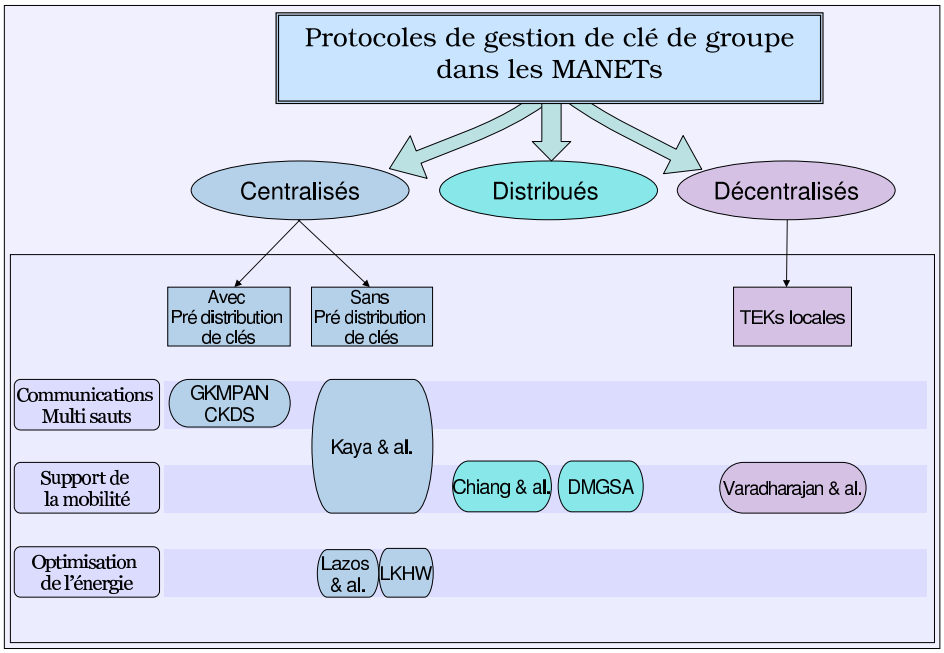


FIG. 2.6 – Taxonomie des protocoles de gestion de clés dans les réseaux ad hoc

2.3.1 Approche centralisée

Dans cette approche, la gestion de clé de groupe est centralisée autour d'une entité unique dans le réseau. Nous divisons cette approche en deux catégories : avec et sans phase de pré-distribution de clés.

Protocoles avec phase de pré-distribution de clés

Ces protocoles configurent les hôtes hors ligne en pré-déployant un ensemble de clés dans chaque nœud. Ces clés lui permettent de déchiffrer le flux multicast envoyé par la source, ou d'obtenir la clé de chiffrement de données envoyée par la source, quand le processus de renouvellement de clé est déclenché. La pré-distribution de clés est utilisée dans les protocoles GKMPAN [ZSXJ04a] et CKDS [MME04] du fait de l'absence d'infrastructure fixe de ces réseaux, qui rend difficile la disponibilité d'une entité centrale assurant l'initialisation de la gestion de clés en ligne.

Le protocole GKMPAN

Le protocole GKMPAN [ZSXJ04a, ZSXJ04b] est basé sur une phase de pré-distribution de listes de clés aux membres du groupe multicast et sur de multiples phases de renouvellement de clés, à la charge d'un serveur de clés.

Pré-distribution des clés : lors de cette phase, chaque membre u du groupe obtient avant le déploiement du réseau ad hoc les clés suivantes :

- un ensemble R_u composé de m clés parmi l ; l étant le nombre total de clés à pré distribuer à tous les membres du groupe $k_1, k_2, ..., k_l$. I_u est l'ensemble des identificateurs des clés,

correspondant à l'ensemble R_u. Les clés de R_u sont utilisées comme clés de chiffrement de clés (KEK : *Key Encryption Key*). L'algorithme de pré distribution de clés permet à chaque nœud i qui connaît l'identité d'un autre nœud j, de déterminer l'ensemble des identifiants des clés I_j et ainsi de déterminer quelle(s) clé(s) utiliser pour pouvoir communiquer avec lui de façon sécurisée.
- la clé de groupe initiale k_g utilisée pour sécuriser les communications entre les membres du groupe.
- une clé secrète partagée entre le serveur de clés et chaque membre individuellement.
- l'authentification des données de la source est assurée via le protocole TESLA [PCTS02, HD03] (cf. ci-dessous). Pour cela, la première clé de chaînage TESLA (clé de validation ou dite de «*commitment*») est chargée dans chaque nœud .

Des membres peuvent joindre le groupe dans GKMPAN, même après la phase de pré-déploiement de clés. Le serveur de clés peut par exemple ajouter des entités dans le groupe pour compenser les membres exclus. Avant d'ajouter un membre u dans le groupe, le serveur de clés y déploie son ensemble de clés R_u ainsi que la clé du groupe courante. Suite à cet événement et dépendant de l'application à sécuriser, le serveur de clés peut décider ou non de renouveler la clé du groupe k_g pour assurer le secret passé et de diffuser ainsi un message de renouvellement de la clé du groupe $k'_g = f_{k_g}(0)$, f est une fonction pseudo-aléatoire.

Distribution de la clé du groupe : le processus de distribution de la clé du groupe est initié par le serveur de clés qui génère une nouvelle clé de groupe. Il la distribue ensuite, saut par saut, en utilisant les KEKs pré-déployées pour assurer le chiffrement de la clé du groupe. Ainsi, le serveur de clés ne délivre la clé qu'à ses voisins immédiats (à un seul saut), qui acheminent la clé à leurs voisins d'une façon récursive et sécurisée. De cette manière, GKMPAN exploite la propriété de communication multi-sauts dans les réseaux ad hoc, où les membres du groupe jouent le rôle d'hôtes et de routeurs.

Exclusion d'un membre du groupe : lors de la révocation d'un membre malicieux, le serveur de clés diffuse une notification de révocation dans le réseau ad hoc, contenant l'identifiant du membre exclu, l'identifiant de la clé non compromise i qui est connue du plus grand nombre de membres du groupe et la nouvelle clé du groupe chiffrée avec la clé choisie d'identifiant i. Les membres ne détenant pas la KEK i utilisée pour le chiffrement de la clé de groupe, recevront cette clé acheminée par leurs voisins, chiffrée avec d'autres KEKs non compromises. Le message de notification est authentifié via le protocole TESLA qui permet en plus d'assurer la tolérance aux pertes de messages. Pour cela, le serveur de clés et les nœuds du groupe sont synchronisés ; chaque membre connaît une limite supérieure du temps de synchronisation avec le serveur, notée δ_t.

Le temps est divisé en des intervalles de durée T_{int} chacun. A chaque intervalle I_j, lui correspond une clé d'authentification k'_j. La source génère une chaîne de clés $k_0, ..., k_t$ en utilisant une fonction à sens unique f. Pour cela, la dernière clé k_t est générée aléatoirement et les autres clés sont générées via l'équation : $k_{j-1} = f(k_j)$. Ensuite, la source génère les clés MAC (*Message Authentication Code*) d'authentification $k'_j = g(k_j)$, g étant une autre fonction à sens unique. La figure 2.7 illustre le chaînage de clés de TESLA.

La source authentifie chaque paquet P_i avec la clé de l'intervalle du temps courant j et inclut les informations d'authentification avec les données MAC(K'_j, P_i). La source inclut aussi la clé k_{j-d} utilisée pour authentifier les paquets envoyées avant d intervalles de temps ; d étant le délai de révélation de TESLA.

Les récepteurs membres du groupe vérifient l'authenticité des messages envoyés par la

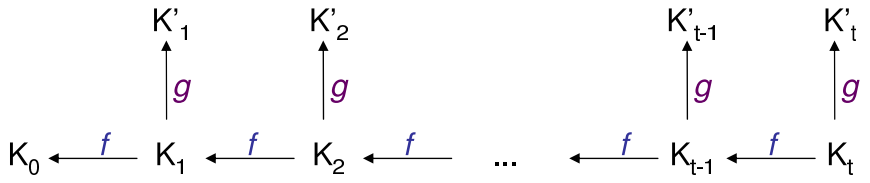


FIG. 2.7 – Illustration de la chaîne de clés MAC dans TESLA

source en vérifiant si la clé révélée (après d intervalles) coïncide avec le résultat de la fonction à sens unique f : $k_0 = f^j(k_j)$.

Mise à jour des clés compromises : toutes les clés KEKs connues du membre exclu u sont également compromises et doivent être renouvelées par les autres membres détenant ces clés, de la façon suivante :

- M est l'identifiant de la clé non compromise qui est connue du plus grand nombre de membres du groupe,
- le serveur de clés génère une clé intermédiaire $k_{im} = f_{k_M}(k_g)$.
- les clés k_i détenues par le membre exclu u (R_u) sont renouvelées par des clés k'_i comme suit : $k'_i = f_{k_{im}}(f_{k_i}(0))$.

Le protocole CKDS

CKDS [MME04] (*Combinatorial Key Distribution Scheme*) est un protocole de sécurité des communications de groupe multicast, au niveau applicatif. La distribution de clés dans CKDS est basée sur le système combinatoire EBS (*Exclusion Basis System*) [SR04], combiné avec CAN (*Content Addressable Network*) [RFH+01].

Un système EBS de dimension (n, k, m) tels que $1 < k$ et $m < n$ est une collection Γ de sous-ensembles de $[1, n] = \{1, , ..., n\}$ tel que pour tout élément $t \in [1, n]$, deux propriétés sont vérifiées :

- t appartient à au plus k sous-ensembles de Γ,
- il existe exactement m sous-ensembles dans Γ $(A_1, A_2, ..., A_m)$ qui vérifient que l'union $\cup_{i=1}^{m} A_i$ est égale à $[1, n] - \{t\}$ (chaque élément t est exclu par l'union d'exactement m sous-ensembles de Γ).

Par exemple, un système EBS de dimensions $(8, 3, 2)$ est une collection des sous-ensembles $\Gamma = \{A_1 = \{5, 6, 7, 8\}, A_2 = \{2, 3, 4, 8\}, A_3 = \{1, 3, 4, 6, 7\}, A_4 = \{1, 2, 4, 5, 7\}, A_5 = \{1, 2, 3, 5, 6, 8\}\}$. Ces sous-ensembles vérifient les propriétés suivantes :

$[1, 8] - \{1\} = A_1 \cup A_2$
$[1, 8] - \{2\} = A_1 \cup A_3$
$[1, 8] - \{3\} = A_1 \cup A_4$
$[1, 8] - \{4\} = A_1 \cup A_5$
$[1, 8] - \{5\} = A_2 \cup A_3$
$[1, 8] - \{6\} = A_2 \cup A_4$
$[1, 8] - \{7\} = A_2 \cup A_5$
$[1, 8] - \{8\} = A_3 \cup A_4$

Un système d'exclusion EBS est utilisé pour représenter un groupe multicast ; n étant le

nombre de membres du groupe et A_i une clé i détenue par le serveur de clés. Toute clé A_i est connue par tous les membres dont l'identifiant apparaît dans le sous-ensemble A_i et vérifiant les deux propriétés d'un système EBS présentées ci-dessus.

Le protocole CKDS utilise le système EBS lors de la phase de pré-déploiement des clés aux membres du groupe. Ainsi, chaque nœud dans CKDS détient k clés (clés connues) et ne connaît pas m clés (clés inconnues). La figure 2.8 illustre un exemple d'une matrice EBS, avec k=3, m=2 et 10 membres U1 à U10. Une case (i,j) est égale à 1 si le membre Uj connaît la clé Ki. Cet exemple est présenté dans [MME04].

Nodes / Keys	U1	U2	U3	U4	U5	U6	U7	U8	U9	U10
K1	1	1	1	1	1	1	0	0	0	0
K2	1	1	1	0	0	0	1	1	1	0
K3	1	0	0	1	1	0	1	1	0	1
K4	0	1	0	1	0	1	1	0	1	1
K5	0	0	1	0	1	1	0	1	1	1

FIG. 2.8 – Matrice EBS dans CKDS [MME04]

CAN est une table de hachage distribuée, utilisée pour réaliser une répartition de tous les nœuds du réseau en un espace m-dimensionnel. Ainsi, chaque nœud a un quadrant dans l'espace, localisé selon ses clés inconnues dans le système EBS.

Pour distribuer et renouveler les clés, une entité appelée contrôleur global est supposée disponible dans le réseau. Elle est responsable de la génération des clés du groupe et de la construction des messages de renouvellement de ces clés. La tâche de distribution de ces messages est déléguée aux membres du groupe eux mêmes, selon deux schémas de distribution de clé possibles.

1. Le schéma de distribution multicast m-dimensionnel. Lors de l'exclusion d'un membre du groupe, les clés détenues par ce membre sont compromises et doivent être renouvelées. L'algorithme de renouvellement de clés sera déclenché par un nœud diagonal dans l'espace partitionné (le nœud qui détient toutes les clés inconnues du membre compromis). Ce nœud est noté IGD (Initial Global Distributor). L'IGD reçoit de la part du contrôleur global les messages de renouvellement de clés à acheminer aux autres membres du groupe, non compromis. Dans la figure 2.8, si le nœud U1 est compromis, U6, U9 et U10 peuvent accomplir le processus de renouvellement de clés, parce qu'ils connaissent les clés inconnues d'U1 (K4 et K5). L'IGD sélectionné commence par localiser les membres centraux dans chaque quadrant de l'espace m-dimensionnel, appelés LQD (Local Quadrant Distributor). Ensuite, il leur envoie le message de renouvellement des clés approprié, via une technique d'inondation directe. Le distributeur local LQD achemine ensuite, en multicast, le message de renouvellement de clés aux membres de son quadrant. Comme GKMPAN, CKDS exploite de cette manière la propriété de communications multi-sauts des réseaux ad hoc.

2. Le schéma de distribution 2D-multicast. Ce schéma, basé également sur les distributeurs de clés initial et locaux (IGD et LQDs), tend à réduire le surcoût de communication et de chiffrement du schéma m-dimensionnel de distribution des clés. En effet, dans le schéma de distribution multicast m-dimensionnel, des messages de renouvellement de clés peuvent atteindre des membres qui n'ont besoin que d'un ensemble des clés renouvelées et non pas de la totalité. De plus, lors de l'acheminement des messages de renouvellement de clés, le distributeur initial IGD et les distributeurs locaux LQD doivent réaliser des opérations de rechiffrement. Les membres finaux doivent ainsi réaliser deux opérations de déchiffrement, coûteuses dans un environnement ad hoc.

   Le schéma de distribution 2D-multicast propose donc de diriger les messages de renouvellement de clés aux seuls membres intéressés par ce renouvellement. Ainsi, un message de renouvellement de clés ne contient qu'une seule clé mise à jour. De plus, pour éviter les doubles opérations de déchiffrement, les nouvelles clés sont chiffrées avec une nouvelle clé de chiffrement, construite à partir de la clé compromise et d'une autre clé Ki non détenue par le membre malicieux, via une fonction de hachage. Un message de mise à jour d'une clé Kj à K'j est noté Rij et a la forme suivante : Rij = Ki|Kj (K'j), avec Ki|Kj est la clé de chiffrement générée à partir de Ki et Kj.

Protocoles sans phase de pré-distribution de clés

Cette catégorie de protocoles n'a pas de phase de pré-distribution de clés. Trois protocoles appartiennent à cette approche : Kaya et al. [KLNY03], Lazos et al. [LP03] et LKHW [PML+03]. Nous les présentons dans ce qui suit.

Le protocole de Kaya et al.

Kaya et al. [KLNY03] proposent un protocole de gestion de clé de groupe dans les MANETs, tenant compte de la mobilité des nœuds, et des communications multi-sauts de cet environnement. En effet, les membres joignent le groupe multicast via le voisin le plus proche, appartenant déjà au groupe, en utilisant l'information GPS (*Global Positionning System*). Les requêtes de Join sont diffusées en anycast (seul le voisin le plus proche répond à la requête), avec une portée limitée (champ TTL) pour atteindre le premier membre du groupe. Ainsi, en plus du gain en terme de surcoût de communications, cette méthode permet la construction d'un arbre multicast avec les chemins les plus courts, facilitant et optimisant la distribution de clés.
Un service de certification est fourni par ce protocole, pour assurer le contrôle d'accès des membres et la révocation des nœuds malicieux. Seuls les nœuds ayant un certificat valide sont capables d'accéder au flux multicast. Un nœud voulant joindre le groupe, doit obtenir le certificat de sécurité auparavant (hors ligne). Ce certificat doit être signé par une autorité de certification tierce (TTP : *Trusted Third Party*). Si l'authentification d'un nœud voulant joindre le groupe auprès d'un autre membre du groupe réussit, les deux entités génèrent et partagent une clé secrète. Ensuite, le contrôle d'accès du nouveau membre est vérifié via son certificat. En cas de réussite, ce membre pourra accéder au flux multicast envoyé par la source, saut par saut, chiffré avec la clé secrète obtenue lors de l'authentification. Les nœuds exclus du groupe et donc ayant des certificats bannis, ne doivent plus être capables d'accéder aux données multicast. Pour cela, la source du groupe envoie périodiquement et en multicast, un message contenant la liste de tous les certificats révoqués. Les membres du groupe enregistrent cette liste et pourront ainsi authentifier et contrôler l'accès de nouveaux membres voulant joindre

le groupe.

Le protocole de Lazos et al.

La proposition de LAZOS ET AL. [LP03] suit une architecture centralisée de gestion de clés, prenant en compte la contrainte de l'énergie limitée dans les réseaux ad hoc. Elle améliore le schéma de distribution de clé LKH [WGL98] (*Logical Key Hierarchy*) et l'adapte au contexte des réseaux ad hoc statiques, en optimisant la consommation de l'énergie, avec l'utilisation de l'information de localisation géographique des membres (GPS). L'idée de base de ce protocole est que les membres géographiquement proches peuvent potentiellement être atteints par un message de diffusion, ou utiliser le même chemin pour accéder au flux multicast. Le réseau ad hoc est représenté par un espace à deux dimensions et l'algorithme de clusterisation K-means [Mac67] est utilisé pour former des groupes de forte corrélation et déduire ensuite l'arbre de distribution de la clé de groupe.
Le processus de distribution de clés, basé sur l'algorithme K-means, est composé de plusieurs étapes. Tout d'abord, tous les membres du groupe sont affectés à un seul cluster. Ensuite, chaque cluster est divisé en deux sous clusters via l'algorithme K-means. Une procédure de raffinement est utilisée pour équilibrer le nombre de membres par cluster. Ces étapes sont réitérées, jusqu'à aboutir à des clusters formés de un ou deux membres. Les clusters formés d'un seul membre sont fusionnés, si cela est possible. L'étape finale consiste à faire correspondre la hiérarchie de clusters en une hiérarchie logique de distribution de clés LKH. La figure 2.9 illustre une exécution de cet algorithme. Dans cet exemple, les membres M4 et M6 sont proches géographiquement. Ils sont donc frères dans l'arbre de distribution de clés LKH.

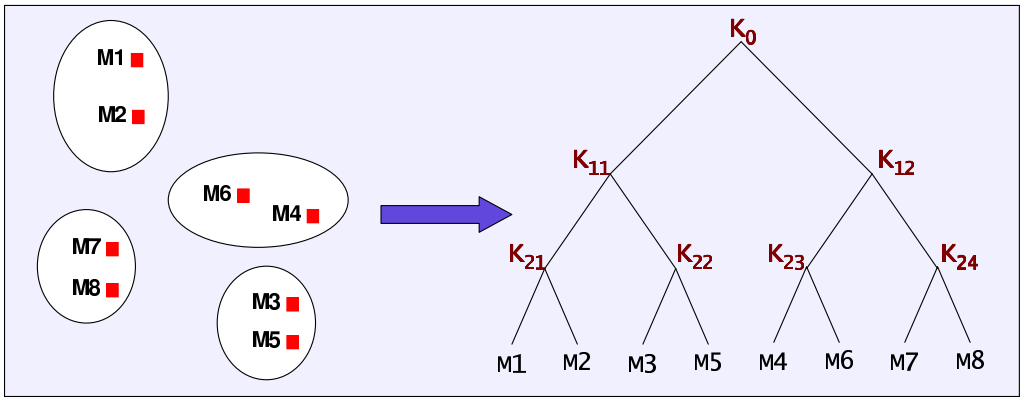


FIG. 2.9 – Processus de distribution de clés basé sur l'algorithme K-means

Le protocole LKHW

LKHW [PML+03] est un schéma de sécurisation des communications multicast, destiné à opérer dans les réseaux de capteurs (WSN : *Wireless Sensors Networks*). Les acteurs de LKHW sont la source du groupe et les capteurs. La source est responsable de la collection des données issues des capteurs du réseau. Ces nœuds sont des entités du réseau qui peuvent

fournir des données demandées par la source. Les capteurs ont des capacités physiques faibles, en termes de communication et de calcul. La distribution de clés est basée sur LKH 2.2 et les renouvellements de clés utilisent la diffusion directe, optimisant la consommation de l'énergie. Les services de sécurité assurés par LKHW sont la confidentialité, l'intégrité et l'authentification des données. Les secrets futur et passé sont également assurés dans LKHW.

Les principales phases de LKHW sont l'initialisation du groupe et les processus de renouvellement de clés. Lors de l'initialisation du groupe, l'établissement du groupe sécurisé commence quand la source construit la hiérarchie logique de clés LKH. Initialement, la source diffuse un message d'exploration à tous les membres du réseau, pour chercher les nœuds capables de lui fournir les données qu'elle requiert. Les membres intéressés répondent à ce message, en déclarant les tâches qu'ils peuvent accomplir. La source collecte ensuite ces réponses et envoie pour chaque membre son identifiant et l'ensemble de clés correspondant à sa localisation logique dans l'arbre LKH. À cette étape, les communications sécurisées du groupe peuvent commencer.

Le processus de renouvellement de clés est déclenché après tout événement de Join ou de Leave dans le groupe. Quand un nœud veut joindre le groupe, la source lui envoie l'ensemble de clés correspondant à sa position dans l'arbre LKH. De plus, tous les membres du groupe doivent mettre à jour l'ensemble de leurs clés (notés K_i) pour garantir le secret passé. Pour cela, la source envoie un message à tous ses membres contenant un entier appelé *seed*, authentifié avec l'ancienne clé du groupe. Les membres du groupe, après avoir authentifié ce message, extrait le *seed* et génèrent la nouvelle clé du groupe de la façon suivante :

$$K_i' = \{k' \textit{ tel que } k' = MAC_k(seed),\ \forall k \in K_i\}$$

De même, quand un nœud quitte le groupe, les clés de l'arbre LKH de sa position à la racine sont mises à jour pour garantir le secret futur. Pour cela, la source du groupe commence par identifier les clés compromises (qui étaient connues du membre quittant le groupe), ensuite les régénère via une fonction à sens unique H. Par exemple, si les clés K_0, K_2, K_6 et K_{12} sont à renouveler, la source procède comme suit :

$$K_0' = H(K_2') = H(K_6') = H(K_{12}')$$

K_0' étant la nouvelle clé du groupe (racine de l'arbre LKH). Ces nouvelles clés seront ensuite distribuées aux membres concernées par ce renouvellement, de façon sécurisée via les clés qui sont à leur possession (non détenues par le membre quittant).

La technique de diffusion directe dans LKHW est optimisée, via l'utilisation des caches, la suppression des messages dupliqués et la prévention contre les cycles.

2.3.2 Approche distribuée

La gestion de clé de groupe dans une approche distribuée est à la charge de tous les membres du groupe multicast, qui collaborent et coopèrent pour partager une clé de groupe, afin d'assurer des communications sécurisées entre eux. Les protocoles présentés dans Chiang et al. [CH03] et DMGSA [KLG06] illustrent ce type d'approche.

Le protocole de Chiang et al.

Chiang et al. proposent un protocole distribué de gestion de clé de groupe, pour les MANETs [CH03], basé sur les données GPS (latitude, longitude et altitude) et sur le protocole

d'échange de clés GDH (*Group Diffie Hellman*) [ITW82].
À l'initialisation du protocole, chaque nœud ad hoc A génère sa clé publique K_{pubA} de la façon suivante : $K_{pubA} = \alpha^a \ mod \ p$, avec α un entier, p un grand nombre premier (α et p sont connus par tous les participants au groupe) et a un entier aléatoire privé. Ensuite, chaque nœud diffuse sa localisation GPS et sa clé publique à toutes les autres entités du réseau. Grâce à ces informations, chaque membre du groupe connaît la topologie de tout le réseau.

Quand une source veut émettre des données en multicast à tous les membres du groupe, elle construit l'arbre multicast minimal, utilisant l'information de localisation des membres via le GPS. L'arbre multicast est ensuite codé en un nombre de Prüfer [Prü18] et envoyé à tous les membres du groupe, qui le décodent et savent ainsi s'il doivent ou non acheminer les données multicast envoyées par la source du groupe.

L'algorithme de Prüfer calcule un nombre de Prüfer approprié pour coder un arbre multicast, se basant sur les degrés[10] des membres du groupe. En effet, un nœud de degré d apparaît exactement $d-1$ fois dans le nombre de Prüfer.

L'algorithme de codage de Prüfer, faisant correspondre à un arbre multicast T un nombre de Prüfer P, est basé sur les trois étapes suivantes :

- Étape 1 : Étant donné i la feuille de l'arbre de plus petit identificateur. j étant le nœud parent de i. La première étape consiste à ajouter j au nombre de Prüfer (de gauche à droite).
- Étape 2 : Ensuite le nœud i ainsi que l'arc (i, j) sont enlevés de l'arbre T.
- Étape 3 : Les étapes 1 et 2 sont réitérées jusqu'à ce qu'un seul arc reste dans l'arbre T.

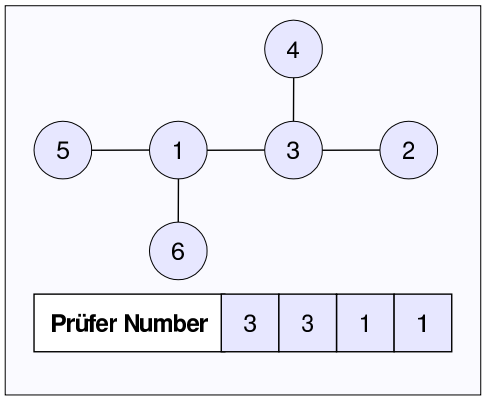


FIG. 2.10 – Un arbre multicast et son nombre de Prüfer [CH03]

L'algorithme de décodage fait correspondre à un nombre de Prüfer P un unique arbre multicast T et se base également sur trois étapes présentées ci-dessous (P' est l'ensemble de membres de T qui n'apparaissent pas dans P) :

- Étape 1 : i est le plus petit entier de P' et j est le premier entier à gauche de P. La première étape consiste à ajouter l'arc (i, j) dans l'arbre T et enlever i de P' et j de P. Si j n'apparaît pas une autre fois dans P, il faut l'ajouter dans P'.
- Étape 2 : L'étape 1 est ré itérée jusqu'à ce que P soit vide.

[10]Le degré d'un membre dans un groupe multicast est égal au nombre de ses liens dans l'arbre multicast

- Étape 3 : À ce niveau, deux éléments r et s restent dans P'. La dernière étape consiste à ajouter l'arc (r, s) dans T.

La figure 2.10 illustre un exemple d'arbre multicast et son nombre de Prüfer. Le degré des nœuds 3 et 1 est égal à trois, ils apparaissent donc deux fois dans le nombre de Prüfer.

La clé du groupe est générée par tous les membres du groupe, via le protocole d'échange de clés GDH. La clé du groupe est construite à partir d'une combinaison de leurs clés publiques. La source envoie ensuite la séquence Prüfer à tous les membres du groupe, en multicast, chiffrée avec la clé du groupe. En recevant la séquence de Prüfer, un membre pourra décoder l'arbre multicast construit par la source et saura s'il doit ou non acheminer les paquets du flux multicast à d'autres membres du groupe.
Un groupe sécurisé est ainsi représenté par un graphe de clés, composé de deux types de nœuds, les feuilles représentant les membres du groupe (U) et les nœuds intermédiaires (K) représentant les clés publiques des membres. La racine de l'arbre k_p indique la clé du groupe (appelée aussi clé de Prüfer P). Le groupe sécurisé est ainsi noté (U, K, P). La figure 2.11 illustre un exemple de graphe de clés extrait de [CH03].

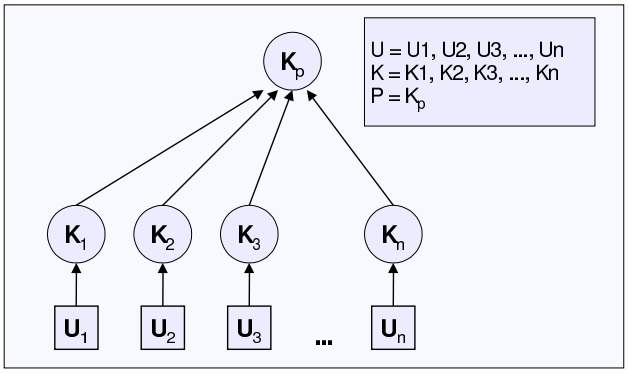


FIG. 2.11 – Un graphe de clés dans CHIANG ET AL. [CH03]

Le graphe de distribution de clés peut être étendu pour assurer des communications sécurisées entre plusieurs groupes multicast. La clé de fusion des groupes est construite, de façon hiérarchique, à partir des clés de groupe initiales.

Le protocole DMGSA

L'architecture de sécurité DMGSA [KLG06] (*Distributed Multicast Group Security Architecture*) est une architecture distribuée et clustérisée. Elle prend en compte la mobilité et la densité des nœuds lors de la création des clusters.

La gestion de la clé du groupe est réalisée à travers des entités spécifiques dans le réseau, appelées GCKSs (*Group Control Key Server*). Ces entités jouent le rôle de clusterheads et forment ensemble le cœur du groupe multicast. Il n'existe donc pas de point de vulnérabilité central.

Dans chaque voisinage de k sauts, un GCKS est élu à chaque changement ou modification de la topologie. L'élection des GCKS est réalisée de façon distribuée, suivant deux étapes : une phase de formation et une phase de maintenance des clusters.
La phase de formation distribuée des clusters est initiée par un nœud qui n'appartient pas

encore à un cluster. Ce nœud diffuse des messages d'élections, se réclamant lui même comme clusterhead (GCKS). La diffusion de ces messages est réalisée dans le voisinage à k sauts de l'émetteur (le champs TTL du paquet est positionné à k). Le choix de k se base sur l'estimation du nœud de la densité locale dans son voisinage. Cette estimation est calculée via un algorithme de détection de voisinage. En cas de concurrence entre deux entités, le nœud détenant la plus petite valeur de k et le plus petit identifiant est élu comme GCKS.
Durant la phase de maintenance distribuée des clusters, chaque clusterhead envoie périodiquement un message pour se réclamer GCKS dans la portée de k sauts, maintenant ainsi les membres qui reçoivent ce message dans son cluster. Un membre ne recevant pas un message de la part de son clusterhead pendant une durée de temps déterminée, se joindra à un autre cluster.

La gestion de clés dans DMGSA consiste à partager une clé de groupe TEK, gérée par le groupe des GCKSs. Un membre du groupe multicast reçoit donc la TEK de la part du GCKS le plus proche de sa localisation (k sauts au maximum). Afin de distribuer de façon sécurisée la TEK à ses membres, un GCKS authentifie ses membres locaux lors de leurs adhésion au groupe et contrôlent leur accès au groupe, à travers des certificats déployés (hors ligne). En cas de succès il établit avec chaque membre de son cluster, une clé secrète appelée KEK, qu'il utilisera pour chiffrer la TEK du groupe.
Le renouvellement de la TEK est déclenché quand la fréquence d'adhésion à un cluster (événements de Join et de Leave), évaluée par le GCKS, dépasse un certain seuil. Dans ce cas, le GCKS génère une nouvelle TEK, l'envoie à ses membres locaux individuellement et de façon sécurisée via leurs KEKs et l'achemine aux autres GCKS du groupe multicast. La sécurisation des messages transmis entre les différents clusterheads n'est pas détaillée dans [KLG06].

2.3.3 Approche décentralisée

L'approche décentralisée divise le groupe multicast en des sous-groupes ou clusters. Chaque cluster est géré séparément par un contrôleur local, responsable de la sécurité des membres de son sous-groupe et de la gestion d'une clé de chiffrement TEK locale au sein de son cluster (protocoles à "**TEKs locales**").

Ainsi, les contrôleurs locaux génèrent et distribuent la TEK locale à leurs membres locaux. En recevant le flux multicast envoyé par la source, les contrôleurs locaux le déchiffrent avec la clé appropriée, le rechiffrent avec les clés locales de leur cluster et l'acheminent à leurs membres locaux. L'avantage de cette approche est qu'elle permet d'assurer les secrets futur et passé tout en atténuant le phénomène "1 affecte n". Le changement de la clé locale d'un cluster, dû à un événement de Join ou de Leave, n'affecte pas les autres clusters. Cependant, les doubles opérations de déchiffrement et de rechiffrement du côté des contrôleurs locaux constituent un inconvénient majeur.

Le protocole de Varadharajan et al.

Le protocole de gestion de clés proposé par VARADHARAJAN ET AL. [VHS01] a opté pour un protocole à TEKs locales. Il opère dans les réseaux NTDR (Near Term Digital Radio). L'architecture d'un réseau NTDR est composée d'un ensemble de clusters, contenant chacun un clusterhead et l'ensemble des clusterheads forme le cœur de routage du réseau. Les communications entre clusters sont restreintes aux clusterheads (cf. Figure 2.12), qui partagent une clé symétrique de chiffrement notée CHG_K (Clusterheads Group Key). Un cluster est constitué de nœuds à un seul niveau du clusterhead. Tous les membres d'un réseau NTDR détiennent

des certificats qu'ils ont reçus préalablement de la part d'une autorité de certification.

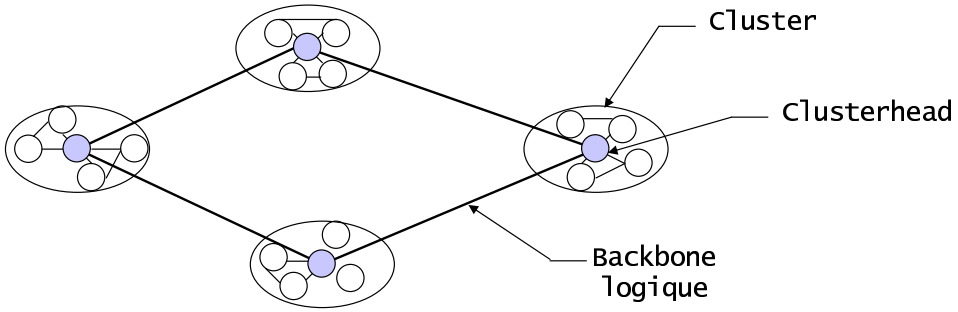


FIG. 2.12 – Architecture d'un réseau NTDR

La mobilité des nœuds est prise en compte dans ce protocole, lors de la formation des clusters et l'élection des clusterheads. En effet, chaque nœud se comporte comme un clusterhead s'il ne détecte aucun clusterhead dans son voisinage. Des mécanismes sont utilisés pour limiter le nombre de membres qui essaient simultanément de devenir des clusterheads. Une fois qu'un nœud assure le rôle d'un clusterhead, il envoie immédiatement à tous les autres membres du cluster son nouvel état.

Les fonctionnalités d'un clusterhead au sein de son cluster sont principalement la maintenance de la liste de ses voisins, l'acceptation ou le refus de l'adhésion d'un nœud à son cluster (à travers son certificat) et l'acheminement des informations inter et intra clusters. Une procédure de notification est prévue dans [VHS01], précédant un déplacement ou un départ d'un clusterhead et anticipant une phase de ré-élection d'un autre clusterhead au sein du cluster.

La confidentialité des communications multicast est réalisée via deux types de clés :

- La clé locale du cluster (GCK), utilisée pour le chiffrement des données intra-cluster.
- La clé de chiffrement de clé KEK partagée entre un clusterhead et un membre de son cluster. Cette clé est une combinaison d'un secret partagé s et de l'adresse IP du membre, comme suit : KEK $= f(s, @IP)$.

Le chef du cluster chiffre la GCK par la KEK et l'envoie à ses membres locaux respectivement. Ainsi, tous les membres peuvent chiffrer et déchiffrer des données au sein de leurs clusters.

2.4 Discussions

Dans cette section, nous évaluons et comparons les protocoles de gestion de clé de groupe dans les réseaux ad hoc et nous analysons leurs performances et leurs propriétés de sécurité.

Les métriques de comparaison que nous utilisons sont les contraintes et pré-requis des protocoles et leur applicabilité réelle, les propriétés de sécurité à assurer pour sécuriser les communications de groupe dans les réseaux ad hoc (authentification, confidentialité et intégrité des données, révocation des membres malicieux, ...), le passage à l'échelle en terme du

		Contraintes et pré-requis	Services de sécurité	Passage à l'échelle		Vulnérabilités
				Surcoût de calcul	Surcoût de stockage	Surcoût de communications	
Centralisés	Avec pré distribution de clés	GKMPAN [ZSXJ04a]	-Pré distribution de clés -Synchronisation	-Déchiffrement et re chiffrement de la TEK (saut par saut) -Algorithme de reconnaissance des clés communes entre les membres -Authentification TESLA (opération de hachage par paquet)	-m Clés pré distribuées, choisis parmi **l** -Clé d'authentification TESLA -Clé secrète entre le serveur et tout membre -Bufferisation TESLA pendant *d* temps de révélation de TESLA	Acheminement de la clé du groupe saut par saut, dépendant de m et **l**. La probabilité d'avoir la même clé entre deux nœuds est : $P = 1 - \binom{l-m}{m} / \binom{l}{m}$	-Serveur de clés
		CKDS [MME04]	-Pré distribution de clés -Contrôleur global -EBS & CAN	-m-dimensionnel : Déchiffrements et re chiffrements intermédiaires de la TEK	-k Clés pré distribuées -matrice EBS à la source du groupe (N * (k+m)) -Voisinage : 2*(m+k) voisins	m-dimensionnel : -Acheminement de la TEK par les LCDs -Inondation des message de renouvellement de clés par le IGD	-Contrôleur global
	Sans pré distribution de clés	Kaya et al. [KLNY03]	-GPS -Autorité de certification -Synchronisation	-Gestion de la liste de révocation par la source -Déchiffrement et rechiffrement des données entre chaque nœud parent de l'arbre et ses membres fils -Authentification par TESLA	-Liste de révocation -Certificats -Clé partagée KEKs entre un membre de l'arbre et son nœud parent -Bufferisation des paquets pour authentification TESLA	-Messages de maintenance et d'optimisation de l'arbre multicast	Mise à jour de la liste de révocation
		Lazos et al. [LP03]	-GPS -Algorithme K-means	-Algorithme de clusterisation K-means en O(N²)	-Clés de l'arbre LKH Contrôleur O(N) Membres O(log N)	-Initialisation du groupe (N messages de LKH)	-Source -dédié aux réseaux ad hoc statiques
		LKHW [PML+03]	-Algorithme de diffusion directe	-Hachage des messages de renouvellements de la clé du groupe -Génération de la nouvelle clé du groupe	-Clés de l'arbre LKH Contrôleur O(N) Membres O(log N)	-Signalisation d'initialisation du groupe (3*N messages), Inondation des messages	Source
Distribués		Chiang et al. [CH03]	-GPS -Protocole GDH	-Calcul des clés publiques -Codage et décodage du nombre de Prüfer en O(N) -Chiffrement et déchiffrement de la séquence de Prüfer	-Clés publiques -Séquence Prüfer	-Inondation de la localisation GPS et des clés publiques	-Inondation GPS -Coûts élevés de calcul
		DMGSA [KLG06]	Algorithme de clusterisation	Confidentialité des données	-Déchiffrements et re chiffrements intermédiaires de la TEK -Algorithme de détection de voisinage	Clés partagées KEKs entre un GCKS et tout membre de son cluster	-Messages d'initialisation de la clusterisation du groupe en O(N) -Message périodique de maintenance de cluster par tout GCKS	Clusterheads
Décentralisés		Varadharajan et al. [VHS01]	Algorithme de clusterisation	-Confidentialité des données	- Données multicast déchiffrées et re chiffrées	- Clés partagées entre un clusterhead et tout membre de son cluster - Bufferisation des données pour déchiffrement et re chiffrement par les clusterheads	- Acheminement de tous les messages via les clusterheads	Clusterheads

N : nombre de membres du groupe, c : nombre moyen de membres par cluster

TAB. 2.1 – Évaluation des protocoles de gestion de clé de groupe dans les MANETs

surcoût de calcul, du surcoût de stockage et du surcoût de communication, les vulnérabilités et l'efficacité contre les goulots d'étranglement. Le tableau 2.1 résume ces comparaisons.

2.4.1 Contraintes et pré-requis

Les propositions de KAYA ET AL., CHIANG ET AL. et LAZOS ET AL. requièrent un système de localisation GPS, pour prendre en considération la position géographique des membres. L'information GPS y est utilisée pour construire efficacement les chemins entre membres de groupe. Cependant, dans CHIANG ET AL., l'information GPS est diffusée par inondation dans le réseau permettant à tout nœud de connaître la topologie complète du groupe. Cette inondation est très coûteuse dans les MANETs et peut compromettre la mise en œuvre effective du protocole.

Tous les protocoles proposés qui requièrent que tout membre du groupe détienne une clé publique ([KLNY03, VHS01] ...), supposent la disponibilité d'une autorité de certification et de confiance dans un réseau ad hoc, capable de fournir la preuve de l'identité des membres. Cette supposition est très difficile à satisfaire dans un environnement sans infrastructure fixe, où les liens sont éphémères et dynamiques.

La validation de la liste de clés dans KAYA ET AL. et GKMPAN nécessite l'authentification via TESLA, la synchronisation temporelle entre membres du groupe et la bufferisation des messages reçus du côté des récepteurs, difficiles à déployer dans un environnement ad hoc, dans lequel les liens entre nœuds ne sont pas fixes et la capacité de stockage est limitée.

2.4.2 Services de sécurité

La confidentialité des données est assurée par les protocoles de gestion de clé de groupe présentés précédemment. Ce service est basé sur la distribution de la clé de groupe utilisée par la source du groupe pour chiffrer le flux multicast et par les récepteurs pour le déchiffrer. Cependant, les services d'authentification et de contrôle d'accès ne sont fournis que par KAYA ET AL. [KLNY03], grâce une gestion de certification ou une liste de contrôle d'accès.
Dans KAYA ET AL., l'autorité de certification offre hors ligne des certificats de sécurité à tous les membres du groupe, leur permettant de s'authentifier, prouver leurs identités et joindre le groupe multicast en ligne.
La révocation des membres malicieux est assurée via le processus de pré-distribution de clés dans GKMPAN [ZSXJ04b] et CKDS [MME04]. Les clés d'un nœud compromis dans ces deux protocoles sont également compromises et isolées et ne seront plus utilisées lors du renouvellement des clés par les autres membres du groupe. Cependant, la procédure de Join est difficile à déployer dans ces protocoles, puisqu'un nouveau membre doit avoir des clés pré-déployées.

2.4.3 Passage à l'échelle

Le passage à l'échelle est une métrique à prendre en compte lors de la conception d'un protocole de gestion de clés dans les réseaux ad hoc, car un grand nombre de membres peuvent participer à une session multicast dans un tel réseau. Nous étudions dans ce qui suit les protocoles discutés selon trois éléments intervenant dans le passage à l'échelle : le surcoût de calcul, le surcoût de stockage et le surcoût de communications.

Surcoût de calcul

La métrique d'opérations intermédiaires de chiffrement du flux de données est très importante dans les environnements ad hoc, du fait des capacités de calcul généralement limitées des équipements et entités du réseau. La solution de gestion de clés la plus appropriée ne doit donc pas utiliser des opérations intermédiaires de chiffrement et de déchiffrement du flux. Les données multicast sécurisées sont donc déchiffrées seulement par les récepteurs finaux, comme cela est réalisé dans KAYA ET AL., LAZOS ET AL., LKHW, CHIANG ET AL. et la version 2D-multicast de CKDS. L'inconvénient de ces protocoles est qu'ils sont centralisés autour d'une seule entité du réseau, responsable de la génération et de la distribution de la clé de chiffrement de données, ainsi que de l'envoi du flux de données chiffrées. Cette centralisation autour d'un seul serveur de clés, accroît le phénomène "1 affecte n", qui consiste à ce que le changement de l'état d'un membre affecte tous les autres membres du groupe (notamment dans le cadre d'un Join ou un Leave).

Pour atténuer ce phénomène sans utiliser des doubles opérations de chiffrement et de déchiffrement du flux, des protocoles utilisant la clusterisation ont opté pour déléguer la tâche de gestion de clés, à des entités du réseau, autres que le serveur de clés. Ces entités sont les clusterheads dans DMGSA. Pour acheminer la clé de chiffrement de données à leurs membres dans DMGSA, l'envoi de la TEK aux membres locaux s'effectue individuellement (en unicast), entre le clusterhead et tout membre de son cluster, ce qui induit un surcoût de communications, non négligeable dans un environnement ad hoc.

Les protocoles proposés dans [VHS01] et [KLNY03] sont très difficilement applicables pour des équipements de faibles capacité de calcul, puisque des opérations intermédiaires de chiffrement et de déchiffrement de tout le flux multicast sont requises. Ces opérations sont également réalisées par les contrôleurs locaux ou clusterheads, qui deviennent en conséquence des points de vulnérabilités et de goulots d'étranglement.

Les protocoles utilisant la technique TESLA pour assurer l'authentification de la source et l'intégrité des données [ZSXJ04b, KLNY03] souffrent d'un surcoût important de calcul, dû aux opérations de hachage de paquets du coté de la source et des récepteurs, en plus du surcoût de stockage et de la synchronisation temporelle, difficile à assurer dans un MANET.

Surcoût de stockage

La maîtrise du coût de stockage est une véritable contrainte dans les réseaux ad hoc, à faibles capacités de mémoire et de stockage. Le protocole décentralisé proposé par VARADHARAJAN ET AL. ainsi que le protocole centralisé proposé dans [KLNY03], requièrent un coût de stockage important dû aux opérations intermédiaires de déchiffrement et de rechiffrement du flux multicast.

L'algorithme de Prüfer utilisé dans CHIANG ET AL. requiert aussi un grand espace de mémorisation et de calcul, surtout pour un grand nombre de participants. Il est à noter également que tout changement dans la topologie du réseau affecte la séquence de Prüfer et par conséquence l'arbre multicast correspondant. Une forte mobilité des membres du groupe aurait donc un grand impact sur le surcoût de stockage dans CHIANG ET AL..

Le stockage dans LAZOS ET AL. et LKHW concerne les clés de l'arbre LKH dont le nombre dépend du nombre total de membres N (en O(N) pour le contrôleur et en O(log(N)) pour les membres du groupe), alors que GKMPAN et CKDS stockent les clés pré-distribuées pour chaque nœud, indépendamment du nombre total de membres du groupe multicast.

Pour GKMPAN, augmenter le nombre de clé pré distribuées m ou diminuer le nombre de clés disponibles initialement l, augmentera le nombre de chemins directs entre membres du groupe. En effet, le nombre de clés communes que détiennent deux membres est évalué à $\frac{m^2}{l}$. La probabilité que n'importe quels deux nœuds détiennent une même clé est également évaluée à : $P = 1 - \left(\begin{array}{c} l-m \\ m \end{array} \right) / \left(\begin{array}{c} l \\ m \end{array} \right)$. Par exemple, pour m = 100 et l = 2000, 0.5 % des membres recevront les messages de renouvellement de clé indirectement (de la part de leurs voisins). Cependant, il est préférable d'un point de vue capacité de stockage, de diminuer m. Plus m est petit et l est grand, plus le risque de coalition entre les membres est minimal et par conséquent plus le niveau de sécurité est grand et le risque d'attaques faible. Le choix de m et l doit ainsi tenir compte des considérations et de la politique de sécurité de l'application concernée.
Dans le protocole CKDS, le stockage de la matrice EBS du côté du contrôleur global est très contraignant du fait que sa taille est égale à $N * (k + m)$, N est le nombre de membres du groupe et k et m sont les nombres de clés connues (respectivement inconnues) par un membre du groupe dans le système EBS.

Surcoût de communications

Les protocoles centralisés sans pré-distribution de clés n'assurent pas le passage à l'échelle de par leur architecture (phénomène "1 affecte n"). Le protocole proposé par CHIANG ET AL. présente également un problème de passage à l'échelle en terme de communications à cause de l'inondation des localisations GPS à tous les membres du groupe et à l'exécution de l'algorithme de Prüfer, contraignant et coûteux pour un large nombre de participants.
Le protocole DMGSA est contraint par le nombre de membres locaux par cluster, car chaque clusterhead partage avec tout membre de son sous-groupe un secret partagé, pour chiffrer la clé du groupe et l'envoyer de façon sécurisée. De plus, la maintenance distribuée des clusters requiert également des envois périodiques de messages, engendrant un surcoût de communication important.
Dans la version m-dimensionnel du protocole CKDS, l'entité IGD inonde le réseau pour acheminer les nouvelles clés du groupe aux entités locales LQDs. Cette inondation est coûteuse et requiert de plus une opération intermédiaire de déchiffrement et de rechiffrement des clés. En plus, les membres recevant ces messages ne sont intéressés que par un ensemble restreint de ces nouvelles clés et non pas de toutes les clés mises à jour. La version 2D-multicast de CKDS a remédié à ce problème en envoyant un message de distribution de clés, en multicast, aux seuls membres concernés par ces renouvellements.

2.4.4 Vulnérabilités et faiblesses

Les protocoles centralisés ([ZSXJ04b, MME04, KLNY03, LP03, PML+03]), se basent sur une seule entité du réseau ad hoc, qui gère les clés du groupe ou les certificats des membres. Cette entité centrale constitue ainsi un point de vulnérabilité en terme de sécurité. De plus, un serveur centralisé présente un goulot d'étranglement dans une architecture de sécurité et peut être la cible d'attaques de type déni de service.
Étant basée sur les certificats, l'approche de KAYA ET AL. implique que chaque membre du groupe doit stocker son certificat et la liste de révocation envoyée et mise à jour par la source du groupe. Pour éviter que cette liste n'atteigne une taille trop importante, une technique

de suppression d'entrées est mise en place périodiquement, au risque que des membres exclus puissent rejoindre le groupe après un certain délai.
Dans le protocole [VHS01], les clusterheads forment le cœur de routage du réseau et assurent en plus la gestion de clés. Ces entités représentent des cibles privilégiées pour des attaques malicieuses.

2.5 Conclusion

La solution la plus appropriée pour assurer des communications de groupe sécurisées au sein d'un réseau ad hoc est l'établissement d'un protocole de gestion de clé du groupe. Ce protocole doit assurer la confidentialité des données en chiffrant le flux du côté de la source et en le déchiffrant du côté des récepteurs, avec la clé du groupe TEK. De plus, l'authentification et le contrôle d'accès peuvent être assurés car seuls les membres détenant la clé du groupe peuvent accéder aux données.
Cependant, la conception d'un protocole de gestion de clé de groupe dans les MANETs doit être adaptée aux caractéristiques et spécificités d'un tel environnement, telles que la mobilité et la dynamicité des membres, les ressources limitées en terme d'énergie, de bande passante et de capacités de stockage et de calcul, ainsi que l'absence d'infrastructure fixe au sein du réseau. Les services de sécurité offerts par un protocole de sécurité de groupe dans un réseau ad hoc sont également étroitement liés à la nature de l'application à sécuriser et ainsi au niveau de sécurité requis pour les données multicast envoyées par la source, pour faire face aux attaques malicieuses qui peuvent les cibler.

Dans ce chapitre, nous avons passé en revue les protocoles de gestion de clés dans les MANETs, en les classant suivant leur architecture et leur prise en compte des spécificités et caractéristiques des réseaux ad hoc (le support de mobilité, l'optimisation de l'énergie et la conscience des communications multi-sauts). Ensuite, nous avons comparé et discuté les protocoles présentés, selon des métriques bien définies (services de sécurité assurés, contraintes et pré-requis, opérations intermédiaires de chiffrement et de déchiffrement, coût de stockage, passage à l'échelle et vulnérabilité). Nous avons montré que l'approche la plus appropriée des protocoles de gestion de clés est l'approche décentralisée, puisqu'elle ne souffre pas du phénomène "1 affecte n" et ne repose pas sur une seule entité vulnérable. Le passage à l'échelle reste cependant le critère le plus important pour assurer une applicabilité réelle d'un protocole de gestion de clés dans les MANETs. Ce critère n'est pas satisfait au sein du protocole décentralisé proposé par VARADHARAJAN ET AL. [VHS01].

Partant de cette problématique, nous proposons dans un premier temps un protocole décentralisé de gestion de clé de groupe dédié aux MANETs, à "**TEKs locales**". Ce protocole nommé Enhanced BAAL, assure les services de sécurité requis pour sécuriser des communications multicast dans un réseau ad hoc (confidentialité des données, authentification et contrôle d'accès des membres), tout en garantissant les secrets futur (*Forward Secrecy*) et passé (*Backward Secrecy*). Les événements d'ajout ou de retrait d'une entité du groupe requièrent donc des opérations de renouvellement des clés de sessions locales. Enhanced BAAL est ainsi "**orienté récepteurs**", tout en étant adapté à la mobilité et à la dynamicité des MANETs et en assurant le passage à l'échelle.

Des opérations intermédiaires de déchiffrement et de rechiffrement des données sont requises dans Enhanced BAAL, engendrant un surcoût de calcul et de stockage. Les renouvellements de la clé du groupe suite à tout événement de Join ou de Leave dans le groupe (afin

d'assurer les secrets futur et passé) sont également contraignants dans les MANETs. Pour remédier à ces deux limitations, nous proposons dans un second temps un protocole de gestion de clé de groupe dans les MANETs, "**orienté sources**" du groupe, prenant en compte la mobilité et la localisation des membres tout en optimisant la consommation de l'énergie et de la bande passante et en garantissant le passage à l'échelle, dans un modèle de communications de groupe multi-sources séquentielles. Ce protocole dénommé BALADE fait partie des protocoles décentralisés à "**TEK commune**" ; il utilise une seule clé de chiffrement de données TEK, pour tous les membres des clusters du groupe. À notre connaissance, aucun protocole de gestion de clés dans les MANETs n'a adopté cette approche. La source du groupe utilise la clé TEK pour chiffrer le flux multicast et les membres pour le déchiffrer. Les opérations intermédiaires de chiffrement et de déchiffrement du flux multicast ne sont donc plus requises. Le renouvellement de la TEK pour tous les membres du groupe est géré par la source courante et dépend de la sémantique de données émises. La dynamicité des membres du groupe n'influe donc plus sur la clé de chiffrement de données dans BALADE.

Deuxième partie

Contributions

Chapitre 3

Protocole de gestion de clé orienté récepteurs dans les réseaux ad hoc

Sommaire

3.1 Introduction

Nous présentons dans ce chapitre un protocole décentralisé de gestion de clé de groupe dans les réseaux ad hoc. Ce protocole dénommé **Enhanced BAAL** [BCF04] tient compte de la mobilité des membres du groupe, ainsi que de l'absence d'infrastructure fixe des MANETs, tout en assurant les services requis pour la sécurisation des communications de groupe (confidentialité des données, authentification et contrôle d'accès des membres). De par son architecture décentralisée à TEKs locales, Enhanced BAAL est orienté récepteurs membres du groupe (les secrets futur et passé sont assurés).

Le principe de notre approche n'étant pas de développer une nouvelle solution spécifique aux MANETs, mais d'adapter le protocole BAAL [CCS00] de gestion de clés, déjà testé et validé dans le cadre des réseaux filaires, au contexte des réseaux ad hoc. BAAL, tel qu'il a été conçu, n'est pas adapté à la mobilité et à la dynamicité des membres adhérents au groupe multicast. De plus, il souffre du phénomène "1 affecte n" lors des renouvellements de la clé du groupe. Pour remédier à ces limitations, Enhanced BAAL combine l'architecture fonctionnelle du protocole BAAL, avec le support dynamique que fournit le protocole AKMP [BBC02] (*Adaptive Key Management Protocol*).
Pour assurer l'authentification et la génération sécurisée des clés du groupe, nous avons choisi d'utiliser la cryptographie à seuil, jouant le rôle d'une architecture PKI dans le cadre d'un environnement sans fil.

Pour présenter notre approche, ce chapitre est structuré de la façon suivante. La section 3.2 présente les modules fonctionnels utilisés dans la conception de Enhanced BAAL. Dans la section 3.3, nous présentons l'architecture de notre protocole, ainsi que les différents opérations de gestion et de sécurité qu'il garantit. La section 3.4 conclut ce chapitre.

3.2 Les modules fonctionnels

Cette section décrit les blocs fonctionnels que nous avons utilisés pour pouvoir adapter le protocole de gestion de clés BAAL, dans le cadre des réseaux ad hoc. La figure 3.1 montre les trois modules de notre approche étendue :

- l'architecture fonctionnelle du protocole BAAL [CCS00],
- le support hybride du protocole AKMP [BBC02],
- la cryptographie à seuil [ZH99].

3.2.1 Le protocole BAAL

BAAL est un protocole de gestion de clé de groupe défini au sein de l'équipe MADYNES par CHADDOUD ET AL.. BAAL assure la confidentialité des données, l'authentification et le contrôle d'accès des membres du groupe, dans le cadre des réseaux filaires. Il adopte une architecture décentralisée dans laquelle une seule clé de chiffrement de données est utilisée au sein du groupe multicast, renouvelées à tout événement de Join ou Leave dans le groupe. La figure 3.2 comporte une illustration de l'architecture du protocole BAAL, composée des trois acteurs suivants :

1. Le contrôleur global (CG) peut être un organisateur de conférences qui crée un ou plusieurs groupes multicast sur Internet. Il détient la liste des participants du groupe. Le CG génère la clé du groupe, notée K_{grp} et la distribue à tous les participants du groupe,

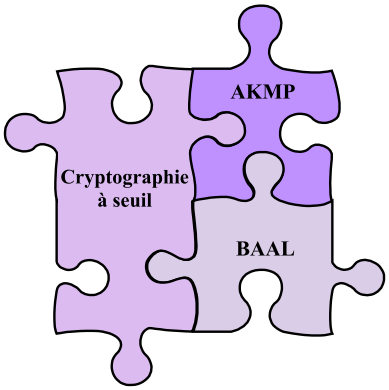


FIG. 3.1 – Architecture étendue de BAAL

via les contrôleurs locaux. Il est également responsable du renouvellement périodique ou occasionnel de cette clé de groupe.

2. Les contrôleurs locaux (CLs) sont délégués par le contrôleur global après authentification. Un contrôleur local est un routeur qui achemine le flux multicast aux adhérents au groupe multicast appartenant à son réseau, comme le montre la figure 3.2. Il reçoit la clé du groupe et la distribue à tous les participants de son réseau, durant la configuration initiale du groupe. Un CL peut créer et distribuer une nouvelle clé du groupe, accepter ou refuser la demande d'adhésion d'un membre et notifier les autres CLs dans le cas de changements du groupe.
3. Les membres du groupe (MGs) correspondent aux membres de la liste des participants, ou à tout membre pouvant joindre le groupe ultérieurement.

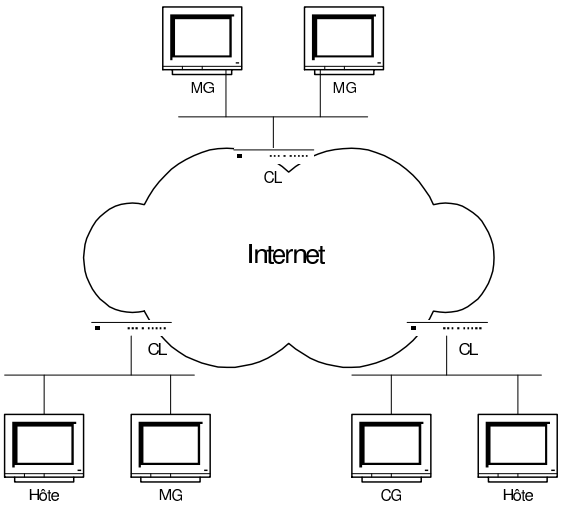


FIG. 3.2 – Architecture de BAAL

Les opérations définies pour la gestion de clés sont les suivantes :

- Initialisation du groupe. Durant cette opération, la clé K_{grp} est distribuée, de façon sécurisée, à tous les éléments de la liste de participants. Le contrôleur global délègue également les contrôleurs locaux, pour pouvoir coopérer et collaborer avec eux pour une meilleure gestion de la clé K_{grp} et un contrôle local efficace des membres du groupe.
  L'initialisation du groupe est réalisée en deux phases : une phase d'invitation qui permet au CG d'inviter tous les membres de la liste des participants à se joindre au groupe multicast et une phase de distribution de la clé du groupe et de délégation de CLs.
- Ajout d'un nouveau membre. Pour se joindre à un groupe multicast, une entité doit auparavant être autorisée à le faire. Cette condition est vérifiée par le contrôleur local, qui décide d'ajouter ou non le membre au groupe. En cas de succès du contrôle d'accès, un processus de renouvellement de la clé du groupe doit être déclenché, pour préserver la propriété de secret passé.
  Si le CL est déjà délégué, il génère et distribue une nouvelle clé à tous les membres du groupe, sinon il négocie avec le CG pour obtenir la permission de déclencher le processus de renouvellement de la clé K_{grp}.
- Retrait d'une entité du groupe. Un membre voulant quitter le groupe, envoie un message à son CL afin de ne plus recevoir le trafic multicast émis par la source. Aussi, un membre classé indésirable, est automatiquement exclu du groupe. Un processus de renouvellement de la clé K_{grp} est également déclenché après le départ d'une entité du groupe, afin de préserver la propriété de secret futur.
- Renouvellement périodique de la clé. Les clés cryptographiques ont une durée de vie limitée et doivent être renouvelées périodiquement. Ce renouvellement peut être à la charge du CG ou des CLs.

3.2.2 Le protocole AKMP

De par son architecture statiquement décentralisée à TEK commune, le protocole BAAL n'est pas adapté à la dynamicité et la mobilité des membres d'un groupe multicast dans un réseau ad hoc. De plus, BAAL souffre du phénomène "1 affecte n" lors des renouvellements de la clé du groupe après chaque événement d'ajout ou de retrait d'une entité dans le groupe.

Pour remédier à ces limites, nous avons choisi d'intégrer dans le protocole Enhanced BAAL le support dynamique du protocole AKMP [BBC02]. AKMP (*Adaptive Key Management Protocol*) est un protocole de gestion de clé de groupe, défini pour opérer dans les réseaux filaires, tenant compte de la dynamicité des membres et atténuant le phénomène "1 affecte n". L'idée de base de AKMP est de commencer une session d'un groupe multicast en adoptant une approche centralisée et de diviser le groupe en clusters dès que la fréquence d'adhésion au groupe atteint un certain seuil. La dynamicité des membres d'un cluster n'affecte donc pas les autres clusters du groupe.

À l'initialisation, un seul groupe est créé, partageant une TEK unique, gérée par un routeur AKMP. Durant la session multicast sécurisée, si un routeur AKMP détecte une forte dynamicité locale, il passe de l'état passif à l'état actif : il initie et crée son propre sous-groupe, ensuite il génère une clé locale de chiffrement de données indépendante, notée DK (*Downstream Key*), qu'il distribue à tous ses membres locaux.

Pour acheminer le flux multicast émis par une source, un routeur AKMP déchiffre les paquets reçus avec la clé de son routeur AKMP parent, notée UK (*Upstream Key*), les rechiffre avec la clé DK et les envoie à ses membres locaux. AKMP régule ainsi le surcoût dû aux

opérations de chiffrement et de déchiffrement de données, tout en atténuant le phénomène "1 affecte n".

Au sein de chaque routeur AKMP, est implantée une fonction d'évaluation f, qui décide l'état du routeur (passif ou actif), selon le nombre de changements locaux des membres de son sous-groupe. Quand un routeur AKMP i détecte une forte dynamicité au sein de son sous-groupe (f_i = true), il passe à l'état actif, génère une nouvelle clé DK_i, et la distribue à tous ses membres locaux. Ensuite, il doit envoyer sa nouvelle clé locale à son routeur AKMP parent j, qui doit à son tour générer et distribuer une nouvelle clé DK_j à tous ses membres locaux.

Si (f_i = false), le routeur AKMP reste à l'état passif. Ainsi, quand il détecte un événement Join ou Leave au sein de son sous-groupe, il doit notifier son routeur AKMP parent j, afin qu'il déclenche un processus de mise à jour de la clé DK_j.

3.2.3 La technique de cryptographie à seuil

BAAL utilise une infrastructure à clé publique, ce qui implique la présence d'une autorité de certification. Cependant, dans un réseau ad hoc, la présence d'une seule autorité de certification représente un point de vulnérabilité et de faiblesse. Si la disponibilité de cette autorité n'est pas assurée, la sécurisation des communications entre les membres du réseau devient impossible. Des attaques peuvent également viser ce point de vulnérabilité, pour compromettre tout le réseau. La duplication de l'autorité de certification pourrait apporter une meilleure disponibilité, mais augmente aussi les risques d'attaques malicieuses du fait qu'elle duplique également la possibilité de compromettre une autorité de certification.

La cryptographie à seuil [ZH99, YK02] résout ce problème. Cette technique de certification consiste à partager la confiance de l'autorité de certification entre des nœuds ayant une sécurité physique et une puissance de calcul, relativement élevées.

Enhanced BAAL utilise la technique de cryptographie à seuil (B-Unicast) proposée dans [YK02] (cf. Chapitre 1), pour assurer l'authentification et le contrôle d'accès des membres adhérents au groupe multicast ainsi que pour garantir une génération sécurisée des clés de chiffrement du groupe.

3.3 Protocole hybride de gestion de clé de groupe dans les réseaux ad hoc

Le contexte de ce protocole de gestion de clés est un ensemble de nœuds ad hoc, ayant la capacité de communiquer en unicast et en multicast. Le but de cette proposition est de mettre en place un ensemble de mécanismes permettant d'assurer les différents services de sécurité pour des communications de groupe (l'authentification, la confidentialité, l'intégrité et la non répudiation), pour sécuriser des données de haut niveau de sensibilité. Assurer les secrets futur et passé est donc indispensable au sein de cette architecture orientée récepteurs.

Les opérations définies pour la gestion de la clé du groupe sont les mêmes que celles définies dans BAAL : l'initialisation du groupe (distribution de la clé du groupe), l'ajout et le retrait d'une nouvelle entité (renouvellement de la clé du groupe) et le renouvellement périodique de la clé du groupe.

Au sein de notre environnement ad hoc, est établi un mécanisme de cryptographie à seuil, assurant :

- la génération d'une clé publique et privée (K_i , k_i) à toute entité i du réseau,
- la génération des clés de chiffrement des données du groupe multicast.

3.3.1 Architecture

Les acteurs principaux de notre approche sont le contrôleur global (CG), les contrôleurs locaux (CLs) et les membres du groupe (GMs).

- Le contrôleur global (**CG**) est la source du groupe multicast. Initialement, cette entité détient la liste des participants au groupe. Durant la phase d'initialisation, le contrôleur global est le responsable de la génération de la clé du groupe. Il assure également le renouvellement périodique de la clé du groupe et la gestion de la sécurité au sein du groupe (contrôler les comportements des contrôleurs locaux et des membres du groupe). Tous les contrôleurs (global et locaux), détiennent la même liste de révocation, contenant les membres exclus du groupe. Cette liste est utilisée lors des contrôles d'accès des nouvelles entités voulant se joindre au groupe. La cohérence et la distribution de cette liste de révocation aux CLs est à la charge du contrôleur global.
- Les contrôleurs locaux (**CLs**). Tout nœud mobile appartenant à l'arbre multicast et ayant des nœuds fils auxquels il achemine le trafic multicast, est considéré comme contrôleur local passif. Chaque contrôleur local détient la liste de ses membres locaux, auxquels il doit acheminer le trafic multicast émis par la source. Un contrôleur local passif détient la même clé de chiffrement que son nœud parent, il reçoit le trafic multicast et l'achemine vers ses membres locaux qui détiennent eux aussi la même clé de chiffrement. Lorsque le taux local de dynamicité ou le nombre de membres locaux atteignent des seuils définis, le contrôleur local décide de passer à l'état actif, c'est ainsi qu'il génère une clé de chiffrement locale et forme avec ses membres locaux un sous-groupe ou cluster.
  Chaque contrôleur local actif doit assurer un renouvellement périodique de sa clé locale. En d'autres termes, il joue le même rôle que le contrôleur global, au sein de son sous-groupe.
  Pour décider de son état, chaque contrôleur local détient une fonction d'évaluation de dynamicité, présentée dans l'algorithme 1 :

Algorithm 1 Fonction d'évaluation

if (mcf > d1 or mn > d2) **then**
  fi = true ; // état actif
else
  fi = false ; // état passif
end if

Avec :
  mcf : nombre de changements des membres par unité de temps,
  d1 : un seuil de fréquence prédéfini,
  mn : nombre de membres locaux,
  d2 : un seuil de membres prédéfini.

Cette fonction d'évaluation diffère de celle de AKMP, car elle tient en compte en plus

de la fréquence de changements des membres, de leur nombre. Cette modification est nécessaire dans le cadre de la sécurité des communications dans les réseaux ad hoc pour deux raisons. Premièrement, tous les membres du groupe sont également des routeurs et peuvent être considérés comme des contrôleurs locaux s'ils ont à leur charge des membres locaux. De plus, lorsqu'un membre quitte le groupe multicast, le contrôleur local actif doit renouveler la clé de chiffrement locale et la distribuer à ses membres locaux, en unicast.
mn est calculé de la façon suivante : un contrôleur local compte le nombre de membres passifs qui lui sont attachés et leurs descendants et le nombre de membres actifs qui lui sont attachés sans leurs descendants. La figure 3.3 illustre un exemple de calcul de *mn*, d2 étant choisi égal à 4.

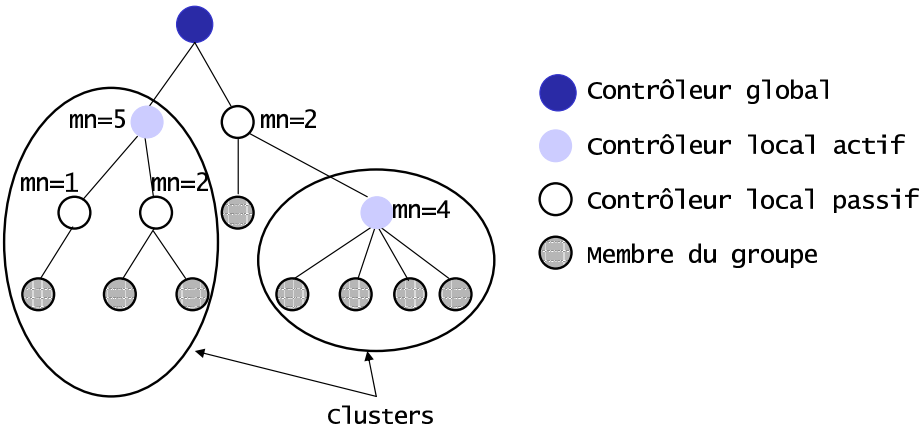


FIG. 3.3 – Exemple d'évaluation du nombre de membres

– Les membres du groupe (**MGs**) sont les membres de la liste des participants ou tout membre qui rejoint le groupe ultérieurement.

3.3.2 Opérations de gestion de clés

Initialisation du groupe

Le contrôleur global initialise les deux listes de participants et de révocation. Ensuite, il entame la phase de génération et de distribution de la clé du groupe. La génération de cette clé est réalisée via la technique de cryptographie à seuil, en adoptant l'optimisation B-unicast, comme cela est illustré dans la figure 3.4.
Si une configuration $(n, t+1)$ est établie (cf. section 1.3.1), le CG consulte sa table de routage et vérifie s'il connaît des routes vers $t+1$ serveurs. Dans ce cas, il leur envoie une requête de génération de clé *Key_ Query*. Autrement, il diffuse ce message dans tout le réseau.

À la réception de cette requête, un serveur commence par authentifier le contrôleur global. Si l'authentification réussit, il génère une signature partielle et l'inclut dans un message *Key_ Resp*. Ce message sera envoyé au CG, chiffré avec sa clé publique.

Le contrôleur global reste en attente des $t+1$ premières signatures partielles, envoyées par les premiers $t+1$ serveurs. À leur arrivée, il les combine pour obtenir une clé de chiffrement de données, authentifiée par les serveurs de cryptographie à seuil. Ces serveurs détiennent les

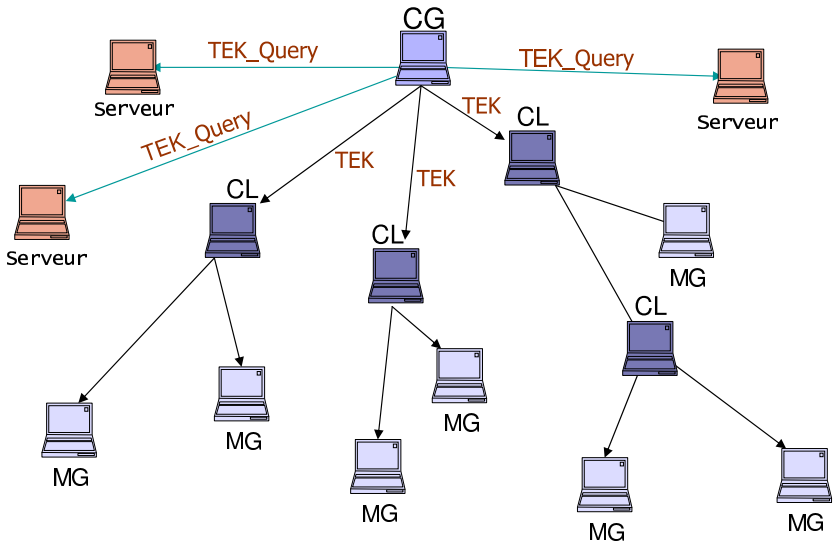


FIG. 3.4 – Génération et distribution de la TEK dans Enhanced BAAL

algorithmes et connaissances nécessaires pour générer aléatoirement des signatures partielles pour les autres entités du réseau.

Afin de sécuriser les communications durant la phase de distribution de clés, les messages envoyés incluent un jeton signé avec la clé privée de l'émetteur, permettant aux récepteurs de pouvoir vérifier l'authenticité de la source du message.

Un jeton contient :

- l'identité de l'émetteur, par exemple son adresse IP ;
- une estampille ;
- un nombre aléatoire, utilisé pour protéger les récepteurs contre les attaques de rejeu de messages.

Le CG envoie le message **(1)** aux membres de la Participant_List (MG_i), chiffré avec leurs clés publiques respectives (Pub_MG_i). Ce message contient la clé de groupe, l'identité du groupe, l'identité et le jeton signé du CG.

Dès la réception de ce message, chaque membre du groupe le déchiffre, extrait la clé du groupe. Ensuite, il répond avec un message d'acquittement **(2)**, contenant l'identité du groupe ainsi que son identité et son jeton signé, chiffré avec la clé publique du CG.

---

(1) $CG \longrightarrow MG_i : \{K_{grp}, IDG, ID_{CG}, [jeton\_CG]`Pri\_CG\}`Pub\_MG_i$
avec IDG : *identité du groupe,* ID_{CG} : *identité du CG,* $[jeton\_CG]`Pri\_CG$: *jeton signé du CG.*

(2) $MG_i \longrightarrow CG$: $\{IDG, ID\_M_i, [jeton\_M_i]`Pri\_M_i\}`Pub\_CG$
avec Pub_CG : *clé publique du CG,* $[jeton\_M_i]`Pri\_M_i$: *jeton signé de* M_i.

Ajout d'un nouveau membre

Une entité voulant joindre le groupe, envoie un message *Report* à son CL (message **3**). Ce message contient son jeton signé, son identité et l'identité du groupe multicast.

(**3**) $M_i \longrightarrow CL : \{IDG, ID\_M_i, [jeton\_M_i]`Pri\_M_i\}`Pub\_CL$
avec CL : *le contrôleur local de la nouvelle entité.*

Dès la réception de ce message, le CL authentifie le jeton signé. En cas de succès, il vérifie si la nouvelle entité n'a pas été exclue du groupe et donc n'appartient pas à la liste de révocation. À ce niveau, le CL exécute sa fonction d'évaluation. Deux cas se présentent :

1. Si (f_i = true) : le CL passe à l'état actif. Ainsi, il doit générer une nouvelle clé locale K_i^{loc}, en utilisant la cryptographie à seuil, la distribuer à tous ses membres locaux, chiffrée avec leur ancienne clé locale old$_K_i^{loc}$ (message **4**) et l'envoyer au nouveau membre, chiffrée avec sa clé publique (message **5**).
   De plus, le CL doit envoyer son ancienne clé locale à son CL parent actif M_l (message **6**), afin qu'il puisse mettre à jour sa clé K_l^{loc} dans le cas où old$_K_i^{loc} = K_l^{loc}$ (quand le CL passe pour la première fois à l'état actif, sa clé est la même que celle de son CL parent).

(**4**) $for j : 1..nb\_anciens\_membres$
$CL \longrightarrow M_j : \{K_i^{loc}\}`old\_K_i^{loc}$

(**5**) $CL \longrightarrow M_i : \{K_i^{loc}\}`Pub\_M_i$
avec M_i : *nouveau membre*

(**6**) $CL \longrightarrow Active\_Parent\_Node\_M_l : \{old\_K_i^{loc}\}`Pub\_M_l$
avec Pub_M_l : *clé publique de* M_l.

2. Si (f_i = false) : Le CL reste à l'état passif. Il notifie donc son CL parent du Join du nouveau membre et lui demande de déclencher un processus de génération et de renouvellement de la clé locale.

Retrait d'une entité du groupe

Cette opération distingue deux cas : retrait volontaire et expulsion. Le premier cas correspond à un membre qui décide de quitter le groupe. Il envoie donc un message Leave à son CL, contenant son jeton signé. Le CL supprime cette entité de sa liste de membres locaux et entame ensuite une procédure de renouvellement de la clé.

Le deuxième cas de retrait correspond à une révocation d'un membre du groupe. Le CL ajoute cette entité à la liste de révocation, la supprime de sa liste de membres locaux et commence la procédure de renouvellement de la clé locale.

Durant la phase de renouvellement de la clé locale, deux cas sont identifiés :

1. Si (f_i = true) : le CL passe à l'état actif. Ainsi, il doit générer une nouvelle clé locale K_i^{loc} , en utilisant la cryptographie à seuil, la distribuer à tous ses membres locaux, à l'exception du membre quittant le groupe, chiffrée avec leurs clés publiques respectives (message **7**).
   De plus, le CL doit envoyer son ancienne clé locale à son CL parent actif M_l (message **8**), afin qu'il puisse mettre à jour sa clé K_l^{loc} si $old\_K_i^{loc} = K_l^{loc}$.

   (**7**) M_s *membre quittant le groupe,*
   $for j : 1..nb\_anciens\_membres,\ j\ diff\acute{e}rent\ de\ s,$
   $CL \longrightarrow M_j : \{K_i^{loc}\}`Pub\_M_j$

   (**8**) $CL \longrightarrow Noeud\_Parent\_Actif\ M_l : \{old\_K_i^{loc}\}`Pub\_M_l$
   avec Pub_M_l : *clé publique du parent actif* M_l.

2. Si (f_i = false) : Le CL reste à l'état passif. Il notifie donc son CL parent du retrait du membre du groupe et lui demande de déclencher un processus de génération et de renouvellement de la clé locale.

Renouvellement périodique de la clé du groupe

La période de renouvellement de clés est déterminée en fonction de la taille de la clé et de l'algorithme de génération avec lequel elle a été créée. Le renouvellement de clés doit être effectué par le contrôleur global et tous les contrôleurs locaux à l'état actif; ce renouvellement se fait en deux étapes : génération d'une nouvelle clé en utilisant la cryptographie à seuil et distribution de cette clé à tous les membres locaux.

3.3.3 Gestion de la mobilité

Notre protocole de gestion de clés doit être adapté à la mobilité des nœuds, qui représente une caractéristique et le principal challenge dans les réseaux ad hoc. Nous étudions dans ce qui suit des scénarii de mobilité, incluant les principaux acteurs de notre architecture de distribution de clés :

1. Quand un CL se déplace, tous ses membres locaux ne pourront plus accéder au trafic multicast émis par la source. Pour remédier à ce problème, ces membres doivent immédiatement s'attacher à un autre contrôleur local et pouvoir ainsi continuer à recevoir le flux multicast. Reste à voir comment est réalisée cette transition entre deux clusters et pendant combien de temps les membres transitant d'un cluster à un autre ne pourront plus accéder aux trafic multicast. Deux cas sont distingués :

   (a) Le CL se déplace avec notification : cette notification est un message envoyé par le CL, en multicast, à tous les membres de son cluster. Ces membres envoient alors des messages *Report* à d'autres CLs afin de joindre leurs clusters et continuer à accéder au flux multicast.

   (b) Le CL se déplace sans notification : les membres du cluster local se rendent compte, après un certain délai, que la route vers la source du groupe n'est plus assurée par

leur contrôleur local. Ils essayent donc de joindre le groupe multicast à travers d'autres CLs.

Ces solutions sont dépendantes des politiques de sécurité et de qualité de service (QoS) établies au sein du réseau ad hoc. Selon ces politiques, une latence nécessaire pour un membre en déplacement pour accéder de nouveau au flux multicast est permise ou non.

2. Quand un membre du groupe se déplace, il risque de perdre sa connectivité avec son contrôleur local. Pour cela, il doit prévoir un autre chemin vers la source, en s'attachant à un autre contrôleur local. La solution est donc d'essayer de s'authentifier auprès d'autres contrôleurs locaux, dès le début de son déplacement.

3.3.4 Estimation des seuils de fréquence et du nombre de membres

Dans cette section, nous présentons des simulations réalisées dans le but de définir les seuils de fréquence et de nombre de membres, pour paramétrer la fonction d'évaluation au sein de chaque CL. Au delà de ces seuils, un CL passe de l'état passif à l'état actif et forme avec ses membres locaux un cluster. Nous avons mesuré le temps nécessaire au renouvellement de la clé du groupe, suite à un événement de Join et à un événement de Leave, par rapport au nombre de membres du groupe et par rapport à la fréquence d'adhésion au groupe.

Les deux seuils de fréquence et de nombre de membres apportent une évaluation de notre solution. En effet, ils garantissent que le temps nécessaire pour le renouvellement de la clé suite à un join ou un leave, ne peut pas dépasser une valeur maximale. En fixant ces deux seuils, nous pourrons également faire varier la configuration de la cryptographie à seuil, afin d'améliorer les performances de notre architecture de gestion de clés.

Nous avons utilisé le simulateur NS, version ns2.1b9a. Le réseau simulé est un réseau ad hoc, composé de 100 nœuds et utilisant MAODV comme protocole de routage multicast. Pour générer des sessions réelles multicast, on utilise le modèle présenté par ALMEROTH [AA96], qui suggère que l'arrivée des membres suit un processus poissonien de paramètre $\lambda = 10$ et que la durée d'appartenance à un groupe suit une distribution exponentielle de paramètre $\mu = 145$. Ce modèle a été déduit à partir de sessions multicast réelles observées au sein du Mbone.

Nous avons remarqué que le temps de renouvellement de la clé suite à un Join n'est pas proportionnel au nombre de membres de groupe. Cela s'explique par le fait que le coût engendré par le renouvellement de la clé est constant. Les différentes variations que nous avons pu constater sont dues à un temps de latence très variable entre une demande de Join et le Join effectif ; ce temps est dépendant du protocole de routage MAODV et de la localisation des membres du groupe. Nous ne pouvons donc pas définir un seuil relatif au nombre de membres locaux, suite à cette étude du temps de renouvellement de la clé après à un événement de Join.

La figure 3.5 montre que le temps de renouvellement de la clé suite à un Leave est proportionnel au nombre de membres de groupe. Ceci nous permet de fixer un seuil de nombre de membres pour former un cluster. Si nous prenons comme contrainte que le temps de renouvellement de la clé suite à un Leave ne doit pas dépasser 0.05s, le seuil sera alors de 8 membres par cluster.

Pour réaliser la figure 3.6, nous avons calculé le temps de renouvellement moyen de la clé du groupe lors d'un Join ou un Leave, par rapport à des fréquences d'événements (d'ajout ou de retrait d'entités) calculées sur des intervalles de temps égaux. Si nous prenons comme deuxième contrainte que le temps de renouvellement de la clé par rapport à la fréquence

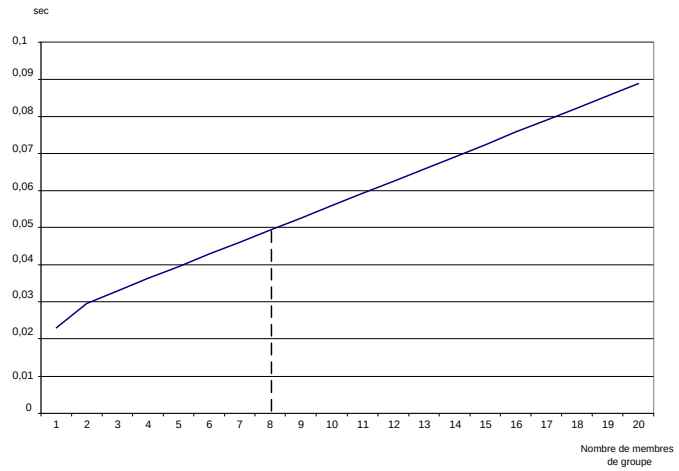


FIG. 3.5 – Temps de renouvellement de la clé suite à un Leave par rapport au nombre de membres

d'adhésion ne doit pas dépasser 1s, la figure 3.6 montre que, une fois le régime permanent établi, définir le seuil de fréquence à 9 événements par unité de temps peut satisfaire cette contrainte.

3.4 Conclusion

Dans ce chapitre, nous avons présenté une architecture étendue de gestion de clé de groupe dans les réseaux ad hoc, basée sur le protocole BAAL, validé et mis en place dans les réseaux filaires.

Pour adapter BAAL au contexte des MANETs, nous avons utilisé le support hybride de AKMP afin d'assurer le passage à l'échelle et de prendre en compte la dynamicité des membres des groupes multicast. Notre approche étendue assure également l'authentification des membres et de la source du groupe, ainsi que la génération sécurisée des clés de chiffrement, via la technique de cryptographie à seuil.

Enhanced BAAL relève certains défis posés dans le contexte de la sécurisation des communications multicast dans les réseaux ad hoc. Le problème d'absence d'infrastructure fixe est résolu via l'utilisation de la cryptographie à seuil. En effet, cette technique consiste à partager une autorité centrale de certification en plusieurs nœuds du réseau, appelés serveurs. Ces serveurs partagent la capacité de signer des certificats aux autres membres du réseau. Une phase de configuration de la cryptographie à seuil reste toutefois indispensable hors ligne.

Le support de AKMP assure la dynamicité et le passage à l'échelle de notre approche. En effet, AKMP diminue le phénomène "1 affecte n", rencontré dans BAAL, tout en atténuant le surcoût des opérations de chiffrement et de déchiffrement du flux émis par la source, suite à la clusterisation du groupe. Notre approche permet désormais une clusterisation dynamique du groupe multicast, tenant compte de la fréquence d'adhésion et du nombre de membres locaux par cluster.

L'architecture décentralisée à TEKs locales adoptée par Enhanced BAAL, le rend orienté récepteurs du groupe. Les secrets futur et passé sont donc garantis au sein de ce protocole,

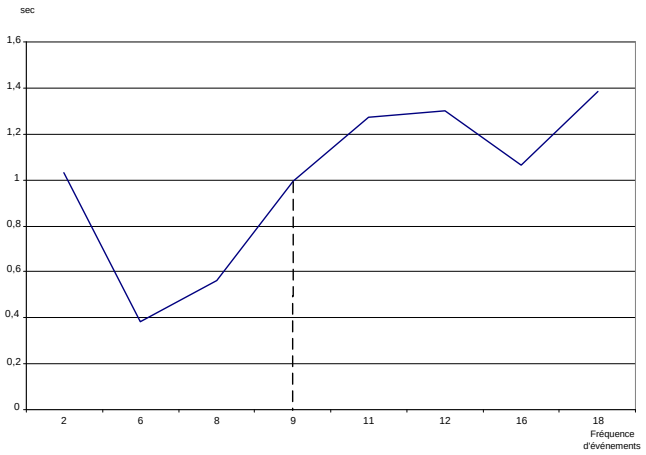


FIG. 3.6 – Temps de renouvellement de la clé par rapport à la fréquence d'événements

le rendant adapté à la sécurisation des communications de données hautement sensibles. Cependant, les contrôleurs locaux doivent effectuer des opérations de déchiffrement et de rechiffrement des données multicast à chaque passage d'un cluster à un autre. Ces opérations sont contraignantes dans un environnement ad hoc, où les capacités de stockage sont limitées. Les contrôleurs locaux responsables de la gestion de la sécurité au sein de leur cluster représentent des goulots d'étranglement dans le réseau.
De plus, les TEKs sont distribuées de façon sécurisée, chiffrées avec les anciennes TEKs. Si une clé de chiffrement de données est corrompue, toute la chaîne de clés TEKs le sera également.

Pour remédier à ces problèmes, nous présentons dans le chapitre suivant un protocole décentralisé de gestion de clé de groupe, dans les réseaux ad hoc, dénommé BALADE. Ce protocole, orienté sources du groupe, élimine les doubles opérations de déchiffrement et de chiffrement, tout en assurant un service de distribution de clés efficace, rapide et adapté aux MANETs.

Chapitre 4

Protocole de gestion de clé orienté sources dans les réseaux ad hoc

Sommaire

4.1 Introduction

Les applications multicast dans les MANETs présentent plus de vulnérabilités en terme de sécurité que les communications point à point et disposent de critères de performances spécifiques quant à la tolérance aux pertes de données, au faible surcoût en communication, au passage à l'échelle de la taille du groupe, ... Nous présentons dans ce chapitre le protocole de gestion de clé de groupe que nous avons défini dénommé **BALADE**[11] [BCF06a, BCF05a]. Ce protocole est conçu pour sécuriser des communications de groupe selon le modèle multi-sources séquentielles, tenant compte des contraintes des MANETs. BALADE assure une distribution décentralisée de clés, efficace et fiable. Le renouvellement de clés est réalisé par la source, prenant en compte la sémantique des données émises (orienté sources du groupe).

Afin de présenter BALADE, ce chapitre est structuré comme suit. Dans la section 4.2, nous présentons nos motivations et choix de conception de BALADE. La section 4.3 présente l'architecture de sécurité BALADE. La section 4.4 comporte une description du fonctionnement global du protocole BALADE. La section 4.5 conclut ce chapitre.

4.2 Motivations et choix

Les services que se propose de réaliser BALADE doivent être adaptés à l'environnement ad hoc, tout en assurant la sécurité requise par les communications de groupe, adoptant le modèle multi-sources séquentielles (modèle 1 vers n séquentiel). Selon ce modèle, à tout instant de la session d'un groupe multicast, une seule source émet un flux de données aux membres du groupe. Dès qu'elle a fini, une autre source prend le relais. Nous avons opté pour ce modèle de communications dans BALADE car il correspond à la plupart des applications de groupe, telles que l'audio et vidéo conférences, les jukebox distribués, ...

BALADE assure la confidentialité des données, ainsi que l'authentification et le contrôle d'accès des membres du groupe. La confidentialité des données implique que seuls les membres du groupe doivent être capables d'accéder au flux multicast. L'authentification permet à un nœud de s'assurer de l'identité des nœuds avec lesquels il communique. Le contrôle d'accès assure que l'accès au groupe est réservé aux membres qui appartiennent à une liste ACL (*Access Control List*) et qui n'ont pas été exclus du groupe. Le contrôle d'accès est effectué à chaque fois qu'un nœud veut joindre le groupe.

BALADE assure également l'accessibilité au flux de données (capacité des récepteurs à accéder au service de réception du flux multicast depuis n'importe quel point du flux). Le délai d'accessibilité évalué comme étant le nombre maximum de messages qu'un nouveau membre doit échanger pour accéder au flux multicast, est optimisé dans BALADE.

La mobilité et la dynamicité des nœuds sont aussi prises en compte dans BALADE, afin d'assurer une distribution de clés efficace et adaptée à l'environnement ad hoc.

L'architecture de BALADE est plate (de bout en bout). Le flux de données est ainsi chiffré du côté de la source et n'est déchiffré que par les récepteurs finaux. Aucune opération intermédiaire de chiffrement et de déchiffrement du flux n'est requise dans BALADE.

Le passage à l'échelle est l'une des principales motivations de BALADE, en termes de coût de communication, de calcul, de stockage et de bande passante.

Concernant l'authentification des membres et des sources dans le cadre de notre approche, nous avons choisi d'utiliser les identificateurs cryptographiques [MC02, BCK96] (cf. Chapitre

[11] Le terme BALADE a été choisi car il rappelle les notions de promenade, de mobilité et de musique

1). Ces identificateurs sont statistiquement uniques et cryptographiquement vérifiables et permettent une forte liaison cryptographique avec leurs composants (clés privée et publique). Dans le cadre de notre approche, l'identification et le contrôle d'accès se font grâce à une liste ACL distribuée qui contient tous les CBIDs des nœuds autorisés à joindre le groupe.

L'architecture de BALADE est basée sur des contrôleurs qui supervisent les membres du groupe multicast. Ces contrôleurs peuvent accéder à la clé de chiffrement de données TEK. C'est pour cette raison qu'un contrôleur local doit impérativement être un membre du groupe. De plus, tous les contrôleurs locaux sont des nœuds ad hoc et non des routeurs comme dans le cas des réseaux filaires et il n'est pas raisonnable qu'un nœud ad hoc non adhérent au groupe se porte volontaire pour assurer le rôle d'un contrôleur local. Un contrôleur local doit obtenir la permission de son contrôleur local parent pour former et gérer son propre sous-groupe. C'est donc le contrôleur local parent qui vérifie l'authenticité du contrôleur local en question et son appartenance au groupe multicast.

4.3 L'architecture de sécurité BALADE

L'architecture de sécurisation des communications de groupe BALADE réalise plusieurs opérations de gestion et de sécurité dans les réseaux ad hoc. Ces opérations se focalisent principalement sur la gestion des membres et des contrôleurs du groupe, la gestion et la fiabilité de la distribution des clés du groupe, la diffusion des données sécurisées et la gestion de la mobilité. Nous les décrivons dans ce qui suit.

4.3.1 Gestion des membres du groupe : clusterisation dynamique

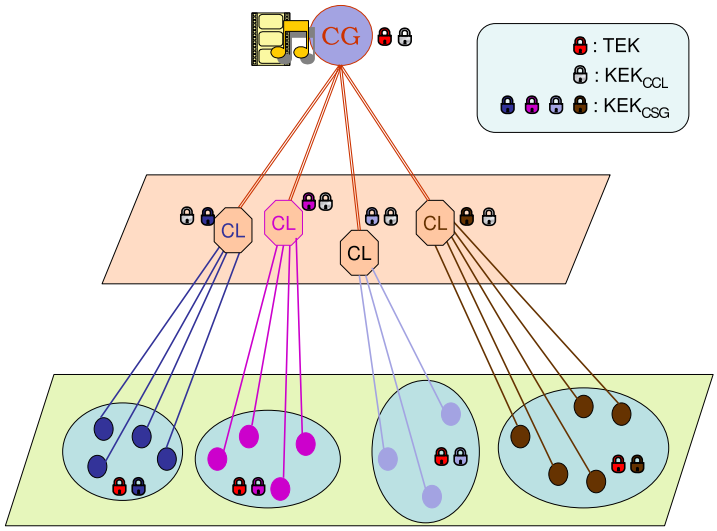


FIG. 4.1 – Architecture globale de BALADE

Comme acteurs principaux de l'architecture BALADE, nous retrouvons ceux définis dans Enhanced BAAL [BCF04] :

- Le Contrôleur Global (**CG**) est associé à la source courante du groupe multicast. Avec une diffusion séquentielle 1 vers n, à un instant donné il existe un seul CG au sein du groupe. Cette entité est responsable de la génération de la clé de chiffrement (TEK), du chiffrement des données et de la distribution des données chiffrées à tous les membres du groupe. La clé de chiffrement du trafic est distribuée séparément via les contrôleurs locaux. Le CG assure aussi le renouvellement de la clé de chiffrement de trafic (TEK) à chaque unité de données, selon la sémantique du flux.
  Tous les contrôleurs (global et locaux) détiennent les mêmes listes ACL et RL (liste de contrôle d'accès et liste des membres bannis). Ces listes seront utilisées lors du contrôle d'accès d'un nouveau membre au groupe. La génération et la cohérence des ces listes sont à la charge d'un algorithme de gestion distribué, présenté dans le chapitre 6.
- Les Contrôleurs Locaux (**CLs**) représentent tout nœud mobile membre du groupe, formant avec ses membres locaux un sous-groupe ou cluster. Ils doivent générer et distribuer une clé à leurs différents membres locaux. Cette clé est appelée KEK_CSG : clé du sous-groupe. Chaque CL détient sa liste de membres locaux. Il doit transmettre la clé de chiffrement de trafic, envoyée par la source du flux, à tous les membres de cette liste. Le flux chiffré étant diffusé séparément.
  Un simple membre du groupe décide de passer à l'état CL si le taux de dynamicité locale (nombre de *Join* et de *Leave* par unité de temps) ou le nombre de membres locaux dépassent des seuils prédéfinis.
  Pour décider de son état, chaque membre détient une fonction d'évaluation [BCF04], implantant l'algorithme 1 présenté dans le chapitre 3. Cette fonction d'évaluation prend en compte la fréquence d'adhésion au groupe et le nombre de membres locaux.
  Il est à noter qu'un membre doit demander l'autorisation auprès de son CL parent pour pouvoir passer à l'état CL. Le CL parent authentifie le membre en question, s'assure qu'il est bien un membre du groupe (pour assurer la confidentialité de la TEK) et en cas de succès, l'autorise à joindre le GCL (Groupe des Contrôleurs Locaux).
- Les Membres du groupe (**MGs**) correspondent aux membres appartenant à la liste ACL (Access Control List).

4.3.2 Diffusion des données sécurisées

La source chiffre les données avec une clé TEK (Traffic Encryption Key). L'algorithme de chiffrement adopté doit être adapté aux flux de données, comme par exemple RC4[12]. Ensuite, la source achemine les données sécurisées via l'arbre multicast de diffusion de données. Les récepteurs ayant reçu la clé de chiffrement TEK par le processus de distribution de clés (présenté ci-dessous), pourront déchiffrer le flux. La figure 4.2 illustre ce processus de diffusion des données sécurisées dans BALADE.

4.3.3 Gestion et distribution des clés

À l'initialisation de l'application, tous les membres du groupe reçoivent de la part de la source la clé de session, appelée KEK_CSG_0 (clé du sous-groupe 0). Ensuite, dynamiquement, de nouveaux sous-groupes vont se créer. Chaque sous-groupe i aura un contrôleur local CL_i et partagera une clé de sous-groupe KEK_CSG_i.

[12] http ://www.rsasecurity.com/

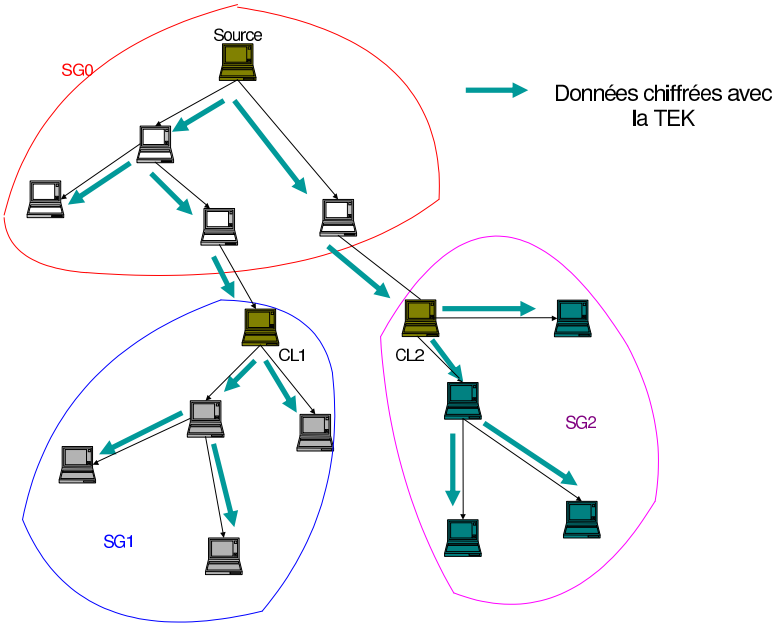


FIG. 4.2 – Diffusion des données sécurisées dans BALADE

Pour envoyer la clé TEK à tous les membres du groupe, la source la chiffre avec la clé KEK_CSG_0 et l'envoie à tous les membres de son sous-groupe. Puis elle envoie cette clé TEK au groupe formé par les contrôleurs locaux (ce groupe partage une clé de groupe appelée KEK_CCL : clé des contrôleurs locaux), chiffrée avec la KEK_CCL. Les contrôleurs locaux appartenant à ce sous-groupe vont déchiffrer le message, extraire la TEK, la réencrypter avec leur clé de sous-groupe respective et l'envoyer à tous leurs membres locaux.

L'avantage de cette solution est de faciliter le processus de chiffrement/déchiffrement aux contrôleurs locaux, qui n'ont qu'à chiffrer et déchiffrer la clé de chiffrement et non plus tout le flux multicast (de façon similaire au protocole de gestion de clé de groupe DEP [DMS99] dédié aux réseaux filaires). Il est à noter que chaque nouvelle source doit joindre le groupe des contrôleurs locaux, pour pouvoir envoyer la clé de chiffrement du trafic cryptée avec la KEK_CCL à tous les autres CLs. Lorsqu'on passe d'une source à une autre séquentiellement, l'arbre de distribution de clés reste potentiellement inchangé. Une source doit, à la fin de sa diffusion, vérifier si elle a des membres fils qui ne sont pas des CLs. Si elle n'en a pas, elle doit quitter le groupe des CLs. Une illustration de la distribution de TEK est donnée dans la figure 4.3.

4.3.4 Gestion du groupe des contrôleurs locaux

Les contrôleurs locaux et la source du groupe appartiennent au groupe multicast GCL. Ce groupe est utilisé lors de la distribution de la clé de chiffrement de données TEK, à tous les membres du groupe. Une clé KEK_CCL est détenue par tous les contrôleurs locaux du groupe et sert à sécuriser la distribution de la clé de chiffrement de données TEK.

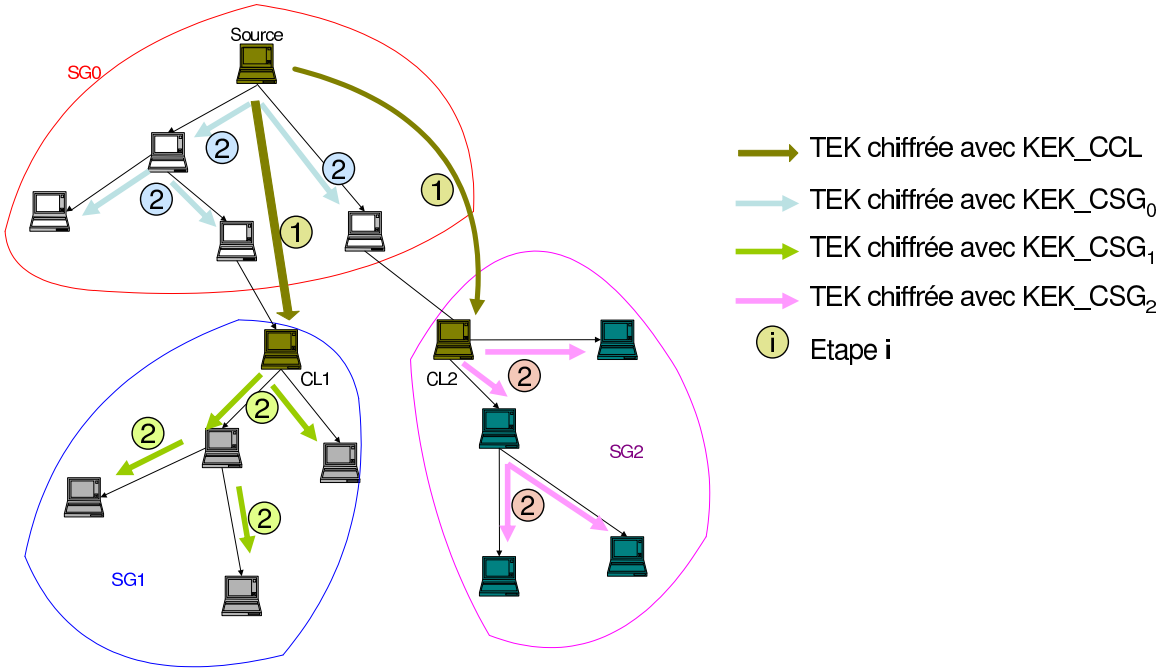


FIG. 4.3 – Distribution de la TEK

La gestion de cette clé est centralisée autour de la source courante du groupe et est effectuée de la façon suivante :

- Tout nouveau contrôleur local joint le groupe GCL et reçoit de la part de son ancien CL la clé KEK_CCL, chiffrée avec sa clé publique.
- Dans le cas où la source courante du groupe ne joue pas le rôle de contrôleur local, elle doit joindre le groupe GCL avant d'entamer la procédure de distribution de la clé de chiffrement de données. Ensuite, en recevant la KEK_CCL de la part de son ancien CL, elle envoie la nouvelle clé de chiffrement de données au groupe GCL, chiffrée avec cette clé KEK_CCL.
- La clé KEK_CCL est renouvelée dès qu'un contrôleur local quitte le GCL pour passer à l'état simple membre du groupe, ou quitte tout le groupe multicast. Ce renouvellement est effectué afin de garantir le secret futur des clés de chiffrement de données. Il est à la charge de la source courante du groupe, qui distribue la nouvelle KEK_CCL aux contrôleurs locaux restants, chiffrées avec leurs clés publiques respectives.
  La KEK_CCL est également renouvelée lors d'un ajout d'un contrôleur local au groupe GCL, afin d'assurer le secret passé des clés de chiffrement de données antérieures et par conséquent du flux de données correspondant. La source courante envoie la nouvelle KEK_CCL aux anciens contrôleurs locaux chiffrée avec l'ancienne KEK_CCL ; elle l'envoie également au contrôleur local joignant le GCL chiffrée avec sa clé publique.
- La source courante du groupe quitte le GCL dès qu'elle finit d'émettre son flux multicast, dans le cas où elle ne joue pas le rôle de contrôleur. La clé KEK_CCL est alors renouvelée comme cela est présenté ci-dessus.

4.3.5 Gestion de la mobilité

À chaque événement de Join ou de Leave dans le groupe, le membre parent (dans l'arbre de distribution de clés), décide via sa fonction d'évaluation de former avec ses membres locaux un cluster, ou de fusionner avec le cluster de son CL parent.

Quand un membre se déplace dans le réseau, il peut perdre sa connectivité avec son CL et ainsi ne plus être atteignable par les flux de distribution de clés. Pour résoudre ce problème, tous les CLs envoient périodiquement des messages "*CL_Queries*", contenant leurs identités. Ainsi, un membre se déplaçant dans le réseau et recevant des messages *CL_Queries*, choisira d'appartenir au cluster géré par le CL le plus proche, par rapport au nombre de sauts. La figure 4.4 illustre ce scénario de mobilité.
Ensuite, pour sa ré-authentification, le membre utilisera le ticket de ré-intégration présenté dans la section 4.4.3.

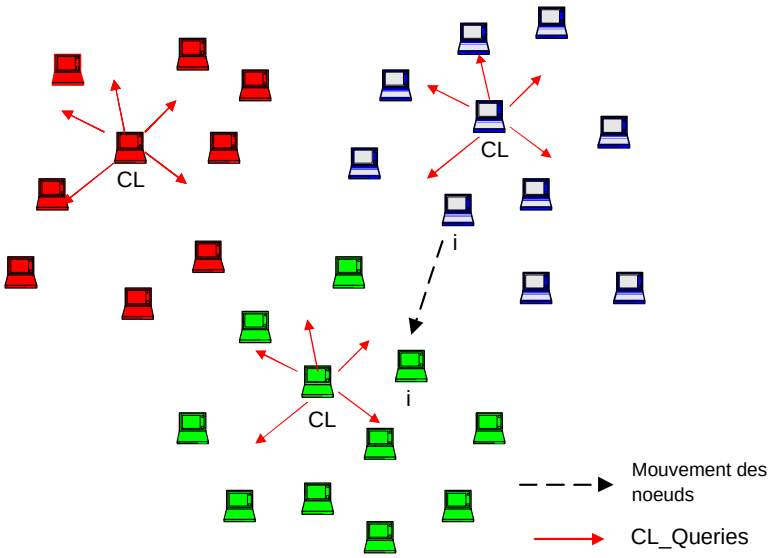


FIG. 4.4 – Mobilité des nœuds dans BALADE

La diffusion des CL_Queries se fait à deux sauts et ce afin de permettre à tous les membres du groupe multicast de connaître tous les contrôleurs locaux à deux sauts de leur localisation géographique, et ainsi de choisir de rejoindre le cluster dont le CL est le plus proche d'eux en cas de déplacement ou de perte de connectivité. En effet, tout membre du groupe a seulement besoin d'avoir une connaissance restreinte de la localisation des contrôleurs locaux, qu'il pourrait éventuellement joindre (en cas de perte de connectivité avec son contrôleur courant).

Le cas d'utilisation présenté dans la figure 4.5 illustre l'envoi des CL_Queries avec un TTL=2, suffisant pour assurer un bon fonctionnement de BALADE sans avoir recours à un broadcast coûteux vers tous les membres du réseau. Dans la figure 4.5.a, le membre du groupe mg_k connaît son contrôleur local (à un saut) et les autres contrôleurs locaux à deux sauts. Au cours de son déplacement (figure 4.5.b), ce membre perd sa connectivité avec son contrôleur local et va essayer de joindre un autre contrôleur local le plus proche de sa localisation géographique. Dans la figure 4.5.c, mg_k connaît son nouveau contrôleur local et les nouveaux contrôleurs locaux à deux sauts et perd la connaissance des contrôleurs locaux qui ne sont plus à deux sauts de lui.

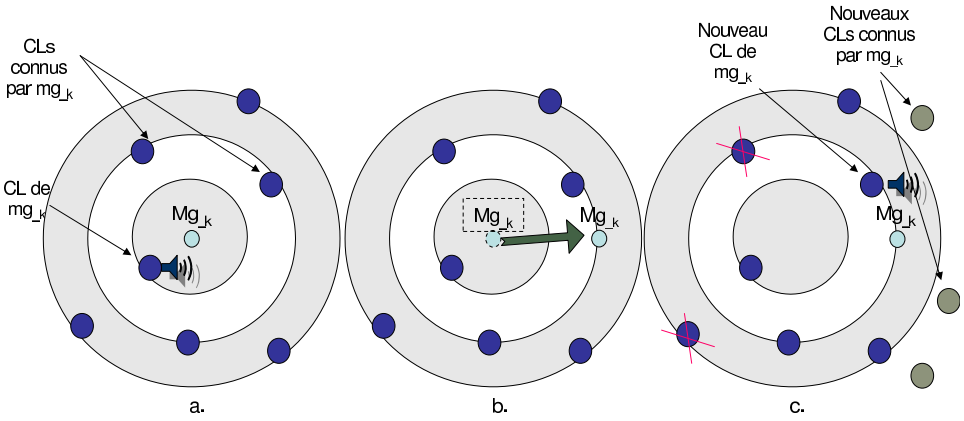


FIG. 4.5 – Envoi périodique des CL_Queries à 2 sauts

Quand un contrôleur local se déplace dans le réseau, quitte le groupe multicast ou disparaît à cause d'un problème de ressources, il envoie auparavant (si c'est possible) un message de notification à ses membres locaux, leur demandant de joindre d'autres clusters du groupe.

4.3.6 Fiabilité de la distribution de clés dans BALADE

Pour améliorer la fiabilité de la transmission de la TEK et assurer la synchronisation entre la distribution de la clé de chiffrement de données et la diffusion des données chiffrées, nous utilisons les numéros de séquence. Un numéro de séquence correspond à chaque TEK. Ce même numéro de séquence est ajouté également au flux multicast chiffré avec cette TEK.

La fiabilité de la gestion des clés dans BALADE, importante dans des milieux hostiles comme les réseaux ad hoc, est ainsi assurée à deux niveaux :

1. Quand un membre du groupe perd sa connectivité avec son cluster, à cause de son déplacement ou en raison d'un problème de ressources, il entame une procédure de ré-intégration, à condition qu'il détienne la clé de chiffrement de données TEK courante. Cette procédure de ré-intégration lui permettra d'obtenir la clé locale de son nouveau sous-groupe KEK_CSG_{ik}.
2. Quand un membre du groupe se rend compte qu'il ne détient pas la clé de chiffrement de données appropriée, (le numéro de séquence de la TEK ne correspond pas à celui des données chiffrées), il demande à son contrôleur local de lui re-transmettre la TEK courante, à condition qu'il détienne la clé locale du cluster auquel il appartient KEK_CSGik.

4.4 Le protocole BALADE

Dans cette section, nous présentons les différentes phases du protocole BALADE : initialisation de BALADE, ajout d'une nouvelle entité dans le groupe, authentification et contrôle d'accès, départ d'un membre du groupe et renouvellement des clés du groupe. La spécification complète des différents sous-protocoles de BALADE, en vue de leur validation et implantation, est présentée dans le chapitre 6.

4.4.1 Initialisation de BALADE

À l'initialisation de l'application, l'authentification et le contrôle d'accès sont réalisés par la première source du groupe. Ensuite, dynamiquement, la source délègue cette tâche aux CLs qui doivent être capables d'autoriser ou de refuser un *Join* d'un nouveau membre au groupe.

L'initialisation de la liste de contrôle d'accès se fait avant le lancement de BALADE (hors ligne) et contient tous les CBIDs des nœuds pouvant joindre le groupe dès son initialisation ou ultérieurement.

L'établissement et la maintenance de la liste des sources du groupe est à la charge d'une application[13] externe qui s'exécute à l'initialisation de l'application en question.

Le contrôleur global CG, qui initialement est la première source du groupe, initialise les deux listes LPL (*Local Participants List*) et RL (*Revocation List*). LPL contiendra tous les membres du groupe appartenant à l'arbre multicast, ayant joint le groupe hors ligne. La liste de révocation est initialement vide. Le CG entame ensuite la phase de distribution de la clé du groupe à tous les membres n_i de la liste LPL (message **1**), il procède alors comme suit :

(**1**) $CG \longrightarrow n_i : \{TEK, KEK\_CSG_0, IDG, ID_{CG}, CBID_{CG}\}$‘$Pub\_n_i$
avec TEK : *Clé de chiffrement de données*, KEK_CSG_0 : *clé du sous − groupe* 0 *gérée par la source*, IDG : *identité du groupe*, ID_{CG} : *identité du CG*, $CBID_{CG}$: *CBID du CG et* Pub_n_i : *clé publique de* n_i

---

4.4.2 Ajout d'une nouvelle entité

Tout nœud n_i doit s'authentifier auprès de la source ou d'un contrôleur local. Si l'authentification et le contrôle d'accès réussissent, il pourra joindre le groupe. L'authentification et le contrôle d'accès sont présentés dans la section 4.4.3.

Chaque nœud n_i génère et détient une paire de clés publique et privée (Pub_n_i, Pri_n_i). À partir de ces clés, chaque nœud calcule son identificateur cryptographique unique (CBID) qui lui sera nécessaire pour s'authentifier auprès de la source.

La source et tous les contrôleurs locaux (CLs) détiennent une ACL (Access Control List) contenant tous les CBIDs des nœuds autorisés à joindre le groupe. Ils détiennent aussi une liste de révocation contenant tous les CBIDs des nœuds exclus du groupe et ne pouvant plus le joindre. Un CL autorise ou non un nœud à joindre le groupe selon ces deux listes.

À ce stade et en cas de succès de l'authentification et du contrôle d'accès du nouveau membre, le membre parent calcule sa fonction d'évaluation. Deux cas se présentent :

- si $f_i = true$, le membre passe à l'état CL. Il doit donc générer une nouvelle clé et la distribuer à ses membres locaux. Pour cela, il envoie la nouvelle clé locale KEK_CSG_i en multicast à tous les anciens membres locaux, cryptée avec l'ancienne clé old$_KEK_CSG_i$ (message **2**) et envoie la même clé KEK_CSG_i au nouvel abonné, cryptée avec sa clé publique (message **3**).

  En plus, si le membre vient de passer à l'état CL et donc son ancienne clé locale est égale à la clé de son nœud parent CL_{parent}, il doit envoyer son ancienne clé locale à son

[13] par exemple une application "peer to peer" qui se charge de créer la liste des sources du groupe et de la diffuser à tous les membres du groupe

CL (message **4**) pour que ce nœud change sa clé locale pour tous ses membres locaux (c'est à dire au cas où old$_KEK_CSG_i = KEK_CSG_{parent}$).

Il est à noter qu'un membre ne peut passer à l'état CL que s'il reçoit l'autorisation de son contrôleur local pour le faire. Ce dernier doit l'authentifier et vérifier qu'il est bien un membre du groupe. Au cas où l'authentification réussit, le contrôleur actif lui envoie la clé du groupe des contrôleurs locaux (KEK_CCL), cryptée avec sa clé publique. Le CL procèdera donc de la façon suivante :

(**2**) $for\ j : 1..nb\_anciens\_membres$
$CL \longrightarrow n_j : \{KEK\_CSG_i\}`old\_KEK\_CSG_i$

(**3**) $CL \longrightarrow n_i : \{KEK\_CSG_i\}`Pub\_n_i\ avec\ n_i :\ nouvel\ abonné$

(**4**) $CL \longrightarrow CL_{parent} : \{old\_KEK\_CSG_i\}`Pub\_CL_{parent}$
$avec\ Pub\_CL_{parent} :\ clé\ publique\ du\ noeud\ parent\ CL_{parent}.$

- si $f_i = false$, le membre parent reste à l'état passif, il envoie donc une demande de renouvellement de clé à son CL, qui entame la génération d'une nouvelle clé et sa distribution à tous ses membres locaux.

4.4.3 Authentification et contrôle d'accès

Pour l'authentification et le contrôle d'accès, nous distinguons deux cas : authentifier et contrôler l'accès d'un nouveau membre qui est à la charge du contrôleur local du sous-groupe en question et authentifier et contrôler l'accès d'un membre du groupe qui a perdu sa connectivité avec l'arbre multicast à cause de sa mobilité ou d'un problème de ressources.

Authentification et contrôle d'accès d'un nouveau membre du groupe

La figure 4.6 illustre l'authentification d'un nouveau membre au groupe.

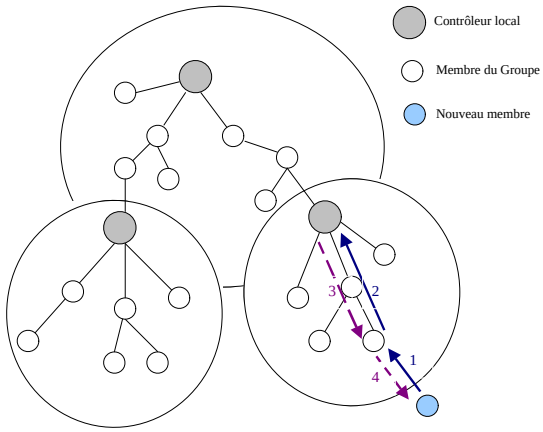


FIG. 4.6 – Authentification et contrôle d'accès d'un nouveau membre au groupe

Le nouveau nœud demande à un membre du groupe de joindre le groupe multicast. Pour cela, il lui envoie un message de demande d'adhésion au groupe, contenant son CBID. Le membre du groupe ne pouvant pas contrôler son accès au groupe, envoie un message de vérification de contrôle d'accès à son contrôleur local. Le contrôleur local qui détient la liste ACL, peut accepter ou non la demande d'adhésion du nouveau nœud. Au cas où l'authentification et le contrôle d'accès réussissent, le contrôleur local renvoie un message d'acceptation d'adhésion au groupe au membre parent, qui se charge d'activer la route de l'arbre multicast vers le nouveau nœud. Le membre parent envoie en plus, un message d'acceptation d'adhésion au nouveau nœud, contenant un mot de passe chiffré avec la clé de chiffrement de données TEK. Ce mot de passe que nous appelons ticket, sera utilisé lors de la ré-authentification du nouveau membre (cf. section suivante).

Authentification et contrôle d'accès d'un membre du groupe

La figure 4.7 illustre l'authentification d'un ancien membre au groupe.

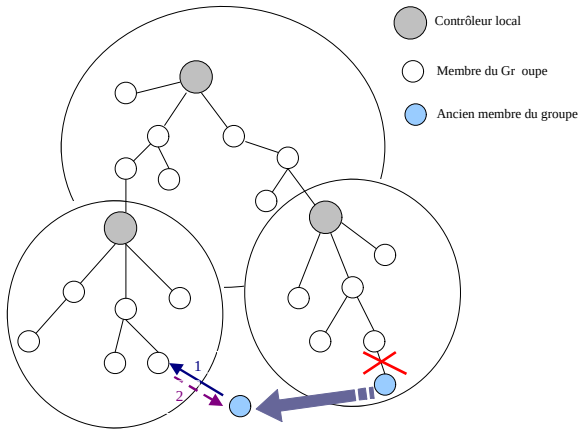


FIG. 4.7 – Authentification et contrôle d'accès d'un ancien membre du groupe

Ce cas d'utilisation est très probable dans le cadre des réseaux ad hoc et correspond à un nœud qui se déplace d'un sous-groupe à un autre, ou bien qui s'éteint brusquement à cause d'un problème de batterie. Le nœud en question demande à un membre du groupe de l'authentifier et de contrôler son accès. Pour cela, il lui envoie un message de demande d'adhésion au groupe contenant son ticket qu'il avait reçu lors de sa première authentification.

Le membre de l'arbre pourra ainsi déchiffrer le ticket (mot de passe) avec la clé TEK, vérifier si le nœud était bien un membre du groupe et en cas de succès accepter la demande d'adhésion du nouveau nœud.

Le ticket est commun à tous les membres du groupe et est chiffré avec la clé de chiffrement de données. Ainsi, ce ticket n'est plus valable après un renouvellement de la TEK. Ceci est nécessaire et efficace dans le cas d'une révocation d'un membre du groupe qui ne doit pas être capable de joindre le groupe de nouveau, depuis n'importe quel cluster.

Pour évaluer le gain en nombre de messages envoyés, engendré par l'authentification d'un ancien membre du groupe par rapport à celle d'un nouveau membre, on considère :

- n : nombre de membres du groupe.
- c : nombre moyen de membres par sous-groupe.

Le nombre de messages nécessaires pour l'authentification d'un nouveau membre est au meilleur cas 2 messages et au pire $2*c$ messages ; soit en moyenne $c+1$ messages nécessaires. Cependant, le nombre de messages nécessaires pour l'authentification d'un ancien membre du groupe est toujours de 2 messages.

4.4.4 Retrait et expulsion d'une entité du groupe

Nous distinguons deux cas : retrait volontaire et expulsion. Le premier cas se produit quand un membre veut quitter le groupe et envoie un message *Leave* incluant son CBID, à son contrôleur local dans le but de stopper le flux de trafic multicast du groupe. Dans ce cas, le CL supprime ce membre de sa liste de membres locaux LPL et entame une phase de renouvellement de clé.

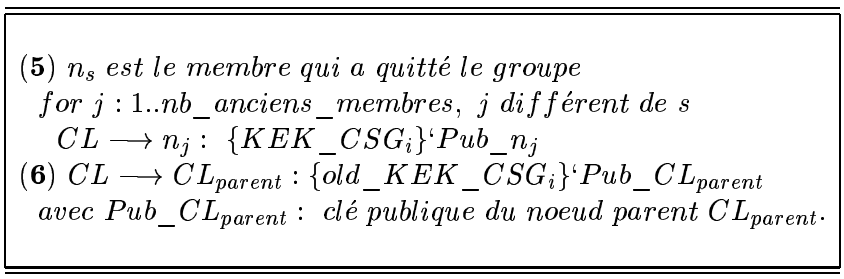
Le deuxième cas (expulsion) est connu sous le nom de révocation du membre et a lieu quand un membre peut mettre la sécurité du groupe en péril. Dans ce cas, le contrôleur local ajoute le nom de ce membre à la liste distribuée de révocation RL. Le membre parent du nœud quittant le groupe le supprime de sa liste de membres locaux LPL. Ensuite il entame une phase de renouvellement de clé.

La phase de renouvellement de clé commence par l'évaluation de la fonction d'état, afin de décider du nouvel état que doit adopter le membre parent du nœud quittant le groupe. Comme pour l'ajout d'une entité au groupe, deux cas se présentent :

- si $f_i = true$, le membre passe à l'état CL. Il doit donc générer une nouvelle clé et la distribuer à ses membres locaux.

  Pour cela, il envoie la nouvelle clé locale KEK_CSG_i en unicast à tous les anciens membres locaux, en excluant le membre qui a quitté le groupe, cryptée avec leurs clés publiques respectives (message **5**).

  En plus, dans le cas où le membre vient de passer à l'état CL et donc sa clé locale ancienne est égale à la clé de son nœud parent CL_{parent}, il doit envoyer sa clé locale ancienne à son CL (message **6**) pour que ce nœud change sa clé locale pour tous ses membres locaux (c'est à dire au cas où old$_KEK_CSG_i = KEK_CSG_{parent}$).



- si $f_i = false$, le membre parent reste à l'état passif, il envoie donc une demande de renouvellement de clé à son CL, qui entame la génération d'une nouvelle clé et sa distribution à tous ses fils, en excluant le membre qui a quitté le groupe.

Le cas de révocation d'un membre du groupe nécessite, en plus du renouvellement de la clé du sous-groupe, le renouvellement de la clé de chiffrement de données TEK. Pour cela, le contrôleur local du sous-groupe en question envoie un message à la source lui demandant de changer la TEK et de la distribuer à tous les membres du groupe.

4.4.5 Renouvellement des clés

Pour assurer la confidentialité des données, un processus de changement de la TEK doit être lancé côté source, selon plusieurs méthodes :

- La source change la TEK à chaque bloc fixe de données :
  Si le bloc est assez petit, le secret futur est assuré dès le *Leave* d'un nœud, à un coût très contraignant côté source. Par contre, si le bloc de données est grand, le surcoût est nettement moins contraignant avec l'inconvénient qu'un nœud peut continuer à déchiffrer le bloc de données courant pendant une grande période de temps.
- La source change la TEK à chaque unité sémantique de données dépendant de l'application en question :
  Cette solution est beaucoup plus adaptée au type de l'application en question. Ainsi, une source qui diffuse un flux MP3 va changer la TEK à chaque titre MP3, alors qu'une source qui diffuse un flux vidéo va renouveler sa TEK à chaque film ou à chaque chapitre d'un film. Cette solution est plus réaliste du fait qu'elle tient compte du mode opératoire de l'application. C'est pour cette raison que nous l'avons adoptée dans BALADE.

Le renouvellement de la TEK est effectué de la façon suivante :

(**7**) *La source* (CG) *génère une nouvelle* TEK *et la distribue à tous les membres locaux de son sous − groupe, chiffrée avec la clé locale du sous − groupe* :
$\forall n_i \in LPL\_CG$:
$CG \longrightarrow n_i : \{TEK\}`KEK\_CSG_{CG}$

(**8**) *La source* (CG) *envoie la nouvelle* TEK *au groupe de* CLs*, chiffrée avec la* KEK_CCL :
$\forall CL_i \in GCL$:
$CG \longrightarrow CL_i : \{TEK\}`KEK\_CCL$

(**9**) *Tout* CL_i *envoie la nouvelle* TEK *à ses membres locaux, chiffrée avec la clé* KEK_CSG_i :
$\forall CL_i \in GCL, \forall n_j \in LPL\_CL_i$:
$CL_i \longrightarrow nj : \{TEK\}`KEK\_CSG_{CL_i}$

En raison de la durée de vie limitée d'une clé de chiffrement symétrique, les clés des sous-groupes servant à chiffrer la TEK doivent également être changées périodiquement, à la charge des contrôleurs locaux des sous-groupes.

4.5 Conclusion

Dans ce chapitre, nous avons présenté BALADE, un protocole de gestion de clé, pour les réseaux ad hoc, permettant de sécuriser les communications de diffusion de groupe, à large échelle. Le modèle de diffusion des services visés est multi-sources séquentielles (1 vers n séquentiel). Selon ce modèle, à tout instant t, il y a une et une seule source qui émet et une fois qu'elle termine, une autre source prend le relais.

L'idée fondatrice de BALADE est de subdiviser le groupe multicast dynamiquement. Chaque sous-groupe est géré et supervisé par un contrôleur local qui partage avec ses membres

locaux une clé de sous-groupe. Le flux multicast est chiffré par la source avec la clé TEK et envoyé en multicast à tous les membres du groupe. La source envoie la clé TEK chiffrée à tous les contrôleurs locaux, chiffrée avec une clé KEK. Ces contrôleurs locaux transmettent alors la TEK à leurs membres locaux, chiffrée avec leurs clés de sous-groupes respectifs. L'avantage de BALADE est que seule la clé TEK est chiffrée et déchiffrée et non plus les données du flux multicast. La clé TEK est renouvelée à chaque unité sémantique des données émises par la source. Ce renouvellement est orienté sources du groupe et tient compte de la nature de données à sécuriser et de la politique de sécurité instaurée au sein de l'application en question.

La clusterisation dans BALADE est réalisée via l'exécution d'une fonction d'évaluation, au sein de chaque membre du groupe, qui prend en compte le nombre de membres et la fréquence d'adhésion locaux. Cependant, la localisation géographique des nœuds n'est pas prise en compte. De plus, la consommation de l'énergie et de la bande passante n'est pas considérée dans le processus de distribution de clés dans BALADE. Pour combler ces lacunes, nous avons conçu un algorithme de clusterisation, appelé OMCT, tenant compte de la localisation et de la mobilité des nœuds, tout en optimisant l'énergie et la bande passante consommées. Cet algorithme, présenté dans le chapitre 5, sera ensuite intégré dans BALADE. La spécification du protocole BALADE couplé avec OMCT est présentée dans le chapitre 6.

Une comparaison du protocole BALADE avec Enhanced BAAL est présentée dans le tableau 1 de la conclusion générale, en fonction des métriques d'évaluation définies dans la section 2.4.

Chapitre 5

OMCT : Algorithme de clusterisation dynamique

Sommaire

5.1 Introduction

Blake et al. ont montré dans [LBC+01] que toute architecture de communications dans un réseau ad hoc se basant sur une topologie "plate" (communications multi-sauts sans clusterisation) entraînera une dégradation significative des performances de ce réseau, voire même un échec de communications au sein d'un MANET à large échelle. Les réseaux ad hoc à architecture plate rencontrent un problème de passage à l'échelle, en raison de leurs limites intrinsèques. En effet, sous des modèles de trafic uniformes, la bande passante disponible à tout nœud du réseau s'approche de zéro quand la taille du réseau augmente. Une raison fondamentale de cette dégradation, démontrée dans [LBC+01], est que les communications ne sont pas locales dans le réseau ad hoc. Les flux de paquets parcourent donc de longues distances et se disputent le médium de communication sans fil avec les autres flux du réseau. En d'autres termes, moins le modèle de communication dans un MANET est local, plus rapidement la capacité des nœuds ad hoc se dégradent quand la taille du réseau augmente.

La solution la plus appropriée pour remédier à ce problème et garantir de meilleures performances dans un MANET à large échelle est l'établissement d'une architecture hiérarchique du réseau (structure de clusterisation logique). Cette contrainte doit ainsi être instaurée dans toute architecture de sécurité établie dans un MANET, telle que le déploiement d'un protocole de gestion de clé de groupe multicast. C'est dans ce cadre que nous avons défini l'algorithme OMCT (*Optimized Multicast Clustering Tree*), de clusterisation dynamique de groupe multicast, tenant compte de la localisation et de la mobilité des nœuds et optimisant la consommation de l'énergie et de la bande passante.

Le chapitre est structuré de la façon suivante. La section 5.2 présente les motivations de la conception d'un algorithme de clusterisation dynamique au sein d'une architecture de gestion de clé de groupe dans les MANETs. Dans la section 5.3, nous décrivons notre algorithme OMCT et nous le validons à travers des analyses et des discussions. L'intégration de OMCT dans notre protocole de gestion de clé de groupe BALADE est présentée dans la section 5.4. Nous proposons dans la section 5.5 le couplage de la technique de relais multipoints utilisée dans OLSR avec l'algorithme OMCT et nous étudions l'apport d'un tel couplage. Finalement, la section 5.6 conclut ce chapitre.

5.2 Motivations

OMCT [BCF05] est un algorithme de clusterisation dynamique orienté distribution de clés de groupe multicast pour les réseaux ad hoc. Exécuté périodiquement par la source du groupe, OMCT divise dynamiquement le groupe multicast en des clusters fortement corrélés, géré chacun par un contrôleur local, noté CL. La forte corrélation géographique au sein d'un cluster est définie par le fait que tous les membres d'un cluster sont à la portée (en un seul saut) de leur contrôleur local. Dynamiquement et périodiquement, chaque contrôleur de cluster créé, exécutera à son tour l'algorithme OMCT. Les objectifs principaux de cet algorithme sont :

- d'exploiter l'avantage des communications en "broadcast" dans un environnement sans fil (diffusion) ;
- d'optimiser le temps de transmission des clés. En effet, un cluster fortement corrélé minimise le nombre de relais responsables de l'acheminement des messages entre source et récepteurs et par conséquent, minimise le temps de transmission des données ;
- d'optimiser la consommation de la bande passante. Cet avantage est déduit de la mini-

misation du nombre de retransmissions multi-sauts au sein d'un cluster corrélé ;
- d'optimiser la consommation de l'énergie. En effet, l'énergie requise pour la transmission d'un message à tous les membres d'un cluster est égale à la somme des énergies utilisées par chaque relais pour acheminer le message à ses membres fils. La minimisation de ces relais diminuera ainsi la consommation d'énergie.
  Dans la figure 5.1, est illustré un exemple de calcul de la consommation de l'énergie, nécessaire pour la diffusion d'un message à tous les membres du réseau (le nœud 1 étant la source du message).

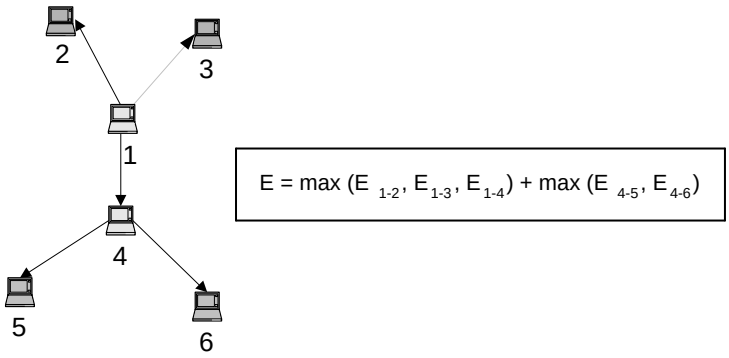


FIG. 5.1 – Exemple de calcul de l'énergie consommée

OMCT prend en compte la mobilité et la localisation des nœuds. En effet, il utilise les informations de localisation géographique de tous les membres du groupe, dans la construction de l'arbre de distribution de clés. La présence d'un système GPS *Global Positioning System*, ou de tout autre support à la localisation, au sein d'un réseau ad hoc, est supposée établie.

5.3 OMCT : algorithme de clusterisation dynamique

Les paramètres du réseau ad hoc sont les suivants :

1. Les nœuds sont représentés par des points dans un espace euclidien à 2 dimensions ; la distance entre deux nœuds $i(x_i, y_i)$ et $j(x_j, y_j)$ est calculée de la façon suivante :

$$d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} = \| i - j \|$$

2. L'énergie requise pour envoyer un message d'un nœud i à un nœud j est proportionnelle à la distance $d(i,j)$. Elle est calculée comme suit [SK98] :

$$P = (d_{ij})^{\gamma}$$

   γ est le facteur de perte de propagation dans le réseau ad hoc et prend généralement sa valeur entre 2 et 4, dépendant des caractéristiques du support de communication.

3. Étant donné un cluster à c membres locaux, la matrice de distance entre ses membres notée D est une matrice diagonale supérieure pour des raisons d'optimisation de stockage et parce que $d(i,j) = d(j,i),\ \forall i, j <= c$.

$$D = \begin{pmatrix} 0 & d_{12} & \cdots & \cdots & d_{1c} \\ 0 & \ddots & & & \vdots \\ \vdots & & d_{ii} = 0 & & \vdots \\ \vdots & & & \ddots & d_{c-1,c} \\ 0 & \cdots & \cdots & \cdots & 0 \end{pmatrix}$$

5.3.1 Description de l'algorithme OMCT

Il est connu que la détermination d'un arbre multicast à coût minimal dans les réseaux filaires est un problème difficile, considéré comme le problème NP-complet de l'arbre de Steiner [WNE00]. De plus, CHELIUS démontre dans sa thèse [Che04] que le problème de détermination d'un arbre multicast avec un nombre limité d'émetteurs dans un réseau ad hoc est un problème NP-complet (preuve par réduction du problème de recherche des relais multipoints [CJA+03]).

Notre problème de clusterisation du groupe multicast et de l'élection des contrôleurs locaux est au moins aussi difficile que celui de la détermination de l'arbre multicast dans les réseaux filaires. Il peut également être considéré comme un cas particulier du problème de détermination d'un arbre multicast avec un nombre limité d'émetteurs dans un réseau ad hoc. Il est donc NP-complet. Une heuristique est ainsi nécessaire pour pouvoir rapprocher la solution optimale à notre problème. Nous présentons dans ce qui suit notre heuristique, sur laquelle est basé l'algorithme OMCT.

Pendant l'initialisation du cluster, tous les membres sont attachés à un contrôleur local qui est responsable de la génération de la clé dans son sous-groupe et de sa distribution à tous ses membres locaux. En outre, le contrôleur vérifie périodiquement si son cluster est fortement corrélé et par conséquent si le processus de distribution de clés est optimal. L'évaluation de la cohésion du cluster est déterminée en calculant le facteur de centralisation du cluster autour du nœud central :

$$Cohésion = \frac{membres\_dans\_la\_portée\_du\_contrôleur}{nombre\_de\_membres\_du\_cluster}$$

Ce paramètre mesure la proximité des membres du cluster par rapport à leur contrôleur. Plus le nombre de membres locaux atteignables par le contrôleur en un seul saut est grand, plus le facteur de cohésion tend vers 1. Ainsi, nous pouvons vérifier avec ce paramètre si le cluster est fortement corrélé ou non.

Nous avons défini $Min\_Cohésion$ comme le seuil minimal en dessous duquel la cohésion est considérée perdue. Dans ce cas, le contrôleur doit prendre la décision de diviser le cluster et d'élire de nouveaux contrôleurs locaux selon leur localisation géographique. Le processus de clusterisation selon OMCT optimise le nombre de nouveaux contrôleurs, tout en atteignant tous les membres du cluster.

Les principales étapes de l'algorithme OMCT sont les suivantes (cf. Algorithme 2) :

- Étape 1 : La liste des c membres du cluster est initialisée. La liste des contrôleurs locaux est initialement vide et contiendra les contrôleurs locaux des nouveaux clusters créés.
- Étape 2 : La liste des membres est triée dans l'ordre croissant, selon leur distance par rapport au contrôleur local courant (initialement le nœud 1).

Algorithm 2 OMCT : Algorithme de clusterisation dynamique de groupe multicast

Liste_CLs = ϕ **(Étape 1)**
Liste_nœuds = {1, 2, 3, ..., c}
i = 1 ;
j = 2 ;

while (Liste_nœuds $\neq \phi$) **do**
Trier_Liste_nœuds(i) ; // Trier la liste de membres selon leurs distances par rapport au nœud i **(Étape 2)**
while $d_{(i,Liste\_noeuds(j)} \leq Max\_Distance$) **do**
// Max_Distance correspond à la portée du nœud i
$Liste\_noeuds = Liste\_noeuds/\{Liste\_noeuds(j)\}$; // Supprimer Liste_nœuds(j) de la liste de membres
j++ ;
end while
//Le nœud List_nœuds(j+1) ne peut pas être joignable directement par le nœud i **(Étape 3)**
s=1 ;
for ($l = 2$ to j) **do**
if ($d_{(s,Liste\_noeuds(j+1)} > d_{(l,Liste\_noeuds(j+1)}$) **then**
s=l ;
end if
end for
CL=s ;
// Le CL élu est le plus proche du nœud Liste_nœuds(j+1) **(Étape 4)**
i=CL ;
$List\_CLs = List\_CLs \cup \{CL\}$; // Ajouter le CL à la liste des contrôleurs locaux
end while(Étape 5)

- Étape 3 : La liste des membres est parcourue jusqu'au nœud j, qui ne peut pas être directement atteint par le contrôleur local parent. Les $j-1$ membres joignables en un seul saut, à partir du contrôleur courant, seront supprimés de la liste des membres du groupe.
- Étape 4 : Le nœud le plus proche du nœud j, dans la portée du contrôleur local courant, sera élu comme nouveau contrôleur local. Il est donc ajouté à la liste des contrôleurs locaux.
- Étape 5 : Les étapes 2, 3 et 4 sont réitérées jusqu'à traiter tous les membres du cluster. Ainsi, tous les membres sont atteignables par les contrôleurs locaux et peuvent commencer à recevoir le flux de données envoyé par la source.

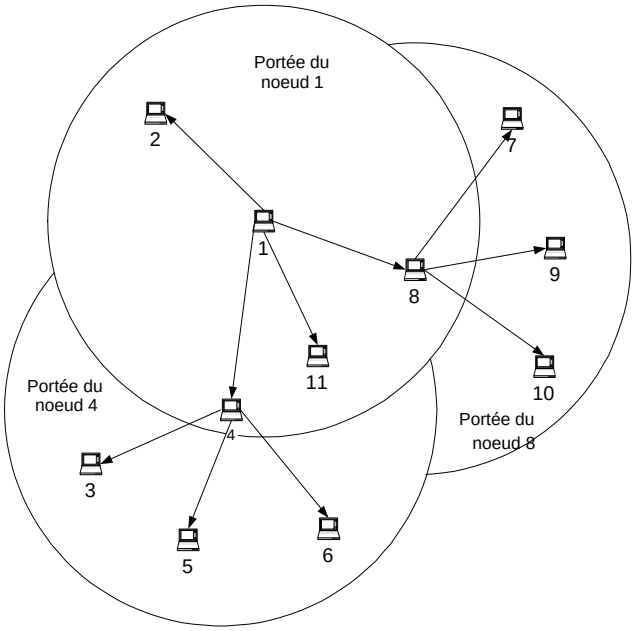


FIG. 5.2 – Exemple d'exécution de l'algorithme OMCT

Lorsque les premiers clusters sont créés au sein du groupe multicast, les nouveaux contrôleurs locaux deviennent responsables de la gestion de la clé locale, de sa distribution à tous les membres locaux et également de la maintenance de la propriété de forte corrélation géographique du cluster.

Ainsi, récursivement et à tout moment de la session du groupe multicast, le groupe est composé de clusters fortement corrélés, assurant que leur paramètre de cohésion est toujours plus grand que le seuil minimal défini $Min\_Cohésion$. La figure 5.2 comporte un exemple d'exécution de OMCT. Les nouveaux contrôleurs locaux sont les nœuds 1, 4 et 8.

Amélioration de la première version de OMCT

OMCT divise le groupe multicast en tenant compte de la localisation et de la mobilité des membres du groupe. Cependant, le nombre de membres au sein de chaque cluster n'est pas considéré. La figure 5.3 comporte un exemple d'exécution d'OMCT, où le nœud 2 forme un cluster avec un seul membre local (nœud 3). Alors que le contrôleur local 4 peut atteindre le

nœud 3, sans avoir besoin de former un nouveau cluster. Une amélioration de OMCT est donc indispensable afin d'optimiser le nombre de contrôleurs locaux et par conséquence la consommation de l'énergie. Cette amélioration consiste à prendre en compte le nombre de membres dans chaque cluster, devant varier entre deux seuils prédéfinis $MIN\_THRESHOLD$ et $MAX\_THRESHOLD$ (cf. Annexe A - Algorithme 3).

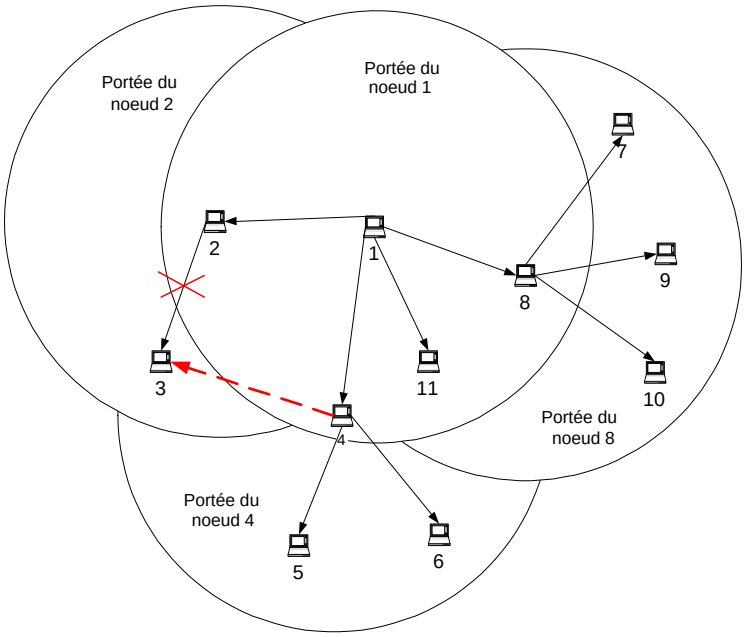


FIG. 5.3 – Exemple d'exécution de la version améliorée de OMCT

Les améliorations de OMCT consistent à modifier l'étape 3 et ajouter une étape 6, comme suit :

- Étape 3' : La liste des membres est parcourue jusqu'au nœud j qui ne peut pas être directement atteint par le contrôleur local courant. Les nœuds à la portée du contrôleur sont supprimés de la liste des membres et ajoutés à la liste de membres locaux de ce contrôleur, tout en respectant la contrainte du seuil maximum de nombre de membres locaux $MAX\_THRESHOLD$. À la fin de cette étape, le nœud j ne peut pas être joint directement par le contrôleur ou le cluster géré par ce contrôleur est saturé.
- Étape 6 : La liste des clusters formés est analysée. Si un cluster a un nombre de membres locaux inférieur au seuil minimal $MIN\_THRESHOLD$, alors la liste des membre locaux de ce cluster est à nouveau parcourue pour essayer de les affecter à d'autres clusters, quand c'est possible.

5.3.2 Analyses et discussions

Dans cette section, nous présentons les analyses réalisées afin de valider le paramètre de cohésion de l'algorithme OMCT et les apports de OMCT par rapport aux travaux existants.

Validation du facteur de cohésion de OMCT

L'étude analytique que nous avons menée consiste à montrer que le paramètre de cohésion choisi apporte une évaluation juste de l'état de corrélation des membres d'un cluster et reflète d'autres métriques du cluster comme :

- le nombre de nouveaux contrôleurs CLs correspondant au nombre de clusters créés, après exécution de OMCT.
- l'énergie requise pour envoyer un message en multicast, de la source à tous les membres du groupe, passant par les nouveaux CLs créés par OMCT. Seule l'énergie requise pour l'envoi des messages est calculée, l'énergie utilisée pour la réception des messages et le traitement des signaux étant négligée.
- Nous utilisons une métrique présentée dans [WNE01], appelée la métrique *yardstick*. Plus l'approche de clusterisation est optimale, plus la métrique yardstick est grande. Cette métrique est calculée de la façon suivante :

$$yardstick = \left(\frac{m}{p}\right)\left(\frac{m}{n}\right) = \left(\frac{m^2}{n*p}\right)$$

  Avec :
  - n est nombre de membres destinataires (nombre de membres du groupe),
  - m est le nombre de membres de groupe, atteignables par le flux multicast émis par la source,
  - p est la somme des énergies nécessaires pour envoyer un message en multicast,
  - (m/n) représente le facteur d'atteignabilité,
  - (m/p) représente le facteur de consommation d'énergie, par rapport au nombre de membres atteignables.

Nous avons généré aléatoirement des topologies de réseaux ad hoc. Les localisations des nœuds sont également distribuées aléatoirement sur toute la surface des réseaux. L'algorithme OMCT est exécuté 10 fois, pour chaque configuration. Une entité est choisie aléatoirement pour jouer le rôle de la source du groupe multicast. D'autres nœuds sont des membres du groupe, et les nœuds restants sont des simples participants. L'élection des entités jouant le rôle de contrôleurs locaux ne se fait que s'ils sont membres du groupe, du fait qu'ils détiennent la clé du groupe et peuvent accéder au flux émis par la source.

Les facteurs que nous avons choisis sont le nombre de membres de groupe (variant entre 10, 20 et 50) et la surface du réseau (variant entre 50*50, 100*100 et 150*150m). Le nombre de nœuds dans le réseau étant fixé à 100. Le facteur de perte de propagation est défini à $\gamma = 2$. Nous assumons que la portée maximale est égale à 50m, tout en sachant qu'une fréquence radio de 2.4 Ghz dans 802.11b offre une portée variant de 30 à 100m, voire 400m selon les caractéristiques physiques des équipements utilisés.

La figure 5.4 montre le nombre de nouveaux clusters comparé au facteur de cohésion. Plus la cohésion d'un cluster est faible, plus il est nécessaire de créer des nouveaux clusters au sein du groupe multicast. La création de ces nouveaux clusters implique la création de nouveaux contrôleurs locaux, utilisés comme relais pour acheminer les flux de données à leurs membres locaux, nécessitant ainsi plus d'énergie de transmission.

La figure 5.4 montre également la liaison entre la métrique yardstick et le facteur de cohésion d'un cluster. Quand la cohésion d'un cluster augmente, le contrôleur local atteint

	Surface	Cohésion	Nombre de clusters (cohésion = 1)	Energie requise	Métrique Yardstick
n = 10	50 * 50	0.99	1.1	1571.73	6.36
	100 * 100	0.5	2.5	3716.45	2.13
	150 * 150	0.35	2.8	4628.53	0.97
n = 20	50 * 50	0.985	1.2	1908.82	10.60
	100 * 100	0.445	3.2	5420.88	3.24
	150 * 150	0.32	3.9	6286.40	1.90
n = 50	50 * 50	0.982	1.2	2195.48	23.03
	100 * 100	0.522	3.9	6749.87	7.47
	150 * 150	0.376	4.9	9229.56	4.34

Avant Clusterisation — Après Clusterisation

FIG. 5.4 – Nombre de cluster, énergie requise et métrique yardstick par rapport à la cohésion

plus de membres locaux en un seul saut, en minimisant la consommation de l'énergie ; ce qui signifie que les facteurs d'atteignabilité (m/n) et de consommation de l'énergie (m/p) augmentent et par conséquent la métrique yardstick.

Nous pouvons ainsi valider le facteur de cohésion utilisé dans OMCT, du fait qu'il est proportionnel au nombre de clusters créés, à l'énergie requise pour l'envoi d'un message en multicast de la source à tous les membres du groupe et à la métrique yardstick. À partir de ce facteur, nous pouvons déduire un seuil minimal, en dessous duquel un groupe multicast doit exécuter l'algorithme de clusterisation OMCT. Il est à noter que le nombre de clusters générés par OMCT est uniquement dépendant du facteur de cohésion et non pas du nombre de membres du groupe.

Comparaison de OMCT avec les travaux existants

L'optimisation de la consommation de l'énergie est une véritable problématique dans la construction des arbres de distribution de clés dans les réseaux ad hoc. Quelques travaux de recherche ont été publiés à ce sujet.

L'algorithme BIP (*Broadcast Incremental Power*) [WNE00], dédié aux réseaux sans fil, détermine l'arbre de diffusion de groupe optimisant la consommation de l'énergie, dont la racine est la source du groupe. Cet arbre, ainsi calculé par BIP, atteint tous les nœuds du réseau sans fil, en calculant leur meilleur point d'attachement, par rapport à l'énergie minimale requise pour les atteindre depuis la source. Cet algorithme profite de l'avantage des communications en diffusion des réseaux sans fil. Cependant, il ne prend pas en compte la mobilité des nœuds.

WIESELTHIER ET AL. [WNE00] déduisent de BIP, un algorithme appelé MIP (*Multicast Incremental Power*) qui calcule le meilleur arbre multicast, également en optimisant l'énergie requise pour atteindre tous les membres du groupe. L'arbre MIP est extrait de celui calculé par BIP, en élaguant les nœuds non adhérents au groupe multicast.

BLU (*Broadcast Least-Unicast-Cost*) [WNE00] superpose les meilleurs chemins unicast, atteignant individuellement tous les nœuds du réseau, selon la métrique de coût minimal. Dans cette approche, ni la mobilité des nœuds, ni l'avantage des communications en diffusion

de l'environnement sans fil ne sont considérés.

Un autre algorithme dédié aux communications multicast, est le MLU (*Multicast Least-Unicast-Cost*). MLU est similaire à l'algorithme BLU, avec la différence que les chemins unicast ne sont établis que pour les membres du groupe multicast. L'arbre multicast est ainsi construit en superposant ces chemins théoriquement optimaux.

Le troisième algorithme présenté par WIESELTHIER ET AL. est BLiMST (*Broadcast Link-Based Minimum-cost Spanning Tree*). Cet algorithme établit l'arbre de diffusion, optimisant l'énergie requise pour atteindre les nœuds du réseau, en utilisant la technique de minimisation des coûts du protocole *spanning tree*. L'avantage des communications sans fil est aussi ignoré dans cet algorithme. Un algorithme appelé MLiMST déduit un arbre multicast depuis celui construit par BLiMST, en élaguant les membres non membres du groupe multicast.

WIESELTHIER ET AL. affirment que MIP établit le meilleur arbre multicast dans les environnements ad hoc, ayant un nombre minimal de relais responsables de l'acheminement du flux multicast et par conséquent l'énergie minimale requise pour atteindre tous les membres du groupe.

Pour comparer notre algorithme OMCT avec MIP, nous utilisons le même exemple présenté dans [WNE00]. Cet exemple correspond à 10 membres adhérents à un groupe multicast et sont joignables à un seul saut de la source du groupe (nœud 10). Avec cette même topologie présentée dans la figure 5.5, OMCT produit un seul cluster avec une cohésion égale à 1, utilisant ainsi un seul contrôleur local (nœud 10 : source du groupe) ; alors que l'algorithme MIP construit l'arbre multicast en calculant les meilleurs points d'attachement des membres du groupe selon l'énergie minimale requise par la source pour les atteindre et non pas l'énergie totale consommée. MIP fournit ainsi une arbre de distribution de clés nécessitant 3 membres relais (10, 9 et 6) et par conséquence plus de consommation d'énergie que l'algorithme OMCT.

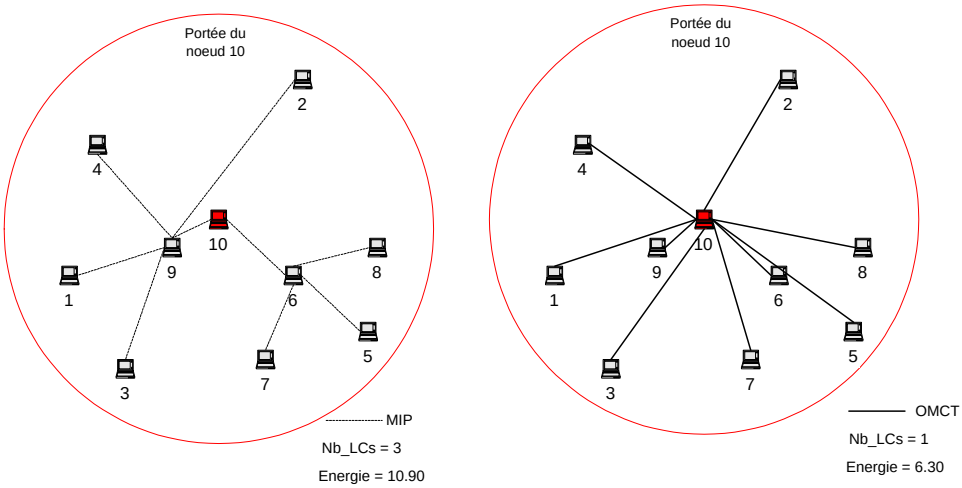


FIG. 5.5 – Comparaison de OMCT par rapport à MIP

La proposition de LAZOS ET AL. [LP03] utilise l'algorithme K-means pour optimiser la consommation de l'énergie du protocole de distribution de clés LKH (*Logical Key Hierarchy*). La procédure de clusterisation avec K-means apporte un gain en énergie de 15% à 37% par rapport au protocole LKH. Cependant comme cela est présenté dans [LP03], pour certaines

topologies, l'algorithme K-means induit une consommation d'énergie plus grande. En effet, l'algorithme K-means échoue parfois à exploiter l'avantage des communications en diffusion de l'environnement sans fil. Dans l'exemple spécifique de deux nœuds, à la même distance de la source mais en directions opposées, l'algorithme K-means les affecte dans deux clusters différents. Cependant, avec l'algorithme OMCT, les deux nœuds appartiendront au même cluster, bénéficiant ainsi de l'avantage des communications en diffusion, optimisant l'énergie et minimisant la latence moyenne de distribution de clés aux membres du groupe multicast.

5.4 Intégration de OMCT dans BALADE

Dans cette section, nous présentons le couplage de l'algorithme de clusterisation dynamique OMCT, avec notre protocole de gestion de clé de groupe BALADE. Nous montrons à travers des simulations présentées dans le chapitre 8 que cette architecture ainsi définie, réduit le délai moyen de transmission des clés aux membres du groupe, la consommation de l'énergie et la bande passante.

La clusterisation en sous-groupes et l'élection des contrôleurs locaux dans BALADE est réalisée par une fonction d'évaluation intégrée au sein de chaque membre du groupe. Cependant, cette fonction ne prend pas en compte la localisation géographique et la mobilité des entités du groupe, ni la consommation d'énergie et de bande passante. Afin de considérer ces contraintes et d'optimiser le processus de distribution de clés dans BALADE, nous avons intégré dans BALADE, à la place de la fonction d'évaluation, l'algorithme de clusterisation dynamique OMCT.

Ainsi, chaque contrôleur local calcule périodiquement le paramètre de cohésion de son cluster. Selon cette valeur, il peut décider ou non d'exécuter l'algorithme OMCT afin de clustériser son sous-groupe. À l'initialisation de BALADE, c'est le contrôleur global CG qui se charge de la clusterisation du groupe multicast.

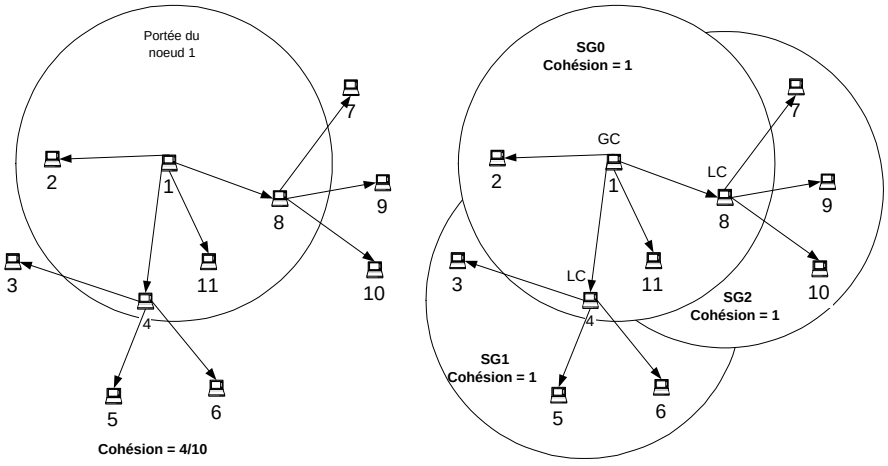


FIG. 5.6 – Intégration de OMCT dans BALADE

La figure 5.6 comporte un exemple d'utilisation de OMCT dans BALADE. Dans cet exemple, le cluster a initialement un facteur de cohésion égal à $\frac{4}{10}$. Ensuite, après la clusterisation avec OMCT, trois nouveaux clusters sont créés ayant des facteurs de cohésion évalués

à 1.

L'utilisation de l'algorithme de clusterisation OMCT dans BALADE rend le processus de distribution de clés de groupe optimisé. En effet, les clusters sont fortement corrélés et tous les membres des sous-groupes appartiennent à la portée de transmission de leurs contrôleurs locaux respectifs. Ainsi, la distribution de la TEK au sein de chaque cluster se fait sans nœuds relais. Les communications inter-clusters se limitent seulement à la transmission de la TEK de la source du groupe à tous les contrôleurs locaux. La spécification du protocole BALADE couplé avec OMCT est présentée dans le chapitre 6, en vue de sa validation et de son implantation.

5.5 Utilisation des relais multipoints dans OMCT

Le concept d'intégration de couches protocolaires (applicative et réseau) est spécialement intéressant dans les réseaux ad hoc où le protocole de routage joue un rôle primordial. De plus, une telle intégration exploite l'avantage des communications de diffusion dans les environnements sans fil. Partant de cette motivation, nous avons décidé de combiner l'algorithme OMCT avec la technique de relais multipoints, utilisée dans le protocole de routage OLSR (*Optimized Link State Routing*) [CJA+03]. Cette intégration de couches protocolaires permet de prendre en compte la localisation et la mobilité des nœuds, tout en évitant l'utilisation du système de localisation GPS. L'efficacité de la combinaison de OMCT avec la technique de relais multipoints en termes de délai et de taux de transmission de clés et de consommation d'énergie, est validée à travers des analyses et des simulations.

5.5.1 Technique de relais multipoints dans OLSR

OLSR [CJA+03] est un protocole de routage pour les réseaux ad hoc. Le principe fondateur de OLSR est l'utilisation de la technique de relais multipoints (MPRs) pour l'inondation des messages de contrôle dans le réseau, en réduisant le nombre de retransmissions. Le concept des MPRs est ainsi une optimisation du mécanisme classique de diffusion.

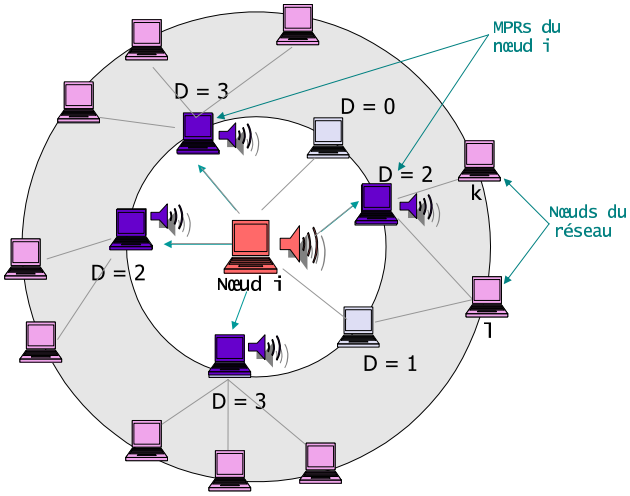


FIG. 5.7 – Diffusion via les MPRs dans OLSR

Chaque nœud dans le réseau élit son ensemble de relais multipoints, indépendamment des autres nœuds. Cet ensemble d'entités MPRs est choisi parmi le voisinage à un saut du nœud. Un nœud doit être capable d'atteindre tous ses voisins à deux sauts, via son ensemble de nœuds MPRs, comme cela est illustré dans la figure 5.7. Dans cette figure, le membre i atteint les nœuds k et l, à travers le nœud j, qui joue le rôle d'un MPR du nœud i.
La maintenance d'un ensemble minimal de nœuds MPRs assure une optimisation du surcoût du protocole. Il est à noter qu'initialement, l'ensemble des MPRs d'un nœud coïncide avec tous ses voisins à un saut. L'heuristique de sélection des entités MPRs est détaillée dans [CJA+03].

5.5.2 Intégration des relais multipoints dans OMCT

Description générale du protocole

L'algorithme de clusterisation OMCT tend à élire l'ensemble optimal des contrôleurs locaux des clusters de façon à optimiser le délai de transmission des clés et améliorer le taux d'acheminement de l'information. D'un autre côté, la technique d'élection des MPRs dans OLSR tend à réduire le nombre de retransmission et de relais dans le réseau.
Vue la similarité de ces motivations, nous nous proposons d'étudier la possibilité d'intégration de la technique des MPRs dans OMCT (intégration des couches protocolaires réseau et application) [BCF06]. L'idée de base de cette intégration est l'utilisation de l'information des relais multipoints (collectée par le protocole OLSR) lors de l'élection des contrôleurs locaux des nouveaux clusters d'OMCT.

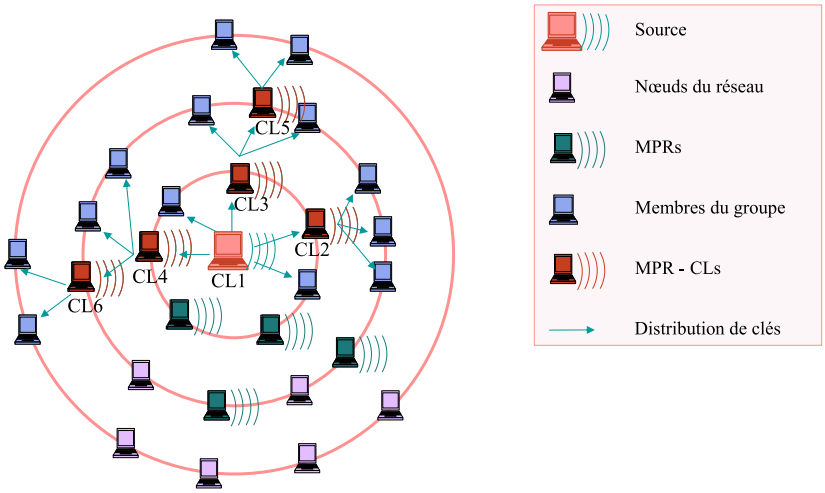


FIG. 5.8 – Intégration des MPRs dans OMCT

Notre nouveau schéma de clusterisation commence par la source du groupe, en collectant son ensemble de nœuds MPRs par OLSR. Ensuite, parmi ces MPRs, sont choisis les MPRs membres du groupe multicast et ayant des membres fils adhérents également au groupe (le MPR appartient au chemin entre la source du groupe et le nœud fils membre du groupe). Les nœuds ainsi sélectionnés seront élus des contrôleurs locaux. À cette étape, le voisinage à deux sauts de la source du groupe est couvert par les contrôleurs locaux élus.

Ce schéma est ensuite réitéré jusqu'à ce que tous les membres du groupe soient couverts

par les contrôleurs locaux, en exécutant la même règle : un MPR d'un contrôleur local, qui est un membre du groupe et qui a des membres fils adhérents au groupe, devient un contrôleur local. L'arbre multicast est ainsi construit, au niveau applicatif, comme cela est illustré dans la figure 5.8. Dans cet exemple, le contrôleur local 1 délègue le processus de distribution de clés aux nœuds 2, 3 et 4 qui appartiennent à l'ensemble de ses MPRs et qui ont des membres fils affiliés au groupe. À la deuxième étape, les nœuds 5 et 6 sont élus respectivement par les contrôleurs 3 et 4, comme des contrôleurs locaux.

Algorithme

Les principales étapes de notre nouvelle approche de clusterisation intégrant les MPRs dans OMCT, sont présentées ci-dessous (cf. Annexe A - Algorithme 4) :

- Étape 1 :
  Initialement, la liste des contrôleurs locaux contient uniquement la source du groupe, qui évalue le facteur de cohésion de son groupe et décide de le diviser en clusters via l'algorithme OMCT. La liste des MPRs du contrôleur local courant est ensuite collectée.
- Étape 2 :
  La liste des MPRs est parcourue tant que des membres du groupe restent non adhérents à de nouveaux clusters (non couverts par des contrôleurs locaux). Tout MPR de cette liste, adhérent au groupe et ayant des membres fils appartenant également au groupe, est ajouté à la liste des contrôleurs locaux. Les membres du groupe accessibles par un nouveau contrôleur local élu, formant avec lui un cluster, sont supprimés de la liste des membres du groupe.
- Étape 3 :
  Dans le cas où des membres du groupe ne sont pas encore couverts par les clusters formés, à cause de l'absence ou de l'insuffisance des MPRs et si le facteur de cohésion reste inférieur au seuil $Min\_Cohésion$ défini, notre approche choisit parmi ces membres (non encore couverts) les nœuds ayant une accessibilité maximale par rapport aux autres nœuds en un seul saut. Cette information d'accessibilité est partiellement collectée via le protocole de routage OLSR, consolidée par la signalisation de OMCT. La figure 5.9 comporte un exemple d'exécution de cette troisième étape. Le contrôleur local CL1 élit les contrôleurs CL2 et CL3, pour assurer la distribution de clés à ses membres locaux. Cependant, des membres du groupe ne sont toujours pas couverts par les clusters formés. Les nœuds 4 et 5 sont ainsi sélectionnés pour assurer le rôle de contrôleurs locaux pour les membres du groupe restants.

La ré-élection des contrôleurs locaux

La liste des MPRs est mise à jour à chaque changement du voisinage à un ou deux sauts. Ainsi, la liste des MPRs élus comme contrôleurs locaux doit aussi être mise à jour. Cette mise à jour ne devrait pas affecter le processus de distribution de clés, en réalisant les opérations suivantes :

- Quand un MPR contrôleur local disparaît du réseau ad hoc à cause d'un problème de ressources, ou se déplace de sa localisation géographique, tous les membres de son cluster se joignent directement au cluster parent. Le contrôleur local de ce cluster déclenchera un processus de clusterisation si le paramètre de cohésion dépasse le seuil minimum défini $Min\_Cohésion$.

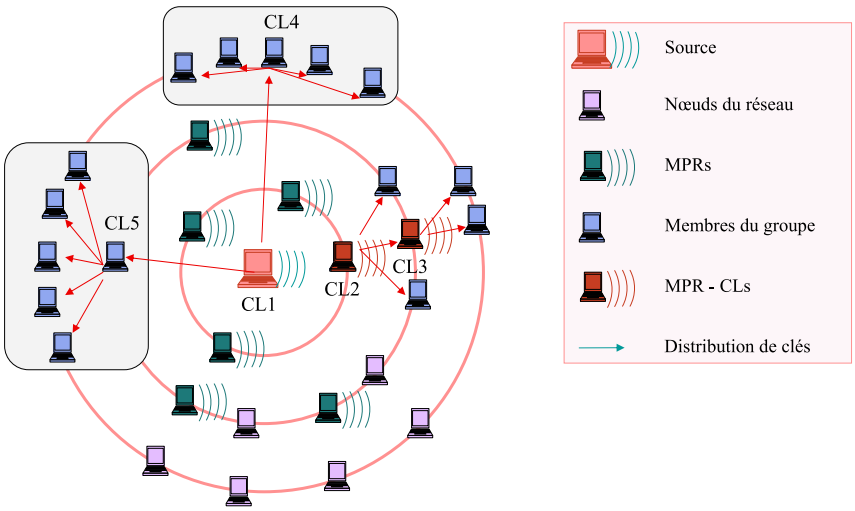


FIG. 5.9 – Élection des membres du groupe en tant que CLs

– Les MPRs contrôleurs locaux envoient périodiquement des messages *CL-Queries*, de façon à ce que les membres du groupe peuvent choisir leurs contrôleurs locaux, les plus proches de leur localisation géographique, en terme de nombre de sauts. Afin d'optimiser la consommation de la bande passante, les messages *CL-Queries* sont intégrés dans les messages de contrôle de la topologie de OLSR (*TC : Topology Control*). Un message TC est envoyé en broadcast dans tout le réseau, par tout nœud pour déclarer ses liens et les nœuds qui l'ont choisi comme leur MPR [CJA+03]. Le champ "*Reserved*" du message TC avec une valeur spéciale (0xFFFF) est interprété comme un *CL-Query* d'un contrôleur local.

5.5.3 Analyse

Les principaux avantages de l'intégration de la technique de relais multipoints dans OMCT sont résumés dans le tableau 5.1. Notre approche ne requiert pas l'information de localisation géographique des membres, collectée via le système GPS, comme cela est défini dans l'algorithme OMCT de base. Elle a recours aux informations collectées via le protocole de routage OLSR, sous forme d'intégration de couches protocolaires.

L'intégration des MPRs dans OMCT réduit le surcoût de calcul, induit par la recherche des CLs dans OMCT. De plus, la complexité de l'algorithme OMCT est évaluée à $O(3n^2)$, alors que celle de l'algorithme OMCT combiné avec les MPRs est évaluée à $O(3n)$, n étant le nombre de membres du groupe.

Notre approche apporte également une économie de consommation de la bande passante, comparée à OMCT. En effet, elle utilise les messages OLSR pour la signalisation et par conséquence, le nombre de messages additionnels transmis dans le réseau est nettement inférieur à celui avec OMCT.

Dans l'algorithme OMCT, la connectivité entre un contrôleur local et les membres de son cluster n'est pas toujours assurée, même si la distance entre eux est inférieure à la portée radio du CL (problèmes de médium, obstacles, ...). Cependant, en utilisant la technique de relais

OMCT	OMCT avec MPRs
Utilise la localisation géographique de tous les membres du groupe, collectée par GPS	Utilise l'information de multipoint relais, collectée par le protocole OLSR (Pas de GPS)
Complexité = $O(3n^2)$	Complexité = $O(3n)$
Signalisation *out-band*	Signalisation *in-band* : via les messages OLSR. Optimisation de la consommation de la bande passante
La connectivité entre les CLs et leurs membres locaux n'est pas toujours assurée, même si la distance entre eux est inférieure à la portée radio du CL	La connectivité entre les CLs et leurs membres locaux est toujours vérifiée : les CLs sont des MPRs élus et maintenus par le protocole de routage OLSR

TAB. 5.1 – Récapitulation des comparaisons

multipoints de OLSR lors de l'élection des contrôleurs locaux, la connectivité entre les CLs et leurs membres est toujours réelle.

5.6 Conclusion

Dans ce chapitre, nous avons présenté une approche de clusterisation dynamique pour la distribution de clés de groupe, dans les réseaux ad hoc. Notre approche, basée sur l'algorithme OMCT, prend en compte la localisation et la mobilité des nœuds et optimise la consommation de l'énergie et de la bande passante.

Le principe de base de OMCT est de diviser le groupe multicast dynamiquement en des clusters, gérés chacun par un contrôleur local, responsable de la gestion de la sécurité au sein de son cluster. L'algorithme OMCT est intégré dans le protocole de gestion de clé de groupe BALADE adapté aux réseaux ad hoc, afin d'aboutir à un processus de distribution de clé plus optimisé. La spécification du protocole BALADE couplé avec OMCT est présentée dans le chapitre 6.

OMCT tel qu'il a été défini, utilise le système de localisation géographique GPS. Pour pouvoir éviter le surcoût engendré par ce système, nous avons étudié un mécanisme d'intégration de couches protocolaires, combinant la technique de relais multipoints utilisée dans le protocole de routage OLSR, avec l'algorithme OMCT. Cette combinaison consiste à élire les contrôleurs locaux de OMCT parmi les entités choisies comme relais multipoints par le protocole OLSR. Le mécanisme ainsi construit, apporte une économie en calcul et en bande passante, tout en assurant un processus de distribution de clés fiable et optimisé. Notre approche d'intégration de couches protocolaires reste toutefois liée à OLSR, le protocole de routage sous-jacent.

L'ensemble des simulations réalisées afin d'évaluer les performances de OMCT et l'intégration des MPRs dans OMCT sont présentées dans le chapitre 9.

Chapitre 6

Spécification de BALADE avec OMCT

Sommaire

6.1 Introduction

BALADE, couplé avec OMCT, est un protocole de gestion de clé de groupe dans les réseaux ad hoc. Il est composé de différents sous-protocoles, formés chacun par un ensemble de messages et de connaissances échangés et faisant intervenir différentes entités du groupe. Avant d'entreprendre la validation et l'implantation de notre protocole, nous procédons à sa spécification semi-formelle. Cette étape primordiale tend à éliminer toute éventuelle ambiguïté due à la description du protocole en langage naturel. Elle consiste à spécifier les entités qui participent au fonctionnement de BALADE, les messages échangés entre ces entités et les connaissances que détient chaque entité (statiques et/ou dynamiques).

Nous entamons cette étape de spécification par la présentation des pré-requis que nous avons utilisés : la notation adoptée, l'identification des entités participantes au protocole, le format des messages échangés entre elles ainsi que leurs connaissances. Nous procédons dans un second temps à la spécification des sous-protocoles suivants :

- Initialisation de BALADE,
- Renouvellement périodique de la clé de chiffrement de données TEK,
- Authentification et contrôle d'accès,
- Join et leave d'un nouveau membre,
- Ré-intégration d'un membre du groupe,
- Exclusion d'un membre du groupe,
- Envoi périodique des CL_Queries par les contrôleurs locaux,
- Gestion des listes de contrôle d'accès (ACL) et de révocation (RL),
- Retransmission de la TEK par les contrôleurs de BALADE,
- Clusterisation avec l'algorithme OMCT.

6.2 Pré-requis de la spécification

6.2.1 Notation adoptée

Dans ce qui suit, nous présentons le glossaire des acronymes que nous utilisons tout au long de ce chapitre :

- G_k : désigne le groupe multicast k
- ACL : liste de contrôle d'accès (*Access Control List*)
- RL : liste de révocation (*Revocation List*)
- LML : liste des membres locaux
- SG_{ik} : sous-groupe i appartenant au groupe k
- GCL : Groupe des contrôleurs locaux
- TEK : clé de chiffrement des données du groupe (*Traffic Encryption Key*)
- KEK_CSG_{ik} : clé locale du sous-groupe i du groupe k, jouant le rôle d'une clé de chiffrement de la clé TEK (*Key Encryption Key*)
- KEK_CCL : clé du groupe des contrôleurs locaux, jouant également le rôle d'une clé de chiffrement de la clé TEK
- CBID_mg_{ik} : identificateur cryptographique (*Crypto Based IDentifier*) du membre mg_{ik}
- Pub_mg_{ik} : clé publique du membre mg_{ik}
- Pri_mg_{ik} : clé privée du membre mg_{ik}
- Coord_nœud : les coordonnées GPS du nœud
- $\rightarrow$: message envoyé en unicast

- $\Rightarrow$: message envoyé en multicast
- $\Rrightarrow$: message envoyé en broadcast.

6.2.2 Entités

Nous distinguons trois types de participants aux différents sous-protocoles de BALADE. Le premier type de participant correspond à un membre ordinaire du groupe multicast. Le deuxième type correspond au contrôleur global de l'ensemble du groupe multicast. Et finalement, le dernier type d'acteurs de BALADE est le contrôleur local, qui a à sa charge d'assurer la sécurité au sein d'un sous-groupe ou cluster.

- mg_k : désigne un membre du groupe k,
- mg_{ik} : désigne un membre du groupe k, appartenant au cluster i,
- CG_k : désigne le contrôleur global du groupe k,
- CL_{ik} : désigne le contrôleur local du cluster i appartenant au groupe k,
- $Admin$: désigne l'administrateur du groupe multicast, responsable de la création et de l'initialisation de la session du groupe.

6.2.3 Messages

Les messages échangés entre les différentes entités de BALADE seront définis dans la spécification des différents sous-protocoles, en utilisant la notation suivante :

- $\{m\}k$: désigne le message m crypté avec la clé k.
- (m_1, m_2) : désigne la concaténation de deux messages m_1 et m_2.

Il est à noter que l'intégrité des messages ainsi que la tolérance aux attaques de rejeux ne sont pas prises en compte dans notre spécification de BALADE couplé avec OMCT. Pour assurer ces deux services de sécurité, il suffit d'ajouter à chaque message envoyé un numéro de séquence et une signature de l'émetteur.

6.2.4 Connaissances

Nous distinguons pour chaque entité de BALADE des connaissances statiques, invariables au cours de la session et des connaissances dynamiques.

Connaissances statiques

Tout nœud du réseau mg détient une clé publique Pub_mg et une clé privée Pri_mg, qu'il a générées de façon autonome, ou qu'il a reçu d'une autorité de certification hors ligne. À partir de ces clés, tout membre du réseau mg construit son identificateur unique $CBID\_mg$, de la façon suivante [MC02] :

$$CBID\_mg = hmac\_sha1\_128(sha1(imprint), sha1(Pub\_mg))$$

avec *imprint* un entier sur 8 octets qui peut être choisi aléatoirement.

Durant une session de communication multicast, tout membre mg connaît également les identificateurs des groupes multicast auxquels il est adhérent.

Connaissances dynamiques

Tout membre du groupe k, noté mg_k, connaît son contrôleur global actuel CG_k, qui est la source du flux multicast à l'instant courant. Tout membre mg_k connaît également l'identificateur i de son sous-groupe SG_{ik}, ainsi que son contrôleur local CL_{ik}.

La clé de chiffrement de données TEK est distribuée par le contrôleur global du groupe à tous ses membres. En plus, chaque membre mg_{ik} connaît la clé de chiffrement de clé de son sous-groupe (clé locale) KEK_CSG_{ik}.

Tous les contrôleurs (global et locaux) connaissent à tout moment la liste de leurs membres locaux, appelée LML. Ils coopèrent et collaborent également pour gérer la liste de contrôle d'accès ACL et celle des membres exclus RL.

6.3 Spécifications des sous-protocoles de BALADE couplé avec OMCT

Pour spécifier les sous-protocoles de BALADE de façon complète, nous spécifions les entités participantes, les échanges de messages entre elles et les connaissances initiales qu'elle doivent détenir.

6.3.1 Initialisation de BALADE

Mise en place de l'ACL

La liste de contrôle d'accès d'un groupe multicast est construite hors ligne et contient tous les membres autorisés à joindre ce groupe, dès le début de la session ou ultérieurement. C'est l'administrateur qui est responsable de la construction de cette ACL et de l'envoi au premier contrôleur global du groupe correspondant. Cet administrateur n'a aucune charge en ligne ; c'est à dire que la gestion et la sécurité des membres du groupe sont assurées par les membres eux-même sans aucune autorité centrale.

Génération et distribution de la TEK

Le contrôleur global envoie un message de distribution de la clé du groupe à tous ses membres locaux (initialement membres du sous-groupe 0).

TEK Distribution
$\forall mg_k,\ CG_k \longrightarrow mg_k : \{TEK, Num\_Seq, KEK\_CSG_{0k}, IDG, ID_{CG}, Pub\_CG,$ $imprint, \{CBID - CG\}Pri\_CG\}Pub\_mg_k$

avec :

1. TEK : clé de chiffrement de trafic,
2. Num_Seq : Numéro de séquence correspondant à la clé de chiffrement de données TEK,
3. KEK_CSG_{0k} : clé locale du cluster 0,
4. IDG : identité du groupe,
5. ID_{CG} : identité du CG, qui peut être son adresse IP,
6. CBID-CG : CBID du CG ,
7. Pub_mg_k : clé publique du membre mg_k

Pour améliorer la fiabilité de la distribution de la clé de chiffrement de données TEK, un numéro de séquence correspond à chaque TEK. Ce même numéro de séquence sera ajouté au flux multicast chiffré par cette TEK. Num_Seq est un entier incrémenté à chaque renouvellement de la TEK. Ainsi, un membre qui ne reçoit pas la clé de chiffrement de données lors du renouvellement de la TEK, à cause d'un problème de connexion, peut détecter que la TEK qu'il détient ne correspond pas au flux chiffré qu'il reçoit et demander ainsi à son contrôleur local de la lui re-transmettre.

Pour s'identifier auprès des membres du groupe, le contrôleur global envoie son CBID chiffré avec sa clé privée, ainsi que sa clé publique et l'entier *imprint* qu'il a utilisé pour générer son CBID. Chaque membre du groupe recevant ce message commence par vérifier l'authenticité du CG en déchiffrant son CBID et le comparant avec le CBID calculé via la clé publique et l'entier *imprint* du CG.

En cas de succès, le membre du groupe stocke la clé de chiffrement des données, le numéro de séquence et la clé locale de son sous-groupe 0.

Connaissances initiales

CG_k	$IDG, ID_{CG}, TEK, Num\_Seq, \forall Pub\_mg_k$
$\forall mg_k$	$Pub\_mg_k, Pri\_mg_k, IDG, ID_{CG}$

6.3.2 Renouvellement périodique de la TEK

On se place dans le cadre où le processus de clusterisation avec OMCT a été enclenché, car un nombre important de membres ont joint le groupe multicast, avec une faible cohésion. La clusterisation divise le groupe multicast en des clusters, gérés chacun par un CL responsable de la gestion et de la sécurité de son sous-groupe.

Le renouvellement de la clé de chiffrement de trafic TEK est enclenché après chaque unité sémantique des données multicast et suit le processus de distribution suivant : le contrôleur global envoie un message de distribution de la clé du groupe ainsi que le numéro de séquence correspondant aux membres de son sous-groupe 0. Ensuite, il envoie ce même message aux contrôleurs locaux du groupe (CLs), qui l'acheminent aux membres de leur cluster respectif.

TEK Renewal
$\forall mg_{0k} \in LML\_CG :$
$CG_k \Rightarrow mg_{0k} : \{ID_{CG}, TEK, Num\_Seq\}KEK\_CSG_{0k}$
$\forall CL_{ik} \in GLC :$
$CG_k \Rightarrow CL_{ik} : \{ID_{CG}, TEK, Num\_Seq\}KEK\_CCL$
$\forall CL_{ik} \in GCL, \forall mg_{ik} \in LML\_CL_{ik} :$
$CL_{ik} \Rightarrow mg_{ik} : \{ID_{CL}, TEK, Num\_Seq\}KEK\_CSG_{ik}$

avec :

1. ID_{CG} : identité du CG
2. ID_{CL} : identité du CL

On note que ce processus de distribution de clé reste valable dans le cas où le groupe multicast est constitué d'un seul cluster, géré par le CG.

Connaissances initiales

CG_k	$IDG, ID_{CG}, TEK, Num\_Seq, KEK\_CSG_{0k}, KEK\_CCL$
$\forall CL_i$	IDG, ID_{CG}, KEK_CCL
$\forall mg_{ik}$	$Pub\_mg_{ik}, Pri\_mg_{ik}, IDG, ID_CG, KEK\_CSG_{ik}$

6.3.3 Ajout d'un nouveau membre (Procédure Join)

Quand un nouveau membre mg_k veut joindre le groupe multicast, il doit s'authentifier auprès d'un contrôleur local CL_{ik} du groupe qui vérifie en plus s'il est autorisé ou non à joindre le groupe. Le nouveau membre peut connaître le contrôleur local CL_{ik} en recevant un message CL_Query de la part de CL_{ik}, l'informant de sa localisation géographique et de l'identité du groupe multicast pour lequel il joue le rôle d'un CL.

Dans le cas où un nouveau membre ne connaît aucun contrôleur local auprès duquel il peut entamer une procédure de Join, il envoie une requête en broadcast à tous les membres du réseau, cherchant à connaître un contrôleur. Le premier membre du groupe qui reçoit cette requête lui répond avec un message contenant l'identité de son contrôleur local.

Dans ce qui suit, nous considérons qu'un nouveau membre connaît l'identité du CL du cluster auquel il veut adhérer.

Node Authentication and Access Control
$mg_k \longrightarrow CL_{ik} : Pub\_mg_k, imprint, \{CBID\_mg_k\}Pri\_mg_k$

Le membre envoie sa clé publique et l'entier *imprint* qu'il a utilisé pour générer son identificateur CBID. Le contrôleur local, ayant reçu ce message, commence par authentifier le membre, en déchiffrant le CBID à l'aide de la clé publique, ensuite en calculant de nouveau le CBID via la clé publique et l'entier *imprint*. En cas de succès, le CL vérifie si le membre est bien autorisé à joindre le groupe multicast, via l'ACL du groupe, gérée par le groupe des contrôleurs locaux. À ce stade, si le contrôle d'accès réussit, la demande d'adhésion du nouveau membre est acceptée au sein du cluster concerné et ce membre nommé mg_{ik}, sera ajouté à la liste des membres locaux LML du CL_{ik}.

Pour faire face aux rejeux de messages et aux attaques de déni de service (attaques DoS), le contrôleur local doit demander à un nouveau membre, voulant joindre le groupe, de répondre à un puzzle (puzzle-request), avant d'entamer la vérification de son authentification et de son contrôle d'accès.

Node Authentication and Access Control
$mg_k \longrightarrow CL_{ik} : Join - Request, Pub\_mg_k$
$CL_{ik} \longrightarrow mg_k : Puzzle - Request$
$mg_k \longrightarrow CL_{ik} : Puzzle - Reply, Pub\_mg_k, imprint, \{CBID\_mg_k\}Pri\_mg_k$

Un puzzle-request est efficace s'il a les propriétés suivantes [ANL01] :

1. la création du puzzle et la vérification ne sont pas coûteuses pour le contrôleur local,

2. le coût pour résoudre le puzzle est facile à ajuster,
3. le puzzle peut être calculé par des nœuds de faible capacité de calcul,
4. le résultat du puzzle ne peut pas être pré-calculé à l'avance.

Un exemple de puzzle-request est donné dans [ANL01]. Il est basé sur une fonction à sens unique (*one way function*)(MD5 ou SHA). Le Puzzle-Request envoyé par le serveur est une valeur aléatoire N_s et un niveau de difficulté k. Pour répondre au puzzle, le client génère une valeur aléatoire N_c et calcule X et Y, de sorte que :

$$(i) \quad h(C, N_s, N_c, X) \; = \; 0_1 0_2 ... 0_k \; Y$$

h étant une fonction cryptographique de hachage et C étant l'identité du client. Le Puzzle-Reply contient donc l'identité du client C, les deux valeurs aléatoires N_s et N_c, ainsi que la solution X. Pour valider cette réponse, le serveur n'a besoin que de vérifier l'égalité (i).

L'ajout d'un nouveau membre au sein d'un cluster nécessite le renouvellement de la clé locale de ce cluster et sa distribution à tous les membres locaux, y compris le nouvel adhérent, comme cela est illustré dans le tableau suivant. Un mot de passe est également envoyé au nouveau membre, pour pouvoir l'utiliser en cas de ré-intégration (voir section suivante).

Join Procedure
$for\ ancien\_mg_{ik} : anciens\ membres\ du\ cluster$
$CL_{ik} \Rightarrow ancien\_mg_{ik} : \{ID_{CL}, KEK\_CSG_{ik}\} ancienne\_KEK\_CSG_{ik}$
$CL_{ik} \longrightarrow mg_{ik} : \{ID_{CL}, TEK, KEK\_CSG_{ik}, Passwd\} Pub\_mg_{ik}$

Connaissances initiales

CL_{ik}	$ID_{CL}, TEK, KEK\_CSG_{ik}, ancienne\_KEK\_CSG_{ik}, Passwd$
mg_{ik}	$Pub\_mg_{ik}, Pri\_mg_{ik}, imprint, CBID_{m}g_{k}$

6.3.4 Ré-intégration d'un membre du groupe

Un membre du groupe mg_k qui se déplace dans le réseau ad hoc, peut perdre sa connectivité vers son cluster et essayer ensuite de rejoindre le groupe multicast. Un membre ad hoc peut également perdre son appartenance au groupe multicast à cause de problèmes de ressources (batterie) et essayer de le ré-intégrer ensuite.

Dans ces deux cas d'utilisation, pour éviter de refaire la procédure de Join, l'ancien membre $ancien\_mg_k$ peut ré-intégrer son groupe en envoyant un message de ré-authentification à n'importe quel membre du groupe mg_{ik}, via le ticket de ré-authentification, comme suit :

Re – authentication Procedure
$ancien\_mg_k \longrightarrow mg_{ik} : Pub\_ancien\_mg_k, \{CBID\_ancien\_mg_k\} Pri\_ancien\_mg_k,$ $imprint,\ \{\{Passwd\} TEK\} Pub\_mg_{ik}$
$mg_{ik} \longrightarrow ancien\_mg_k : \{KEK\_CSG_{ik}\} Pub\_ancien\_mg_k$

Le membre mg_{ik} peut être un contrôleur local du groupe multicast, connu via les messages CL_Queries envoyés périodiquement par tous les contrôleurs locaux. Si l'ancien membre $ancien\_mg_k$ ne connaît aucun membre du groupe (ou aucun contrôleur local), il envoie un message en broadcast à tous les membres du réseau. Le premier membre du groupe recevant cette requête, lui répond avec son identité et l'identité de son contrôleur local.

Le ticket de ré-authentification est un mot de passe connu par tous les membres du groupe et qui doit être envoyé, chiffré avec la clé de chiffrement de données courante (TEK). Dès que le membre du groupe mg_k reçoit le message de ré-authentification de $ancien\_mg_k$, il vérifie si le ticket est valable, c'est à dire si le mot de passe est bien chiffré avec la clé de chiffrement de données courante. Le double chiffrement du mot de passe avec la clé du groupe et la clé publique du destinataire est nécessaire pour assurer la confidentialité de la clé du cluster (cf. 7.3.2).

Connaissances initiales

$ancien\_mg_k$	$Pub\_ancien\_mg_k, Pri\_ancien\_mg_k, CBID\_ancien\_mg_k, imprint,$ $Pub\_mg_{ik}, Passwd, TEK$
mg_{ik}	$Pub\_mg_{ik}, Passwd, TEK, KEK\_CSG_{ik}$

6.3.5 Départ d'un membre du groupe (Procédure Leave)

Une demande pour quitter le groupe Leave est toujours acheminée vers le contrôleur local du cluster concerné.

Node Authentication
$mg_k \longrightarrow CL_{ik} : Pub\_mg_k, imprint, \{CBID\_mg_k\}Pri\_mg_k$

Comme dans le cas de Join, le CL procède à l'authentification du membre voulant quitter le groupe multicast, ensuite et en cas de succès, il enlève le membre sortant de sa liste LML et enclenche un processus de renouvellement de la clé locale du cluster, comme suit :

Leave Procedure
$mg_{ik}\_sortant$ *est le membre quittant le groupe* *for* mg_{ik} *: membre local,* $mg_{ik} \neq mg_{ik}\_sortant$ $CL_{ik} \longrightarrow mg_{ik} : \{ID_{CL}, KEK\_CSG_{ik}\}Pub\_mg_{ik}$

avec ID_{CL} : identité du CL.

Connaissances initiales

$mg_k\_sortant$	$Pub\_mg_k, Pri\_mg_k, imprint, CBID\_mg_k$
CL_{ik}	ID_CL, KEK_CSG_{ik}
$\forall mg_{ik}\_restant$	Pri_mg_{ik}

Leave d'un contrôleur local du groupe multicast

Lorsqu'un contrôleur local veut quitter le groupe multicast, il demande à ses membres locaux d'intégrer d'autres clusters (message Merge), dans une durée maximale notée Merge_Timer. Ensuite, il envoie un message leave au groupe des contrôleurs locaux, qui procèdent au changement de la clé de leur groupe KEK_CCL (par le contrôleur global).

Merge
$\forall mg_{ik}, CL_{ik} \Rightarrow mg_{ik} : \{ID\_Cluster, LML\_CL_{ik}\}KEK\_CSG_{ik}$
Notification_Départ
$CL_{ik} \Rightarrow GCL : \{ID\_CL_{ik}\}KEK\_CCL$ $\forall j \neq i,\ CG_k \Rightarrow CL_{jk} : \{ID\_CG, new\_KEK\_CCL\}Pub\_CL_{jk}$

Dans le cas où un contrôleur local quitte le groupe sans notifier de son départ (à cause d'un problème de batterie par exemple), ses membres locaux vont se rendre compte après un certain délai qu'il ne peuvent plus recevoir la clé de chiffrement de données émise par la source. Ils joindront donc un autre contrôleur local.

6.3.6 Exclusion d'un membre du groupe

La révocation ou l'exclusion d'un membre du groupe est réalisée quand un membre ne respectant pas les consignes du bon déroulement de l'application multicast en question, est détecté[14].

Dans ce cas, le contrôleur local ajoute le membre exclu à la liste de révocation RL (cf. 6.3.8) et l'enlève de la liste LML de ses membres locaux. Ensuite, il enclenche un processus de renouvellement de la clé locale de son cluster, comme est décrit dans le cas de départ volontaire. De plus, la révocation d'un membre du groupe nécessite le renouvellement total de la clé de chiffrement du trafic TEK. Le CL envoie ainsi une demande de renouvellement de la clé TEK au contrôleur global.

Node Exclusion
$mg_{ik}\_sortant$ *est le membre exclu du groupe* $for\ mg_{ik} : membre\ local,\ mg_{ik} \neq mg_{ik}\_sortant$ $CL_{ik} \longrightarrow mg_{ik} : \{ID_{CL}, KEK\_CSG_{ik}\}Pub\_mg_{ik}$
$CL_{ik} \longrightarrow CG_k : \{ID_{CL}, Pub\_CL_k, imprint, \{CBID\_CL_k\}Pri\_CL_{ik}\}$
$\forall mg_{0k} \in LML\_CG :$ $CG_k \Rightarrow mg_{0k} : \{ID_{CG}, TEK, Num\_Seq\}KEK\_CSG_{0k}$ $\forall CL_{ik} \in GLC :$ $CG_k \Rightarrow CL_{ik} : \{ID_{CG}, TEK, Num\_Seq\}KEK\_CCL$ $\forall CL_{ik} \in GCL, \forall mg_{ik} \in LML\_CL_{ik} :$ $CL_{ik} \Rightarrow mg_{ik} : \{ID_{CL}, TEK, Num\_Seq\}KEK\_CSG_{ik}$

[14]La présence d'un mécanisme de détection de comportements malicieux ou égoïstes est assumée au sein du réseau ad hoc.

Connaissances initiales

$\forall mg_{ik} \neq mg_{ik}\_sortant$	Pub_mg_{ik}
CL_{ik}	$ID\_CL, KEK\_CSG_{ik}, \forall Pub\_mg_{ik}$
CG_k	$IDG, ID_{CG}, TEK, Num\_Seq, KEK\_CSG_{0k}, KEK\_CCL$
$\forall CL_{ik}$	IDG, ID_{CG}, KEK_CCL
$\forall mg_{ik}$	$Pub\_mg_{ik}, Pri\_mg_{ik}, IDG, ID_CG, KEK\_CSG_{ik}$

6.3.7 Envoi périodique des CL_Queries

Périodiquement et à chaque intervalle noté CL_Query_Interval, chaque contrôleur local diffuse un message nommé CL_Query, avec un TTL=2. Le CL_Query a la forme suivante :

CL_Query
$CL_{ik} \Rrightarrow (TTL = 2) : \{ID\_CL_{ik}, Pub\_CL_{ik}, imprint, \{CBID\_CL_{ik}\} Pri\_CL_{ik}, Coord\_CL_{ik}\}$

Connaissances initiales

CL_{ik}	$ID\_CL_{ik}, Pub\_CL_{ik}, imp, CBID\_CL_{ik}, Coord\_CL_k$

6.3.8 Gestion des listes ACL et RL

L'objectif de cette gestion est de pouvoir organiser la coopération entre les contrôleurs de BALADE, partager les listes ACL et RL entre ces nœuds et en assurer la maintenance, la disponibilité, l'accessibilité et la cohérence[15]. Deux principaux algorithmes ont été mis en place, l'algorithme de recherche de l'information et l'algorithme de maintenance et de mise à jour de l'information :

- algorithme de recherche de l'information des ACLs, permettant à tout contrôleur local de pouvoir demander l'autorisation d'un nouveau membre du groupe au nœud qui détient l'entrée ACL correspondante.
- algorithme de maintenance et de mise à jour de l'information, permettant d'ajouter ou de supprimer des entrées dans l'ACL distribuée, tout en assurant sa cohérence et sa disponibilité. Une redondance des entrées est donc requise pour pouvoir faire face aux problèmes de ressources limitées et de perte de connectivité dans un réseau ad hoc.
  Considérons n : nombre de membres du groupe multicast, k : nombre des contrôleurs locaux du groupe et f : nombre de redondances requises par la politique de gestion de l'application concernée. Le nombre d'entrées que chaque contrôleur local doit stocker dans son fichier ACL est donc : $f * \frac{n}{k}$.

[15] Ce travail a été réalisé avec des étudiants de l'École Supérieure d'Informatique et d'Applications de Lorraine (ESIAL), dans le cadre de leur projet 2A

Les requêtes que peut émettre un contrôleur local concernant les deux listes ACL et RL sont la demande d'accès d'un nœud et/ou la demande d'ajout ou de suppression d'une entrée dans une liste. Ces requêtes sont envoyées vers le groupe des contrôleurs locaux GCL, chiffrées avec leur clé locale KEK_CCL.

Dans ce qui suit et pour des raisons de simplification, nous considérons que tout contrôleur local a le droit de consulter ou de modifier les deux listes ACL et RL.

Demande d'autorisation d'accès
$CL_k \Rightarrow GCL : \{ID\_CL_k, mg_{new}\}KEK\_CCL$
$CL_{rk} \Rightarrow CL_k : \{ID\_CL_{rk}, ID\_CL_k, mg_{new}, Acceptation\}KEK\_CCL$

avec : mg_{new} est le nouveau membre voulant joindre le groupe et CL_{rk} est le contrôleur local qui détient l'information recherchée et qui répond à la requête d'autorisation d'accès. Si le nouveau membre n'est pas autorisé à joindre le groupe, i.e : il n'existe pas d'entrée ACL correspondante à ce membre, le contrôleur local en question CL_k refuse sa demande d'adhésion au groupe, après l'expiration d'un délai noté Join_Request_Timer.

Ajouter entrée
$CL_k \Rightarrow GCL : \{ID\_CL_k, mg_{à\_ajouter}, Liste\}KEK\_CCL$

avec : $mg_{à\_ajouter}$: membre à ajouter dans la liste ACL ou RL.

Supprimer entrée
$CL_k \Rightarrow GCL : \{ID\_CL_k, mg_{à\_supprimer}, Liste\}KEK\_CCL$

avec : $mg_{à\_supprimer}$: membre à supprimer de la liste ACL ou RL.

6.3.9 Retransmission de la TEK par les contrôleurs de BALADE

Pour améliorer la fiabilité de la transmission de la TEK et assurer la synchronisation entre la distribution de la clé de chiffrement de données et la diffusion des données chiffrées, nous utilisons les numéros de séquence. Quand un membre se rend compte qu'il ne détient pas la clé de chiffrement de données appropriée, il la demande auprès de son contrôleur local, comme suit :

TEK Retransmission
$mg_{ik} \longrightarrow CL_{ik} : \{ID\_mg_{ik}\}KEK\_CSG_{ik}$
$\forall mg_{ik}, CL_{ik} \Rightarrow mg_{ik} : \{ID\_CL, TEK, Num\_Seq\}KEK\_CSG_{ik}$

Connaissances initiales

$\forall mg_{ik}$	ID_mg_{ik}, KEK_CSG_{ik}
CL_{ik}	$ID\_CL, TEK, Num\_Seq, KEK\_CSG_{ik}$

6.3.10 Clusterisation avec OMCT

Nous utilisons l'algorithme OMCT [BCF05] de clusterisation dynamique pour diviser dynamiquement le groupe multicast en clusters et pour élire les contrôleurs locaux de BALADE, selon leurs localisations géographiques par rapport aux autres membres du groupe.

Clusterisation à l'initialisation

À l'initialisation du groupe multicast, tous les membres sont attachés au contrôleur global CG, qui vérifie périodiquement si son groupe est fortement corrélé et par conséquent si le processus de distribution de clés est optimal. La période de vérification de la cohésion au sein d'un cluster est notée Cohésion_Timer. L'évaluation de la cohésion du groupe multicast est déterminée en calculant le facteur de centralisation des membres autour du nœud central CG :

$$Cohésion = \frac{membres\ dans\ la\ portée\ du\ CG}{Nombre\ de\ membres\ du\ groupe} = \frac{Voisins\ à\ un\ seul\ saut\ du\ CG}{Nombre\ de\ membres\ du\ groupe}$$

Ce paramètre mesure la proximité des membres du groupe par rapport à leur CG. En dessous du seuil minimal $Min\_Cohésion$, la cohésion est considérée perdue et la clusterisation du groupe multicast est entamée par le CG.

Clusterisation
$\forall mg_{0k} \in LML\_CG_k$ $CG_k \Rightarrow\ mg_{0k} : \{Clusterisation\} KEK\_CSG_{0k}$ $mg_{0k} \longrightarrow CG_k : \{Id\_mg_{0k}, Coord\_mg_{0k}\} Pub\_CG_k$
Exécution de l'algorithme OMCT *Election des nouveaux contrôleurs locaux* CL_{ik}
$\forall i$ $CG_k \longrightarrow CL_{ik} : \{LML\_CL_{ik}, KEK\_CCL\} Pub\_CL_{ik}$ $CL_{ik} \longrightarrow mg_{ik} : \{ID_{Cluster}, KEK\_CSG_{ik}, Pub\_CL_{ik},$ $imprint, \{CBID - CL_{ik}\} Pri\_CL_{ik}\} Pub\_mg_{ik}$

Le CG commence par envoyer un message en multicast à tous les membres de son sous-groupe, pour entamer la clusterisation selon OMCT. Ces membres lui envoient un message contenant leur identité et leurs coordonnées, crypté avec sa clé publique. À cette étape, le contrôleur global CG exécute l'algorithme OMCT qui sélectionne les nouveaux contrôleurs locaux CL_{ik}, qui doivent former le groupe des contrôleurs locaux et partager la clé de groupe KEK_CCL.

Le CG envoie à tout nouveau contrôleur local CL_{ik}, un message contenant sa liste de membre locaux LML_CL_{ik}, ainsi que la clé partagée par le groupe des contrôleurs locaux KEK_CCL. Ce message est chiffré avec les clés publiques des contrôleurs locaux.

Afin que chaque membre du groupe mg_k sache à quel cluster il appartient (et devient ainsi mg_{ik}), tout nouveau contrôleur local CL_{ik} envoie à tous les membres de sa liste LML_CL_{ik}, un message contenant l'identificateur de son cluster $ID_{Cluster}$ et la clé locale de chiffrement de clé KEK_CSG_{ik}. La clé publique Pub_CL_{ik}, l'entier *imprint* et le CBID du CL_{ik} permettent d'assurer son authentification. Ce message est chiffré avec la clé publique de chaque membre du cluster respectivement.

Connaissances initiales

CG_k	KEK_CSG_{0k}
$\forall mg_{ik}$	$Id\_mg_{0k}, Coord\_mg_{0k}, Pub\_CG_k$

Connaissances après l'exécution de OMCT

CG_k	$LML\_CL_{ik}, KEK\_CCL, Pub\_CL_{ik}$
$\forall CL_{ik}$	$ID_{Cluster}, KEK\_CSG_{ik}, Pub\_CL_{ik}, Pri\_CL_{ik}, imprint, \forall mg_{ik} : Pub\_mg_{ik}$

À la création des nouveaux clusters dans le groupe multicast, les nouveaux contrôleurs locaux deviennent responsables de la gestion des clés locales et de leurs distributions à leurs membres locaux respectifs. En outre, ces nouveaux CLs seront responsables d'assurer la maintenance de leurs clusters fortement corrélés. Ainsi, récursivement et à tout instant de la vie de la session multicast, le groupe multicast reste composé de clusters fortement corrélés, assurant que leur cohésion respective est toujours supérieure au seuil minimal de cohésion $Min\_Cohésion$.

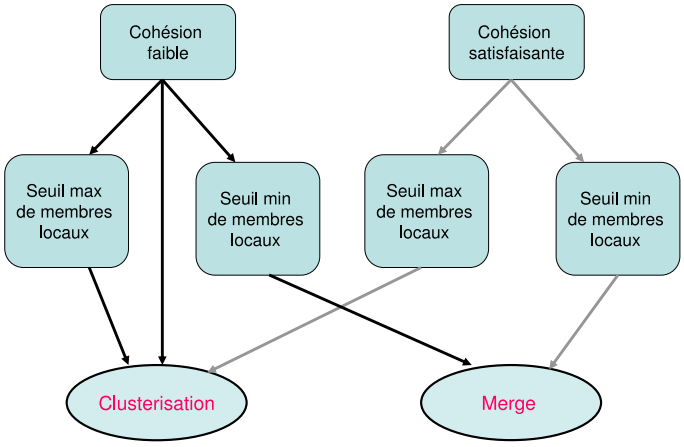


FIG. 6.1 – Formation des clusters dans BALADE

La maintenance de la formation de clusters fortement corrélés est assurée au cours de la session d'un groupe multicast de la façon suivante. Périodiquement, tout contrôleur local calcule le facteur de cohésion au sein de son cluster et le nombre de ses membres locaux. Si le paramètre de cohésion atteint le seuil minimum $Min\_Cohésion$, **ou** si le nombre de membres locaux atteint le seuil maximum (MAX_Threshold), le contrôleur local entame la clusterisation de son sous-groupe.
Si le nombre de membres locaux atteint le seuil minimal (MIN_Threshold), le contrôleur local décide de fusionner son cluster avec d'autres clusters (cf. Figure 6.1) et ainsi de demander à ses

membres locaux de joindre d'autres clusters, dans une durée maximale notée Merge_Timer, comme suit :

Merge
$\forall mg_{ik}, CL_{ik} \Rightarrow mg_{ik} : \{ID\_Cluster, LML\_CL_{ik}\}KEK\_CSG_{ik}$

Cette procédure de Merge est aussi déclenchée lorsqu'un contrôleur local veut notifier ses membres locaux de son départ du groupe multicast.

6.4 Conclusion

Dans ce chapitre, nous avons établi une spécification semi-formelle de notre protocole de gestion de clé de groupe BALADE, couplé avec l'algorithme de clusterisation dynamique OMCT. Cette procédure nous a permis d'identifier les différents sous-protocoles de BALADE, les entités participantes à ces sous-protocoles, les messages échangées entre elles, ainsi que leurs connaissances statiques et dynamiques. Cette phase de spécification semi-formelle de BALADE est également indispensable pour pouvoir entamer sa validation, à travers trois approches complémentaires : la vérification formelle du protocole, son implantation au sein d'une application réelle et l'évaluation de ses performances.

Troisième partie

Validation

Chapitre 7

Vérification formelle de BALADE

Sommaire

7.1 Introduction

La complexité inhérente des protocoles de gestion de clé de groupe tels que BALADE justifie le recours à des techniques de modélisation et de validation formelle, permettant de vérifier les propriétés de sécurité requises pour le bon fonctionnement de ces protocoles.

Pour vérifier la robustesse et l'applicabilité de notre protocole de gestion de clé de groupe BALADE, couplé avec OMCT, nous avons entrepris sa validation, afin de détecter les éventuelles failles de sécurité qu'un intrus peut exploiter pour pirater des données ou compromettre des membres du groupe multicast. Notre approche de validation se base sur l'outil d'analyse des protocoles de sécurité AVISPA [ABB+05]. Cet outil se distingue des outils existants par les nouvelles fonctionnalités qu'il intègre. Tout d'abord, il propose une analyse entièrement automatique. Ensuite, il recouvre un large spectre de protocoles et peut notamment prendre en compte certaines propriétés des opérateurs cryptographiques.

La vérification du protocole BALADE avec AVISPA suit les deux étapes suivantes [BCC+06] :

1. **Spécifier des scénarii des sous-protocoles de BALADE en HLPSL**

   Dans cette étape, nous spécifions les différents sous-protocoles identifiés dans le chapitre 6, à l'aide du langage de spécification HLPSL [CCC+04]. Ce langage fournit un haut niveau d'abstraction et offre plusieurs interfaces requises pour la spécification de protocoles de sécurité à large déploiement dans Internet. Ce langage est basé sur la notion de rôles : rôles basiques représentant tout participant au protocole et rôles composés représentant des scénarii entre les rôles basiques. Les rôles sont indépendants les un des autres. Ils ont des paramètres d'entrée qui représentent des informations initiales et communiquent ensemble à travers des canaux.
   Ensuite, la spécification HLPSL est traduite automatiquement en un langage de bas niveau, nommé IF (Format Intermédiaire), via un traducteur HLPSL2IF. La sortie de ce traducteur correspond à l'entrée de l'outil AVISPA.

2. **Vérifier les spécifications avec l'outil AVISPA (CL-ATSE)**

   Cette dernière étape consiste à utiliser l'outil AVISPA pour vérifier les différents sous-protocoles de BALADE. AVISPA combine plusieurs techniques de vérification de protocoles. Ces différentes techniques reposent soit sur la vérification de modèles, soit sur la résolution de contraintes extraites par exploration «symbolique» des protocoles. Avec la méthode de résolution de contraintes il s'agit de tester si l'intrus peut déchiffrer un message avec les informations qu'il a en sa possession, ou qu'il peut acquérir en se faisant passer pour un autre agent.

7.2 Spécification des sous-protocoles de BALADE en HLPSL

Dans cette section, nous présentons dans un premier temps le langage de spécification formelle HLPSL. Nous détaillons ensuite la spécification de deux scénarii des sous-protocoles de BALADE : initialisation du protocole et ré-intégration d'un membre du groupe. L'ensemble des spécifications des autres sous-protocoles de BALADE en HLPSL est fourni dans l'annexe B.

7.2.1 Présentation du langage HLPSL

HLPSL (*High-Level Protocol Specification Langage*) [Tea05] est un langage de spécification modulaire, basé sur la notion de rôles (participants) et de rôles composés (sessions, instances). Un rôle simple sert à décrire les actions d'un agent lors de l'exécution d'un protocole ou d'un sous-protocole. Un rôle composé instancie plusieurs rôles simples afin de modéliser l'exécution du protocole en entier. Le rôle environnement définit les agents concrets, les sessions d'exécution, ainsi que les connaissances de l'intrus et autres entités globales. Il est à noter que le langage HLPSL ne permet pas de spécifier un nombre infini d'entités, mais des instances spécifiques du protocole, représentant des scénarii bien définis. Cependant, grâce à la notion de rôles de HLPSL, il est possible de faire varier le nombre de membres du groupe. En effet, pour le protocole BALADE, les membres d'un même cluster ont le même comportement concernant la réception et l'émission de messages. Il est ainsi suffisant de définir un rôle représentatif pour chaque cluster.

En plus de la notion de rôle fournie par HLPSL, ce langage a les caractéristiques suivantes :

- support cryptographique varié (clés symétriques, clés publiques et privées, fonctions de hachage, ...),
- information typée (ou non), avec des types simples ou composés,
- propriétés algébriques supportées (concaténation, OU exclusif, exponentiation, ...),
- canaux pour les échanges de messages,
- flux de contrôle assurant des transitions valides,
- propriétés étudiées : confidentialité, authentification forte ou faible.

Les canaux de communications sont de type Dolev et Yao noté (DY) [DY83]. Ce modèle de communication consiste en premier lieu à prendre en compte la quasi-omnipotence des intrus. Un intrus peut espionner et dérouter toutes les lignes de communication, toute communication d'un message M entre deux entités peut être simulée par un envoi de M à l'intrus, suivi d'un envoi de M par l'intrus au destinataire souhaité. Comme l'intrus peut simuler n'importe quel envoi et réception de message, le modèle de communication DY suppose que tout message envoyé est envoyé à l'intrus et tout message reçu l'est de l'intrus.
En second lieu, le modèle de Dolev et Yao suppose que les messages envoyés et reçus ne sont ni des nombres ni des suites de bits, mais des éléments d'une algèbre de termes, éventuellement modulo une théorie équationnelle. Autrement dit, toute tentative de déchiffrer le message M chiffré par la clé k ($\{M\}_K$) en utilisant une clé K' différente de K fournit un terme prouvablement différent de M.
Troisièmement et finalement, le modèle de Dolev et Yao représente l'intrus comme un système déductif, qui peut produire des messages en utilisant des règles en aussi grand nombre et aussi souvent qu'il veut. Par contre, l'intrus ne peut pas produire de messages d'aucune autre façon. Notamment, l'intrus ne peut pas produire M à partir de $\{M\}_K$, à moins de ne pouvoir produire la clé K ou de ne pouvoir produire M sans partir de $\{M\}_K$.

7.2.2 Spécification du sous-protocole "Initialisation de BALADE"

Ce sous-protocole fait intervenir le contrôleur global CG et les membres du groupe mg. Le contrôleur global génère une nouvelle clé de chiffrement de données TEK, une nouvelle clé locale KEK_CSG_{0k} et les envoie au membre mg, chiffrées avec sa clé publique comme illustré ci-dessous (cf. Chapitre 6) :

TEK Distribution
$\forall mg_k,\ CG_k \longrightarrow mg_k : \{TEK, Num\_Seq, KEK\_CSG_{0k}, IDG, ID_{CG}, Pub\_CG,$ $imprint, \{CBID - CG\}Pri\_CG\}Pub\_mg_k$

Les propriétés de sécurité que doit vérifier AVISPA sont la confidentialité des deux clés partagées par les deux entités (clé locale et clé de chiffrement de données).

La première étape pour spécifier ce sous-protocole consiste à identifier le scénario correspondant et les rôles qui y participent. Le premier rôle qui entre en scène dans ce scénario est le contrôleur global CG. Ensuite, sachant que tous les membres du groupe se comportent de la même façon pour l'envoi et la réception des messages et sachant que le langage HLPSL ne permet pas de spécifier un nombre infini d'entités, nous représentons les membres du groupe par un seul rôle, noté *mg*. Deux rôles supplémentaires doivent également être définis : le rôle session regroupant les deux entités de la spécification (CG et *mg*) et le rôle environnement spécifiant les canaux de communications et les connaissances de l'intrus.

```
%% Premier Rôle : Le contrôleur global CG

role membre1 (CG,mg: agent,
                IDG: text,
                IDCG: text,
                Pub-CG: public_key,
                imprint: nat,
                CBID-CG: text,
                Pub-mg: public_key,
                Snd, Rcv: channel(dy))  %% Connaissances initiales du CG
    played_by CG def=

  local State: nat,
       TEK: symmetric_key,
       NumSeq: nat,
       KEK_CSG_0K: symmetric_key  %% Variables locales du CG

  const id1: protocol_id,
        id2: protocol_id  %% Constantes
  init State:=0
```

Les paramètres d'entrée du rôle CG sont ses connaissances initiales, les agents avec lesquels il communique ainsi que les canaux de communications Snd et Rcv qu'il utilise. Ensuite, les variables locales du CG sont listées. Elles correspondent aux variables créés au cours de la session (clé de chiffrement de données, clé de chiffrement de clés et numéro de séquence). Les constantes déclarées sont les identificateurs des propriétés de sécurité à valider.

```
  transition
```

```
     step1. State=0 /\ Rcv(start)
       =|> TEK' := new()      %% génération d'une nouvelle clé TEK
         /\ KEK_CSG_OK' := new()
         /\ NumSeq' := new()       %% Incrémentation du numéro de séquence
         %% Envoi du message de distribution de la TEK
   /\ Snd({TEK'.NumSeq.KEK_CSG_OK'.IDG.IDCG.Pub-CG.imprint.
          {CBID-CG}_inv(Pub-CG)}_Pub-mg)
         /\ State':=1
         %% Propriétés à vérifier par l'outil AVISPA
         /\ secret(TEK',id1,{CG,mg})
         /\ secret(KEK_CSG_OK',id2,{CG,mg})

end role
```

Le CG commence par générer une nouvelle clé TEK, une nouvelle clé locale de son cluster et incrémente son numéro de séquence. Il envoie ensuite le message de distribution de clé au membre du groupe.

```
%% Deuxième Rôle : mg

role membre2 (CG,mg: agent,
           Snd, Rcv: channel(dy))
     played_by mg def=

   local State: nat,
         TEK: message,
         NumSeq: message,
         KEK_CSG_OK: message,
         IDG: message,
         IDCG: message,
         Pub-CG: message,
         imprint: message,
         CBID-CG: message,
         Pub-mg: public_key

    const id1: protocol_id,
          id2: protocol_id
    init State:=0


   transition
     step1. State=0 /\ Rcv({TEK'.NumSeq'.KEK_CSG_OK'.IDG'.IDCG'.Pub-CG'.imprint'.
                {CBID-CG}_inv(Pub-CG)}_Pub-mg)
       =|> State'=1
           /\ secret(TEK',id1,{CG,mg})
```

```
           /\ secret(KEK_CSG_0K',id2,{CG,mg})

%% Ce participant (membre du groupe) n'accepte les clés TEK et KEK_CSG_0K que si
   le CBID est validé authentique

end role
```

Le deuxième rôle est un membre du groupe appartenant au cluster 0 géré par le CG. Ce rôle est représentatif de tous les membres du groupe, qui ont le même comportement à la réception et à l'émission des messages.

```
%%The role session between the two participants

role Initialisation_TEK_Distribution(SC, RC: channel(dy),
                CG, mg: agent,
                TEK: symmetric_key,
                NumSeq: nat,
                KEK_CSG_0K: symmetric_key,
                IDG: text,
                IDCG: text,
                Pub-CG: public_key,
                imprint: nat,
                CBID-CG: text,
                Pub-mg: public_key
           ) def=

     composition
     membre1(IDG,IDCG,Pub-CG,imprint,CBID-CG,Pub-mg,SC,RC)
     /\ membre2(CG,mg,SC,RC)

end role
```

Le rôle session est un rôle composé de HLPSL. Il fait interagir les deux rôles contrôleur global et membre du groupe. Les paramètres d'entrée du rôle session sont les paramètres d'entrées des deux rôles basiques.

```
%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const CG, mg: agent,
         IDG: text,
         IDCG: text,
```

```
        Pub-CG: public_key,
        imprint: nat,
        CBID-CG: text,
        Pub-mg: public_key

   intruder_knowledge = {CG,mg}

   composition
      Initialisation_TEK_Distribution(Snd,Rcv,CG,mg,IDG,IDCG,Pub-CG,
                                      imprint,CBID-CG,Pub-mg)

end role

%% L'objectif de la validation avec AVISPA est la vérification des propriétés de
%% sécurité présentées dans la section goal.
goal
      secrecy_of id1
      secrecy_of id2
end goal

environment()
```

Le rôle principal de HLPSL est le rôle environnement. Ce rôle spécifie tous les agents participants en appelant le rôle session, les canaux de communications à utiliser, ainsi que les connaissances de l'intrus. Généralement, l'intrus connaît les identités des entités participantes au protocole.

L'objectif de la validation avec AVISPA est la vérification des propriétés de sécurité présentées dans la section `goal`. Pour le sous-protocole d'initialisation de BALADE, les propriétés à vérifier sont la confidentialité de la clé de chiffrement de données et celle de la clé de chiffrement de clés.

7.2.3 Spécification du sous-protocole "Ré-intégration d'un membre du groupe"

La spécification du scénario de ré-intégration d'un membre d'un groupe, que nous présentons dans ce qui suit, correspond à la nouvelle version du sous-protocole, décrite dans la section 6.3.4. Un ancien membre du groupe (noté $ancien\_mg_k$) voulant le rejoindre, envoie un message à un membre actuel du groupe (mg_{ik}), contenant sa clé publique, son CBID, l'entier imprint qu'il a utilisé pour générer son CBID et le ticket de ré-authentification chiffré avec la clé du groupe TEK et la clé publique du destinataire. En cas de réussite de l'authentification de l'ancien membre du groupe, (mg_{ik}) lui envoie la clé locale du cluster (KEK_CSSG_{ik}), chiffrée avec sa clé publique.

Re – authentication Procedure
$ancien\_mg_k \longrightarrow mg_{ik} : Pub\_ancien\_mg_k, \{CBID\_ancien\_mg_k\}Pri\_ancien\_mg_k,$ $Imprint,\ \{\{Passwd\}TEK\}Pub\_mg_{ik}$
$mg_{ik} \longrightarrow ancien\_mg_k : \{KEK\_CSG_{ik}\}Pub\_ancien\_mg_k$

L'identification des rôles participants à un scénario de ré-authentification aboutit à un rôle représentant un ancien membre (Amgk) et un rôle représentant un membre du groupe (Mgik). De la même façon que la spécification du sous-protocole d'initialisation de BALADE, deux rôles doivent être ajoutées à la spécification de HLPSL : le rôle session et le rôle environnement. Les propriétés de sécurité que nous vérifions avec AVISPA sont la confidentialité de la clé locale du cluster KEK_CSSG_{ik} ainsi que du ticket de ré-authentification chiffré avec la clé du groupe $\{Passwd\}TEK$.

```
%% Reintegration

%% First Role: Amgk ancien membre voulant ré-intégrer son groupe

role membre1 (Amgk,Mgik: agent,
              PubAmgk,PubMgik: public_key,
              CBIDAmgk,Imp: text,
              TEK: symmetric_key,
              Passwd: text,
              Snd, Rcv: channel(dy))
   played_by Amgk def=

   local State: nat,
         KEKCSGik2: message

   const id1,id2: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd(PubAmgk.Imp.{CBIDAmgk}_inv(PubAmgk).{{Passwd}_TEK}_PubMgik)
           /\ secret({Passwd}_TEK,id1,{Amgk,Mgik})
           /\ State':=1
     step2. State=1 /\ Rcv({KEKCSGik2'}_PubAmgk)
       =|> State':=2
           /\ secret(KEKCSGik2,id2,{Amgk,Mgik})

end role

%% The second role: Mgik membre du groupe multicast
```

```
role membre2 (Amgk,Mgik: agent,
              PubMgik: public_key,
              KEKCSGik,TEK: symmetric_key,
              Snd, Rcv: channel(dy))
   played_by Mgik def=

   local State: nat,
         PubAmgk2: public_key,
         CBIDAMg: message,
         Passwd2: message,
         Imp2: message

    const id1,id2: protocol_id
    init State:=0

   transition
     step1. State=0 /\ Rcv(PubAmgk2'.Imp2'.{CBIDAMg'}_inv(PubAmgk2').
   {{Passwd2'}_TEK}_PubMgik)
       =|> State'=1
           /\ Snd ({KEKCSGik}_PubAmgk2')
           /\ secret({Passwd2'}_TEK,id1,{Amgk,Mgik})
        /\ secret(KEKCSGik,id2,{Amgk,Mgik})

end role

%%The role session between the two participants

role integrer(SC, RC: channel(dy),
          Amgk,Mgik: agent,
          PubAmgk,PubMgik: public_key,
          CBIDAmgk,Imp: text,
          TEK,KEKCSGik: symmetric_key,
          Passwd: text
    ) def=

   composition
     membre1(Amgk,Mgik,PubAmgk,PubMgik,CBIDAmgk,Imp,TEK,Passwd,SC,RC)
     /\ membre2(Amgk,Mgik,PubMgik,KEKCSGik,TEK,SC,RC)

end role

%%The main role

role environment() def=
   local  Snd, Rcv: channel(dy)

   const amgk,mgik: agent,
```

```
         pubamgk,pubmgik: public_key,
         cbidamgk,imp: text,
         tek,kekcsgik: symmetric_key,
         passwd: text

   intruder_knowledge = {amgk,mgik}

   composition
      integrer(Snd,Rcv,amgk,mgik,pubamgk,pubmgik,cbidamgk,imp,tek,kekcsgik,passwd)

end role

goal
      secrecy_of id1
      secrecy_of id2
end goal

environment()
```

7.3 Vérification et validation du protocole BALADE à l'aide de l'outil d'analyse AVISPA

7.3.1 Présentation de l'outil AVISPA

AVISPA[16] (*Automated Validation of Internet Security Protocols and Applications*) [Tea05] est un analyseur de protocoles de sécurité spécifiés en HLPSL, intégrant différentes techniques d'analyse automatique de protocoles, pour un nombre fini de sessions. L'interaction avec l'utilisateur est facilitée par emacs et une interface Web.

Le traducteur CL-ATSE de AVISPA

L'outil AVISPA comporte quatre traducteurs (ou *backends*) schématisés dans la figure 7.1. Pour notre validation, nous utilisons le traducteur CL-ATSE (*Constraint-Logic-based ATtack-SEarcher*). CL-ATSE fournit une traduction des spécifications HLPSL, en un ensemble de contraintes, qui peuvent être efficacement utilisées pour trouver des attaques malicieuses. Les opérations de traduction et de vérification sont entièrement automatiques. La validation avec CL-ATSE se fait donc de la manière suivante :

1. chaque rôle est partiellement pré-exécuté par CL-ATSE, pour extraire la liste de contraintes minimale, le modélisant.
2. les états et les connaissances des rôles participants sont éliminés, grâce à l'utilisation des variables globales.
3. toute étape d'un protocole est exécutée en ajoutant des contraintes nouvelles et en éliminant d'autres contraintes correspondantes.

[16] http ://www.avispa-project.org/

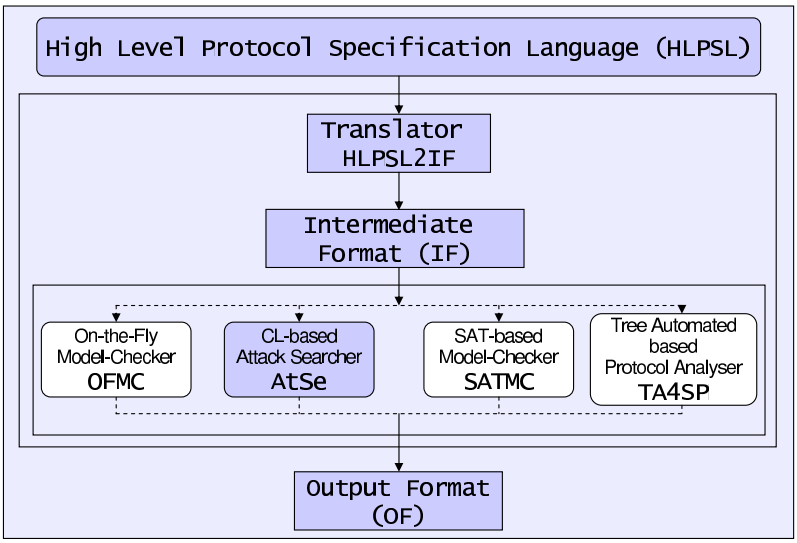


FIG. 7.1 – Le back-end CL-ATSE de AVISPA

4. finalement, toute étape du système est testée pour vérifier les propriétés de sécurité fournies en objectif.

Sortie de AVISPA

Quand l'analyse d'un protocole est réussie, en trouvant ou pas des attaques possibles, la sortie de AVISPA décrit précisément le résultat et sous quelles conditions il a été obtenu :

- la première section *SUMMARY* indique si le protocole est sécurisé ou non, ou si l'analyse n'a pas été concluante,
- la deuxième section *DETAILS* décrit sous quelles conditions le protocole est déclaré sécurisé ou non, sous quelles conditions une attaque est trouvée et finalement pourquoi l'analyse n'a pas été concluante,
- la section *PROTOCOL* rappelle le nom du protocole analysé,
- la section *GOAL* présente le but de l'analyse, comme par exemple la confidentialité de la clé de chiffrement des données,
- la section *BACKEND* désigne le traducteur des spécifications HLPSL, utilisé lors de l'analyse. Nous utilisons pour tous nos analyses le traducteur CL-ATSE de AVISPA.

7.3.2 Validation des sous-protocoles de BALADE avec AVISPA

Dans cette section, nous présentons deux exemples de validation des sous-protocoles de BALADE, initialisation du protocole et ré-intégration d'un membre du groupe. L'ensemble des autres sous-protocoles de BALADE ont été validé avec succès via AVISPA.

Initialisation de BALADE

```
SUMMARY
```

```
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfileSDnRd2.if

GOAL
  As Specified

BACKEND
  CL-AtSe
```

Ré-intégration d'un membre du groupe

Nous avons spécifié dans un premier temps le sous-protocole de ré-intégration comme suit :

Re – authentication Procedure
$ancien\_mg_k \longrightarrow mg_{ik} : Pub\_ancien\_mg_k, CBID\_ancien\_mg_k,$ $\{\{Passwd\}TEK\}Pri\_ancien\_mg_k$
$mg_{ik} \longrightarrow ancien\_mg_k : \{KEK\_CSG_{ik}\}Pub\_ancien\_mg_k$

Cette première version n'est pas valide, puisqu'elle permet à un intrus de pouvoir intercepter {Passwd}TEK et par suite pouvoir récupérer la clé locale du sous-groupe KEK_CSG_{ik} et la clé de chiffrement de données TEK. Cette faille a été détectée par l'outil de validation AVISPA. La figure 7.2 illustre la trace de cette faille, générée par AVISPA. L'intrus étant l'agent i.

Le résultat de la vérification du sous-protocole de ré-intégration par AVISPA est le suivant :

```
SUMMARY
  UNSAFE

DETAILS
  ATTACK_FOUND
  TYPED_MODEL

PROTOCOL
  /users/madyne/bouassid/AVISPA/avispa-1.0//testsuite/results/Reintegration.if

GOAL
  Secrecy attack on ({passwd}_tek)
```

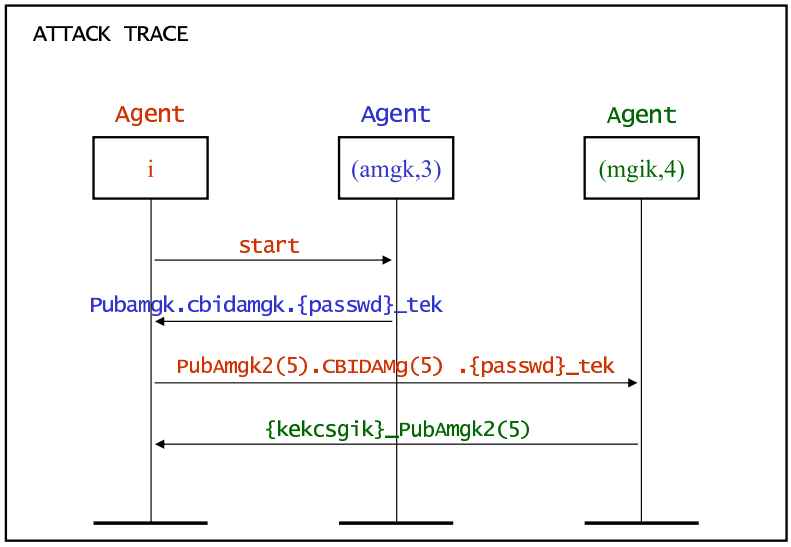


FIG. 7.2 – Trace de la faille de sécurité détectée par AVISPA

```
BACKEND
  CL-AtSe

ATTACK TRACE

  i -> (amgk,3):  start
  (amgk,3) -> i:  pubamgk.cbidamgk.{{passwd}_tek}_(inv(pubamgk))
                  & Secret({passwd}_tek,set_60);  Add amgk to set_60;
                  & Add mgik to set_60;
```

Pour faire face à cette attaque, nous assurons l'authentification et la non répudiation de l'ancien membre voulant joindre le groupe dans la nouvelle version de ce sous-protocole, via son CBID. Le mot de passe chiffré avec la clé du groupe TEK est chiffré avec la clé publique du membre du groupe, pour que lui seul puisse obtenir {Passwd}_TEK et non un intrus. Le nouveau sous-protocole (présenté dans la section 6.3.4) est validé comme suit :

```
SUMMARY
  SAFE

DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL

PROTOCOL
  /users/madyne/bouassid/AVISPA/avispa-1.0//testsuite/results/Reintegration.if
```

```
GOAL
  As Specified

BACKEND
  CL-AtSe
```

7.4 Conclusion

Dans ce chapitre, nous avons présenté notre approche pour la validation de notre protocole de gestion de clé de groupe BALADE, en utilisant l'outil AVISPA. Dans une première étape, nous avons spécifié des différents sous-protocoles de BALADE couplé avec OMCT, en utilisant le langage de spécification HLPSL. Ensuite, nous avons validé ces scénarii avec AVISPA, qui procède à une recherche des attaques que peut faire un intrus, pour mettre en péril le service de sécurité que nous avons défini pour chaque sous-protocole, tel que la confidentialité de la clé de chiffrement de données ou des clés de chiffrement de clés.

Notre approche de validation nous a permis de détecter une faille de sécurité au niveau du sous-protocole de ré-intégration d'un ancien membre du groupe, permettant à un intrus de pouvoir intercepter et récupérer la clé de chiffrement de données. Nous avons remédié à cette faille ; la nouvelle version de ce sous-protocole est désormais valide selon AVISPA. La validation avec AVISPA montre toutefois les limites suivantes :

- Le temps n'est pas pris en compte dans la spécification HLPSL.
- Les canaux de communication sont supposés fiables dans AVISPA. Cependant, le taux de perte de paquets dans l'environnement sans fil des MANETs n'est pas négligeable.
- Les résultats de la vérification avec AVISPA sont influencés par deux principales contraintes. D'abord, les scénarii des sous-protocoles validés par AVISPA correspondent à des instances particulières de ces sous-protocoles, mettant en interaction un nombre fini de rôles. L'infinité du nombre de rôles n'est pour le moment pas supportée par AVISPA. Ensuite, chaque sous-protocole est vérifié indépendamment des autres sous-protocoles ; tandis ce que le risque d'attaques de sécurité provenant de l'interaction de tous les sous-protocoles n'est pas nul.

Après avoir réalisé une validation qualitative de notre protocole avec AVISPA, nous l'avons implanté et avons validé son applicabilité dans le cadre d'une application de Jukebox coopératif. Nous présentons l'implantation de BALADE dans le chapitre suivant.

Chapitre 8

Implantation de BALADE

Sommaire

8.1 Introduction

L'application cible que nous avons choisie pour l'implantation de BALADE est une application de jukebox : une diffusion multicast de titres MP3 pour un ensemble de nœuds ad hoc. Chaque nœud ayant une liste de titres MP3, la déclare au sein du réseau ; une file d'attente de titres MP3 est ainsi distribuée à tous les nœuds. Plusieurs nœuds du réseau peuvent jouer le rôle de source du groupe multicast. Nous avons donc une émission multicast 1 vers n séquentiel. L'émission des titres MP3 doit être sécurisée, ainsi seuls les membres du groupe seront capables de déchiffrer les données.

Ce chapitre suit le plan suivant : la section 2 décrit l'application de Jukebox coopératif appelée JDukebox. Dans la section 3, nous présentons les détails et les choix d'implantation de BALADE. La section 4 décrit l'interaction de BALADE avec JDukebox. La section 5 conclut ce chapitre.

8.2 Présentation de l'application JDukebox

JDukebox est une application test, qui implante une application de jukebox distribuée et coopérative, diffusant des fichiers MP3 à des utilisateurs dans le réseau. Comme cela est illustré dans la figure 8.1, les utilisateurs d'un même groupe de JDukebox détiennent chacun une liste de titres MP3 qu'ils mettent à disposition de tous les autres membres du groupe.

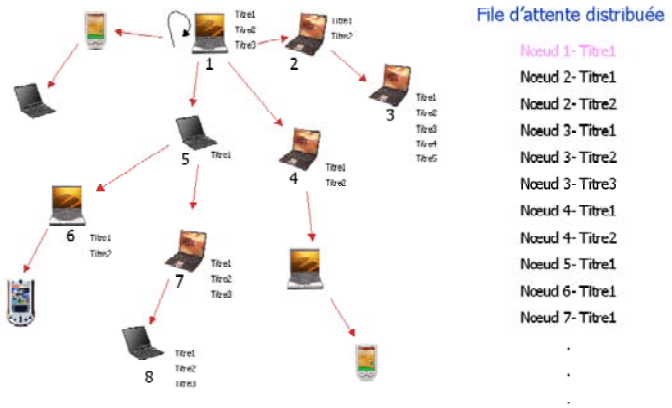


FIG. 8.1 – Jukebox coopératif

Une liste distribuée (*playlist*) est ensuite construite de façon coopérative et distribuée, contenant les titres MP3 que le groupe va écouter successivement, en assurant une transparence de la source effective d'un titre en cours de diffusion.

L'application JDukebox se compose de deux parties essentielles :

- Le démon MJBox[17] est écrit en C. Il est responsable de la diffusion des fichiers MP3 et de la réception des flux RTP. Ce module MJBox est représenté dans la figure 8.3 par la partie plan de données.

[17]Ce module a été réalisé par Abdelkader Lahmadi, Ingénieur expert au sein de l'équipe de recherche MADYNES

– Le corps JDukebox[18] est écrit en java. Il gère le plan de contrôle de l'application, grâce à une couche JXTA (cf. Figure 8.3 Plan de contrôle). JDukebox lance le démon MJBox et communique avec lui par le biais de messages envoyés via une socket. Le module JDukebox, est à son tour, composé de quatre parties :

1. le module *streamers* qui manipule et gère les communications avec le démon MJ-Box ;
2. l'interface *utilisateur* qui fournit les informations de la *playlist* à l'utilisateur et lui permet de former des requêtes sur la *playlist*. La figure 8.2 présente l'interface graphique de la JDukebox.

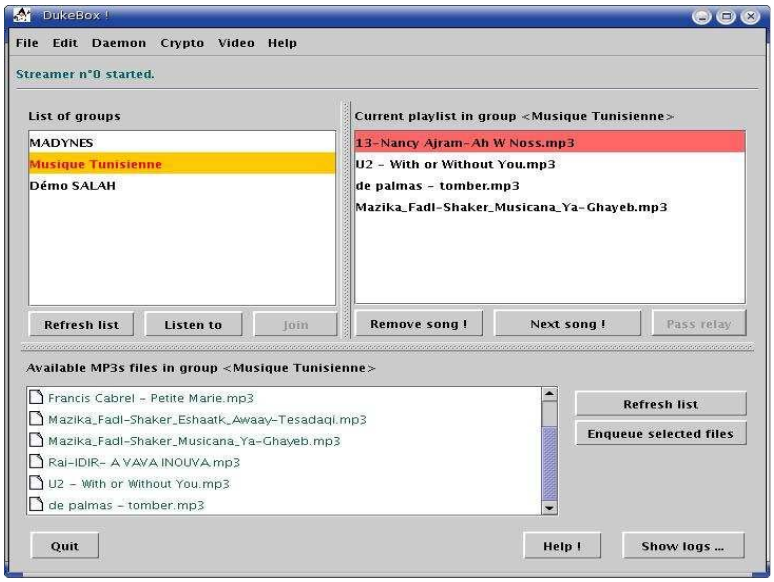


FIG. 8.2 – Imprim-écran JDukebox

3. le module *JXTA* qui gère le contrôle des messages de synchronisation et de la gestion du groupe ;
4. le module *lists* qui gère les algorithmes utilisés pour assurer la maintenance de la consistance de la *playlist*.

8.3 Implantation de BALADE

L'implantation de BALADE est réalisée sous Linux version Mandrake 10.0 Noyau 2.4.x. Le protocole de routage multicast utilisé est MAODV v6.

Le module BALADE supervise et sécurise toutes les diffusions multicast de l'application jukebox. Ce module est constitué des composants suivants :

1. le composant client (MG) qui gère le comportement d'un nœud joignant le groupe ;

[18]Ce module a été réalisé par Adrien Bruneton, dans le cadre de son stage de fin d'études, au sein l'équipe de recherche MADYNES

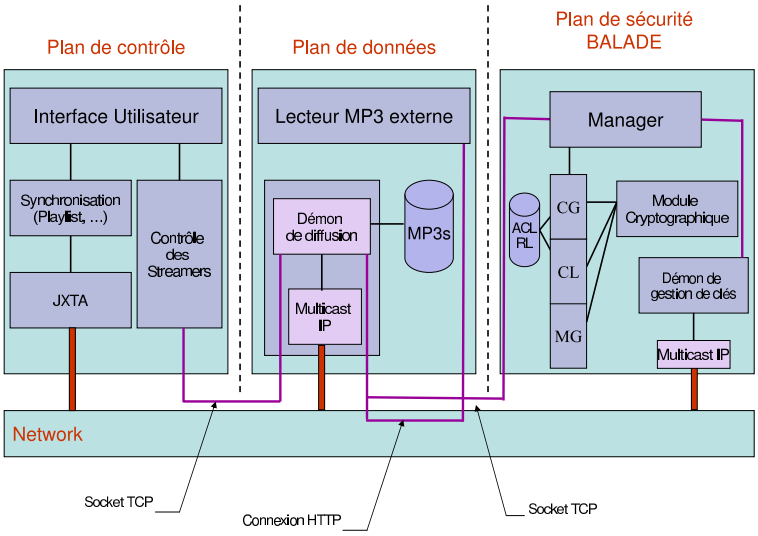


FIG. 8.3 – Architecture fonctionnelle BALADE + JDukebox

2. le composant serveur qui gère le comportement des sources et des contrôleurs de groupe (CG et CLs), ainsi que leur accès aux listes de contrôle d'accès et de révocation (cf. Figure 8.3).
3. le module *manager* qui gère les interactions de BALADE avec Mjbox est responsable de piloter le démon de gestion de clés BALADE.
4. le module *cryptographique* qui rassemble les traitements cryptographiques de BALADE : initialisation des clés privées, publiques et CBIDs, authentification des membres du groupe et chiffrement/déchiffrement des données.
5. le module *démon de gestion de clés* qui constitue le cœur du protocole BALADE. Il est responsable de la génération et de la distribution des messages de clés du côté contrôleurs et de la réception de ces messages du côté membres du groupe.

8.3.1 Identité et entités dans BALADE

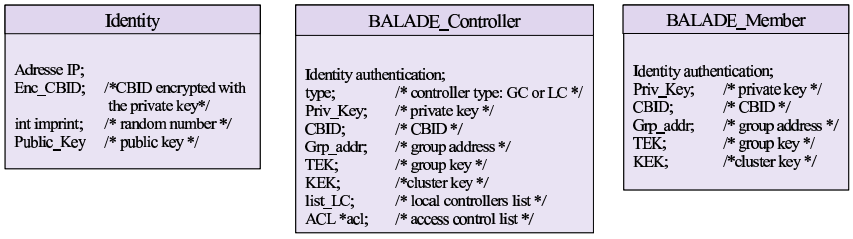


FIG. 8.4 – Structures de données

La Figure 8.4 présente les structures de données utilisées dans BALADE. La structure *Identity* permet d'identifier un nœud dans le réseau et de prouver son authenticité à travers

son CBID et également d'assurer l'authentification de la source de distribution de clés. Cette structure contient les champs suivant :

- l'adresse IP,
- l'entier *imprint* utilisé lors de la génération du CBID,
- la clé publique,
- le CBID, chiffré avec la clé privée du détenteur de l'identité permettant de prouver la liaison cryptographique entre la clé publique et la clé privée.

Les structures correspondantes aux entités dans BALADE sont *BALADE_Controller* et *BALADE_Member* (cf. Figure 8.4). Elles contiennent des champs relatives à l'identité, à l'identifiant du groupe et du cluster et aux clés de chiffrement de données et de chiffrement de clés.

8.3.2 Messages de distribution de clés BALADE

Un message de distribution de clés dans BALADE est encapsulé dans un paquet IP et a la forme présentée dans la figure 8.5. Les champs d'un paquet BALADE sont les suivants :

- Entête BALADE :
  1. Type : identifiant du message (=1 : distribution de la TEK, =2 : distribution de la KEK),
  2. Version : version du protocole BALADE (=1 par défaut),
  3. Réservé : champ réservé pour d'éventuelles modifications futures,
  4. Checksum : contrôle l'intégrité du message,
- Données BALADE :
  1. Clé : correspond à la TEK ou KEK, selon le type du message,
  2. Numéro de séquence : correspond à la clé envoyée,
  3. Identifiant du groupe,
  4. Identifiant du cluster,
  5. Identité de l'émetteur.

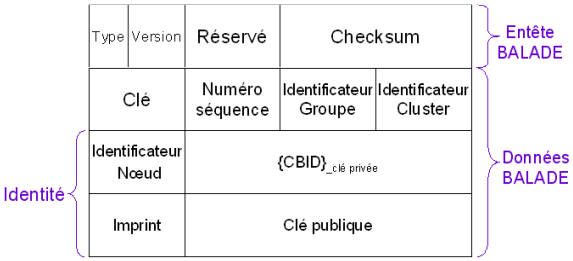


FIG. 8.5 – Paquet BALADE

8.4 Interaction de BALADE avec JDukebox

Le module BALADE interagit avec le module MJBox, via l'envoi de messages de signalisation et de données, à travers une socket TCP, comme cela est illustré dans la figure 8.3. Le manager du module BALADE lance un démon qui reste en écoute des messages de la JDukebox. Ces messages sont définis comme suit :

– message NEW_CONTROLLER : création du contrôleur global du groupe (cf. Figure 8.6).
En recevant l'ordre de commencer la diffusion d'un titre MP3, le module MjBOX envoie un message au module BALADE, lui demandant de créer un nouveau contrôleur de type contrôleur global (CG) (s'il n'existe pas). Ce contrôleur sera responsable de la sécurisation des données envoyées par la source MjBOX. Après avoir été créé, le CG commence par initialiser ses clés publique et privée, ainsi que son CBID. Ensuite, il génère la clé de chiffrement de données TEK et la clé de chiffrement de clé KEK, relative au cluster 0. Le CG charge la liste de contrôle d'accès à partir d'un fichier ACL qu'il détient et qui contient les identités de tous les membres autorisés à joindre le groupe.

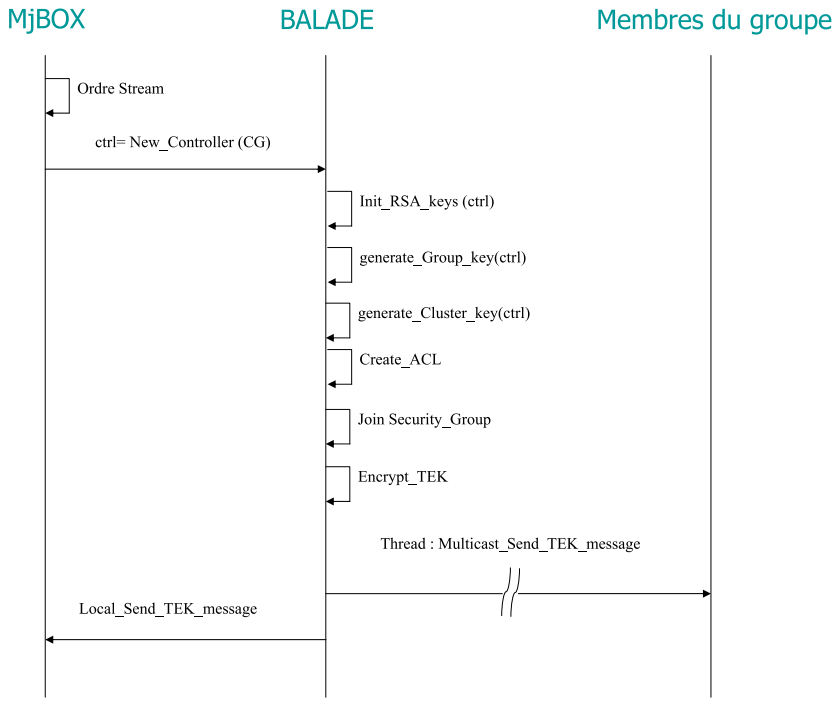


FIG. 8.6 – Message "Nouveau contrôleur"

Le CG entame ensuite la phase de distribution de la clé TEK chiffrée, à tous les membres du groupe multicast. La TEK étant chiffrée avec la clé KEK initiale. Pour cela, un thread *Multicast_Send_TEK_Message* est lancé. Ce thread est responsable de diffuser le message de distribution de la TEK, en multicast, chaque intervalle de temps *Send_TEK_Interval*. Cet envoi continu de la TEK garantira aux membres du groupe la possibilité de recevoir la clé de chiffrement TEK. Finalement, le CG envoie en local, à travers la socket TCP qu'il détient avec MjBOX, la clé TEK à la source du titre MP3, lui permettant de chiffrer son flux et de l'envoyer en multicast à tous les abonnés du groupe de musique. L'algorithme de chiffrement de données adopté dans JDukebox est le RC4, dédié principalement au chiffrement symétrique de flux.
Si le contrôleur global existe déjà, il entamera directement la phase de génération et de distribution de la TEK, aux autres membres du groupe en multicast et au module MjBOX via la socket TCP locale.

– cas NEW_MEMBER : création d'un nouveau membre appartenant au groupe (cf. Figure 8.7).
En recevant un ordre d'écouter un titre MP3, diffusé sur un groupe multicast de musique, le module MjBOX demande au module BALADE de créer un nouveau membre du groupe. Ce membre (MG) commencera par initialiser ses clés publique et privée ainsi que son CBID (chargement à partir de fichiers spécifiques crées au préalable). Ensuite, en cas de succès de l'authentification et du contrôle d'accès, MG joint le groupe multicast. Il pourra ainsi créer un thread pour créer une socket multicast d'écoute, pour recevoir la TEK actuelle de chiffrement du flux. Dès que le MG reçoit cette TEK, il l'envoie au module MjBOX, via la socket locale TCP, pour qu'il puisse commencer à déchiffrer le flux MP3 émis par la source de données.

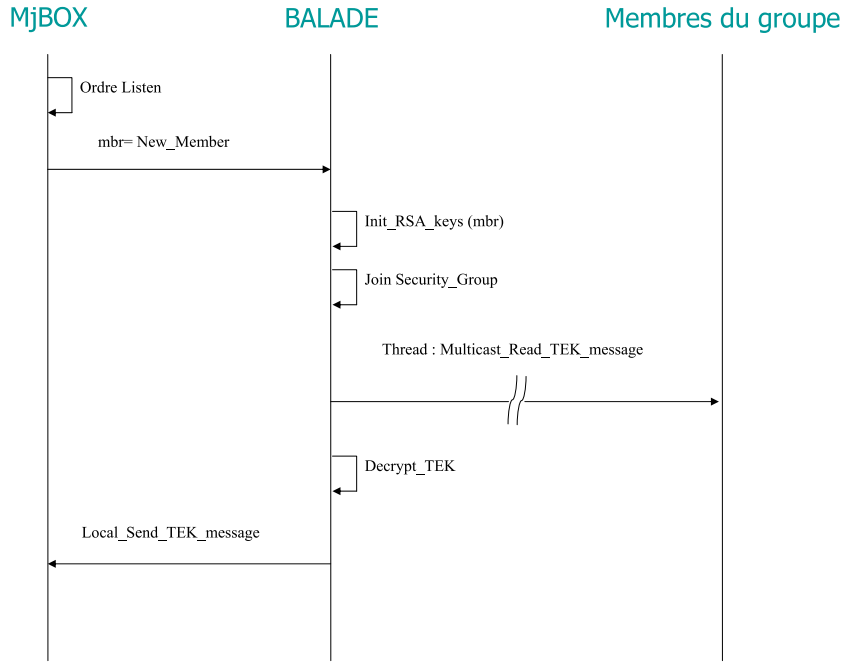


FIG. 8.7 – Message "Nouveau membre"

– Cas TEK_RENEWAL : renouvellement de la clé de chiffrement de données, suite au début de l'émission d'une nouvelle unité sémantique de données (cf. Figure 8.8).
Lors du passage de la diffusion d'un titre MP3 à un autre, la source MjBOX envoie une demande au module BALADE, pour renouveler la clé de chiffrement de données TEK. Le contrôleur global, déjà initialisé, génère alors une nouvelle TEK. Il la chiffre avec la clé KEK_CCL et la distribue au groupe des contrôleurs locaux BALADE. Ensuite, la TEK chiffrée avec la clé KEK_CSG0 est envoyée au cluster local du contrôleur global, via une socket en multicast. Et finalement, la TEK est envoyée en local au module MjBOX, pour pouvoir commencer à diffuser le nouveau titre MP3, chiffré avec la nouvelle clé TEK.
– cas KEK_RENEWAL : renouvellement de la clé de chiffrement de clés, suite à un événement de join ou de leave au sein du groupe (cf. Figure 8.9).

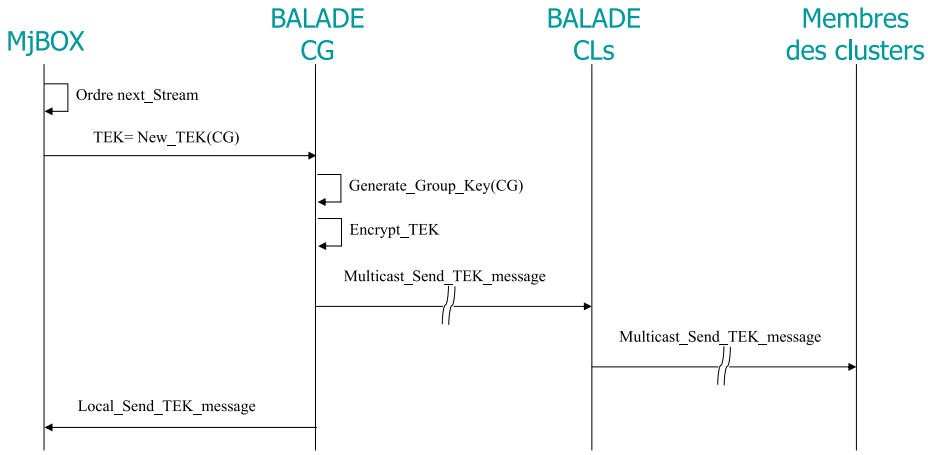


FIG. 8.8 – Message "Renouvellement de la TEK"

Un contrôleur local doit renouveler la clé de chiffrement de clé KEK locale, à chaque événement de Join ou de Leave, survenu au sein de son cluster. Dans un tel cas, le contrôleur local génère une nouvelle clé KEK et la distribue à tous les membres locaux de son cluster.

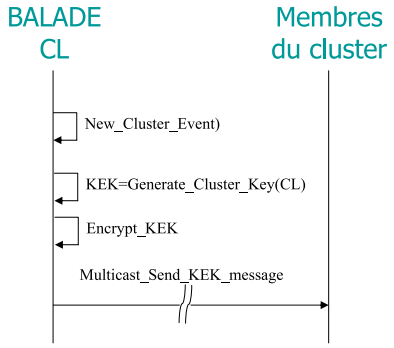


FIG. 8.9 – Message "Renouvellement de la KEK"

8.5 Liste des Timers et des constantes par défaut

Les valeurs des constantes et des timers dépendent des besoins fonctionnels de l'application à sécuriser par BALADE. Dans ce qui suit, nous présentons les valeurs par défaut de ces constantes et timers que nous avons adoptés lors de notre implantation.

1. Merge_Timer : timer permettant aux membres d'un cluster de pouvoir intégrer d'autres clusters, à cause du départ ou du Merge de leur CL. La valeur de ce timer dépend de l'application en question (par défaut une minute).
2. Join_Request_Timer : après l'expiration de ce timer, un contrôleur local considère que sa requête de demande d'adhésion d'un nœud voulant joindre le groupe a été refusée.

La valeur par défaut de cette valeur est 2 minutes.

3. Min_Cohésion : dès que la cohésion d'un cluster atteint ce seuil minimum, son CL décide d'entamer la clusterisation de son sous-groupe. La valeur par défaut de cette valeur est $\frac{1}{2}$.
4. MAX_Threshold : nombre maximum de membres d'un cluster, au delà duquel le cluster est divisé. La valeur par défaut de ce seuil est 20.
5. MIN_Threshold : nombre minimum de membres d'un cluster, au dessous duquel le cluster est fusionné avec d'autres clusters. La valeur par défaut de ce seuil est 3.
6. CL_Query_Interval : période de temps séparant deux envois successifs de message CL_Query. Cette période doit être inférieure au Merge_Timer, afin de permettre à un nœud de choisir le cluster auquel il veut adhérer, avant l'expiration d'un Merge_Timer. La valeur par défaut de cet intervalle est une minute.
7. Cohésion_Timer : période de vérification de la cohésion d'un cluster. La valeur par défaut de cet intervalle est 4mn. Cette valeur est justifiée par le fait que la vitesse maximale des membres est définie à 3m/s (pour des raisons de stabilité [YLN03]), soit une vitesse moyenne de 1.5m/s. La portée moyenne des nœuds du réseau est de 300 à 400m. Ainsi, durant 4 minutes d'intervalle de vérification de la cohésion, un membre parcourrait en moyenne 360m, soit une seule portée.

8.6 Conclusion

La réalisation d'un service pour sécuriser l'application de Jukebox coopératif et distribué, dénommée JDukebox, s'inscrit dans le cadre de la validation du protocole de gestion de clé de groupe que nous avons proposé, BALADE. La mise en œuvre de BALADE a montré que notre protocole tenait parfaitement compte des caractéristiques et spécificités du modèle de communications temps réel, multi-sources séquentielles.

L'implantation de BALADE et JDukebox est disponible dans "http ://potiron.loria.fr/projects/madynes/jdukebox/BALADE". Son déploiement sur une plateforme réelle est en cours de réalisation.

Nous avons également réalisé une évaluation quantitative de BALADE, au travers de simulations dans NS2, que nous présentons dans le chapitre suivant.

Chapitre 9

Évaluation des performances de BALADE avec NS2

Sommaire

9.1 Introduction

L'évaluation des performances d'un système est une phase indispensable, qui permet de valider ses atouts et de les valoriser, ainsi que de le comparer avec d'autres approches existantes selon des métriques bien spécifiées. Dans ce chapitre, nous présentons dans un premier temps l'outil de simulation de réseaux NS2. Ensuite, nous entamons la phase d'évaluation des performances avec NS2, qui consiste à simuler l'algorithme OMCT, évaluer l'impact du modèle de mobilité sur les performances du couplage de BALADE avec OMCT, valider l'apport de l'utilisation des MPRs dans OMCT et finalement, comparer BALADE avec d'autres protocoles de gestion de clé de groupe dans les MANETs.

9.2 Présentation de l'outil de simulation de réseaux NS2

NS2[19] (*Network Simulator*) est un simulateur de réseaux à événements discrets. Il est paru en 1995, dans le cadre du projet VINT (*Virtual InterNetwork Testbed*) de la DARPA (*Defense Advanced Research Projects Agency*). Plusieurs niveaux d'abstraction sont offerts par NS2 : routage, propagation des paquets, modèle de trafic, applications,... NS2 est librement distribué (*open source*), permettant une collaboration aisée entre chercheurs pour le partage de codes et de modèles et facilitant ainsi la comparaison des protocoles aux fonctionnalités similaires. NS2 est un simulateur orienté objet, écrit en C++ avec une interpréteur Otcl comme interface. Le simulateur se base sur une hiérarchie de classes C++ (appelée hiérarchie compilée) et une hiérarchie de classes similaire en Otcl (appelée hiérarchie interprétée). Les deux hiérarchies sont étroitement liées l'une à l'autre, de façon à permettre deux niveaux de simulation :

- La simulation détaillée des protocoles requiert un langage de programmation qui manipule efficacement les octets, les entêtes, les paquets et implante des algorithmes qui s'exécutent sur de larges ensembles de données, en optimisant la vitesse d'exécution (Programmation C++).
- L'étude et l'évaluation de performances des protocoles impliquent de multiples simulations avec des variations continues de paramètres de configuration d'un grand nombre de scénarii (Programmation Otcl).

Trois phases sont nécessaires pour réaliser une simulation. La première consiste à définir la topologie de simulation. Cette phase consiste à déterminer les nœuds du réseau, les liens entre eux (médium de communication) et le protocole de routage à utiliser pour assurer ces communications. Ensuite, au cours de la deuxième phase sont définis les acteurs de la simulation et l'affectation des agents de communications aux nœuds du réseau. Les agents représentent les points terminaux où les paquets se construisent et se consomment. Les échanges de données sont toujours effectués entre les agents, implantant des protocoles de différents niveaux. La dernière phase de simulation consiste à définir la dynamique de la simulation : déterminer les paramètres tels que le trafic entre agents, les scénarii de mobilité des nœuds dans le cas de simulations d'un environnement sans fil, ...

Les résultats des simulations sont présentés de deux manières : un fichier de sortie *.nam qui peut être exploité par l'outil de visualisation NAM et un fichier de sortie *.tr qui sauvegarde les différents échanges de messages entre les nœuds du réseau et qui peut être exploité par des analyseurs de fichiers pour l'évaluation de performances.

[19] http ://www.isi.edu/nsnam/ns/ns-build.html

9.3 Métriques de simulation et de validation

Les métriques de simulations que nous adoptons pour évaluer les performances de nos contributions sont les suivantes :

- le délai de bout en bout (**D**) qui représente la latence moyenne de transmission de clés d'une source vers les récepteurs. Ce facteur permet d'évaluer le temps moyen nécessaire pour acheminer une clé de la source du groupe vers tous ses membres. Pour assurer une synchronisation efficace entre chiffrement et déchiffrement du flux multicast de la part de la source et des récepteurs, respectivement, cette latence doit être optimisée.
- la consommation de l'énergie (**E**) qui est définie comme étant la somme des unités requises pour la transmission des messages de transmissions de clés tout au long de la durée de la simulation. L'évaluation de ce facteur est primordiale dans notre étude, car l'optimisation de la consommation de l'énergie est un véritable challenge pour les nœuds d'un réseau ad hoc.
- le taux d'acheminement de paquets (**PDR**) est le rapport entre le nombre de paquets reçus et le nombre de paquets émis multiplié par le nombre de récepteurs. Cette métrique permet de mesurer le taux de fiabilité du protocole en terme d'acheminement des clés du groupe de la source aux destinations.

$$PDR = \frac{\texttt{Nombre\_de\_paquets\_reçus}}{\texttt{Nombre\_de\_paquets\_émis} * \texttt{Nombre\_de\_récepteurs}}$$

9.4 Validation de OMCT

Nous avons réalisé des simulations, afin de comparer notre approche de clusterisation OMCT, avec l'approche sans clusterisation et l'approche avec clusterisation indépendante de la localisation des membres du groupe, selon les métriques de latence et d'énergie.

L'approche sans clusterisation est représentée par un groupe multicast centralisé autour d'un seul contrôleur global, responsable de la distribution périodique de la clé de groupe à tous les membres adhérents au groupe. Les secrets futur et passé sont également assurés : un processus de renouvellement de la clé de groupe est déclenché par le contrôleur global après chaque événement de Join ou de Leave dans le groupe.

Dans l'approche de clusterisation indépendante de la localisation, la division du groupe en clusters se fait dynamiquement en exécutant une fonction d'évaluation dont les paramètres sont le nombre de membres locaux et la fréquence locale de dynamicité (nombre de Join et de Leave par unité de temps). Chaque membre du groupe calcule sa fonction d'évaluation. Si le nombre de ses membres locaux ou sa fréquence locale dépassent un certain seuil, le membre passe à l'état contrôleur local et forme un cluster avec ses membres locaux. Dans cette approche, les clés de chiffrement locales sont distribuées périodiquement par les contrôleurs du groupe à leurs membres respectifs. Ces clés sont également renouvelées à chaque procédure locale de Join ou de Leave. Cette méthode de clusterisation est présentée dans [BCF04].

Dans l'approche de clusterisation avec OMCT, la clé commune de chiffrement de données est renouvelée périodiquement par les contrôleurs du groupe, qui de plus, renouvellent leurs clés locales (KEKs) à chaque événement de Join ou Leave dans leur cluster.

Pour réaliser nos simulations, nous avons utilisé la version ns-allinone-2.26 de NS2. L'envi-

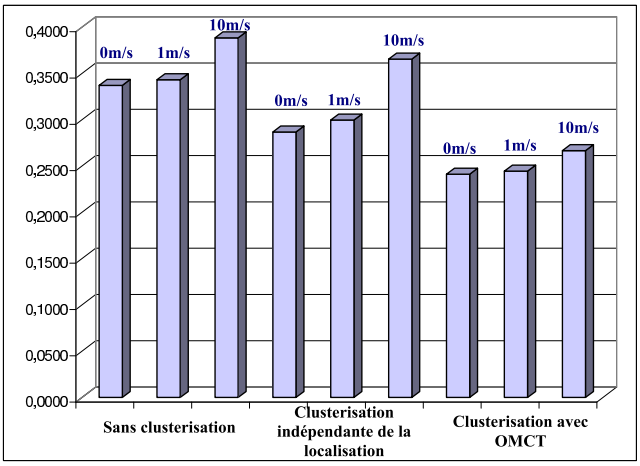


FIG. 9.1 – Latence d'acheminement de clés

ronnement de simulation est défini comme suit. La surface du réseau est 500*500m. Le nombre de nœuds est défini à 50 et 100. La durée de la simulation est 1000s. Dans la couche physique, le standard 802.11 est utilisé, fournissant un débit de 2Mbps et une portée de transmission à 250m. Le modèle de mobilité adopté est le *Random Waypoint*, avec des vitesses de déplacements de 0m/s, 1m/s et 10m/s. Le protocole de routage multicast est MAODV [RP00]. KUNZ ET AL. [ZK04] fournissent une implantation de ce protocole sous NS2.26. L'arrivée dans le groupe suit une loi de poisson, de paramètre λ (arrivées par unité de temps) et la durée d'appartenance au groupe multicast suit une loi exponentielle de moyenne μ. Il est à noter que seul le trafic de distribution de clés existe dans ces simulations.

La figure 9.1 montre la latence moyenne de transmission de clés, avec l'approche sans clusterisation, l'approche avec clusterisation indépendante de la localisation et l'approche avec OMCT.

La clusterisation avec OMCT apporte un profit en latence, évalué de 28% à 31%, comparé à une approche sans clusterisation et de 15% à 27% par rapport à une approche avec clusterisation indépendante de la localisation. En effet, une approche centralisée sans clusterisation souffre du phénomène "1 affecte n" ; l'unique contrôleur est responsable de la distribution de la clé de groupe à tous les membres, après tout événement de Join ou de Leave. Les membres du groupe sont aussi répartis aléatoirement par rapport à la localisation du contrôleur local, ce qui augmente le temps moyen nécessaire pour la transmission de clés dans le réseau ad hoc. La latence évaluée pour les approches avec clusterisation indépendante de la localisation et OMCT, est plus faible que celle calculée dans le cas d'une approche sans clusterisation, de par leur atténuation du phénomène "1 affecte n". Cependant, la meilleure latence est évaluée avec notre approche à clusterisation dynamique. En effet, l'algorithme OMCT assure la formation de clusters fortement corrélés ; chaque cluster étant géré par un contrôleur local géographiquement proche de ses membres locaux, ce qui optimise efficacement le délai moyen de transmission de clés.

La figure 9.2 montre que OMCT apporte une économie d'énergie dans la distribution de clés au sein du groupe multicast. Cette économie est de 7% à 10% comparé à une approche sans clusterisation et de 8% à 23% par rapport à une approche à clusterisation indépendante

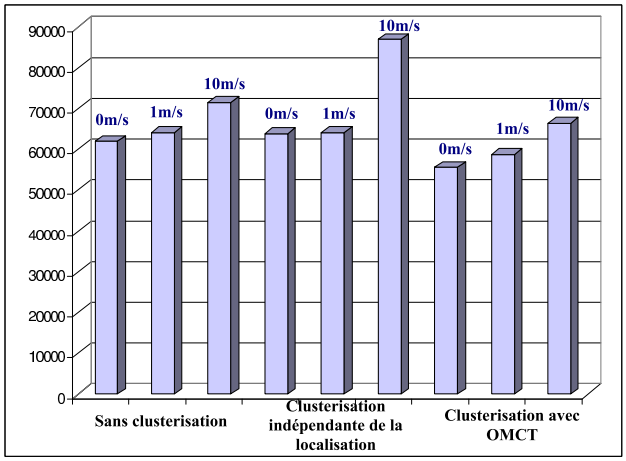


FIG. 9.2 – Consommation en énergie

de la localisation.
L'optimisation de l'énergie évaluée avec OMCT s'explique de la même manière que pour le gain en latence de distribution de clés. En effet, en formant des clusters fortement corrélés géographiquement, ayant des facteurs de cohésion proches de 1, les contrôleurs locaux consomment moins d'énergie pour atteindre tous les membres de leurs clusters.

Le facteur de mobilité des nœuds affecte les performances de notre approche de clusterisation avec OMCT. Ceci est dû au temps nécessaire pour la construction des clusters. Cependant, même pour un environnement fortement mobile, l'approche utilisant OMCT reste la meilleure en termes d'efficacité en consommation d'énergie et de latence moyenne de distribution de clés.

9.5 Impact du modèle de mobilité sur les performances de BALADE couplé avec OMCT

La plupart des travaux de recherche dans les réseaux ad hoc ont utilisé des modèles de mobilité individuelle, principalement en raison de la disponibilité du code de ces modèles avec les simulateurs réseaux tel que NS et GloMoSim et parce que les modèles de groupe ajoutent de nouveaux facteurs et paramètres à prendre en compte dans les simulations [CMS02].

Cependant, dans un réseau ad hoc, il peut être nécessaire de modéliser le comportement de nœuds se déplaçant par groupe dans le réseau. Plusieurs exemples d'applications illustrent ce comportement, par exemple : groupe de soldats dans un scénario militaire, forces civiles dans une mission, pompiers se déplaçant en petits groupes, ...

Dans cette section, nous analysons le comportement du couplage BALADE / OMCT avec différents modèles de mobilité, individuelle et de groupe et étudions l'impact de ces modèles sur les performances de notre approche [BCF06b, BCF05b].

9.5.1 Modèles de mobilité

Les différents modèles de mobilité que nous avons utilisés au cours des simulations sont les suivants :

1. Modèle *Random Waypoint Mobility* (RW)

   Le modèle de mobilité individuelle le plus répandu est le *Random Waypoint Mobility Model*. Le mouvement de nœuds selon ce modèle est caractérisé par deux facteurs : la vitesse maximale et le temps de pause. Chaque nœud commence son mouvement depuis sa position initiale, jusqu'à une position aléatoire dans la surface du réseau. La vitesse du nœud est uniformément distribuée entre 0 et la vitesse maximale. Quand un nœud atteint sa position de destination, il attend le temps de pause, sélectionne une autre destination et se déplace de nouveau. La figure 9.3 illustre la mobilité selon le modèle RW (capture d'écran de NAM : outil de visualisation de NS).

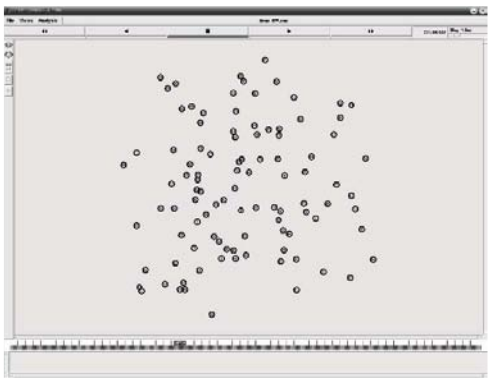


FIG. 9.3 – Random Waypoint Mobility

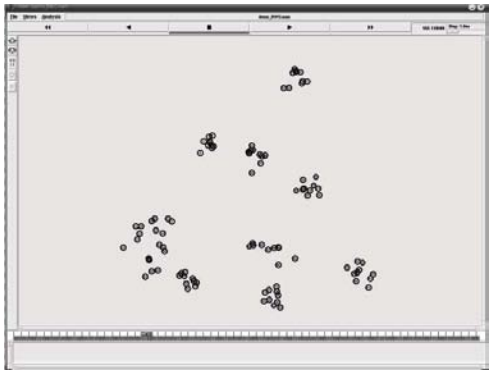


FIG. 9.4 – Random Waypoint Group Mobility

2. Modèle *Random Waypoint Group Mobility* (RWG)

   Ce modèle est une extension du Modèle Random Waypoint Mobility, appliquant la mobilité à un sous-ensemble de nœuds proches géographiquement. Cette extension permet de mettre en évidence les caractéristiques des communications inter et intra sous-groupes. La figure 9.4 illustre la mobilité selon le modèle RWG (capture d'écran de NAM : outil de visualisation de NS).

3. *Modèle Manhattan Group Mobility* (MHG)

   Le modèle MHG permet seulement des mouvements à directions horizontales ou verticales. Ce modèle est adapté à des environnements contraints, où les routes doivent suivre des directions bien spécifiées.

4. *Modèle Sequential Group Mobility* (SQG)

   Le modèle SQG applique le modèle RWG à tous les sous-groupes, en séquence. Les sous-groupes sont ordonnés et chaque sous-groupe i doit se déplacer vers le sous-groupe $i-1$. Ce modèle est adapté à des scénarii militaires tactiques où des groupes se suivent pour effectuer une mission donnée.

9.5.2 Simulations et résultats

Nous réalisons les simulations sous Linux Mandrake 10.0, utilisant le simulateur des réseaux NS2 version ns-allinone-2.26. Le protocole de routage multicast que nous utilisons est MAODV [RP00]. Tous les nœuds sont membres du groupe multicast dès le début de la simulation.

Pour générer des scénarii de mobilité avec les différents modèles présentés ci-dessus, nous utilisons le générateur **setdest** pour le modèle RW de mobilité individuelle et le générateur **grcmob** [CMS02] pour les modèles de mobilité de groupe.

Le générateur **setdest** requiert les paramètres suivants : le nombre total de nœuds=100, le temps de simulation=2000 sec, la surface du réseau=500*500 m, la vitesse maximale=3 m/s et le temps de pause=20 sec.

Le générateur **grcmob** requiert les paramètres suivants :

- le nombre de sous-groupes = 10,
- le nombre total de nœuds = 100, nous assumons que le nombre de nœuds est partagé équitablement pour tous les sous-groupes, soit 10 membres par sous-groupe,
- le temps de simulation = 2000 sec, durant les 1000 premières secondes, les nœuds se déplacent seulement dans le réseau sans génération de trafic. Ceci permettra au système d'être stable avant la génération du trafic [YLN03],
- la surface du réseau =500*500 m,
- la vitesse maximale =3 m/s.

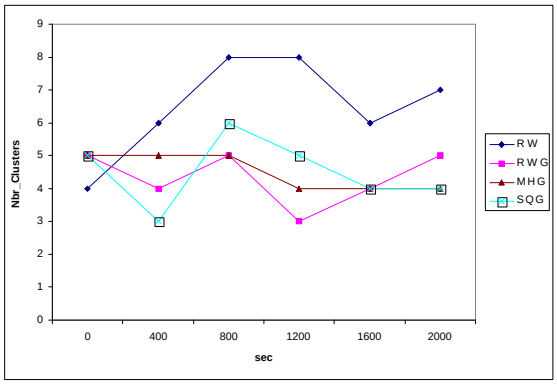


FIG. 9.5 – Nombre de clusters générés par OMCT au cours du temps

Pour un sous-groupe donné, un nœud est désigné comme la référence pour les autres membres de son sous-groupe. Le critère de choix de ce nœud référence n'a pas d'importance ; selon **grcmob**, c'est le nœud détenant le plus petit identificateur. Trois facteurs de mobilité sont définis dans **grcmob** :

- la sensibilité de la distance du groupe qui indique la distance maximale entre le nœud référence et tout autre nœud de son sous-groupe (valeur choisie = 40 m) ;
- la sensibilité de la vitesse du groupe qui indique un intervalle de valeurs pour la vitesse de chaque nœud par rapport au nœud référence (valeur choisie = +-0.15 m/sec) ;
- la sensibilité d'initialisation du mouvement du groupe qui indique quand un nœud commence son mouvement par rapport au nœud référence (valeur choisie = +-0.15 sec).

Tous les modèles de mobilité de groupe que nous utilisons se basent, comme pour le modèle *Random Waypoint Mobility*, sur des périodes de mobilité et des périodes de pause. La période de pause maximale est définie à 20 sec, valeur définie dans [YLN03] pour améliorer la stabilité des simulations.

Le trafic de distribution de clés est simulé de la façon suivante : initialement, la source envoie la clé TEK à tous les membres du groupe multicast, et dynamiquement, des sous-groupes vont se créer selon OMCT. Chaque groupe sera géré par son contrôleur local. À ce

stade, la source envoie la TEK aux différents contrôleurs locaux, qui l'acheminent à leurs membres locaux. Le renouvellement de la clé de chiffrement de données est effectué chaque 250 sec. Seul le trafic de distribution de clés existe dans notre simulation.

L'objectif des ces simulations est de valider l'apport du couplage BALADE / OMCT, dans des environnements ad hoc à mobilité de groupe par rapport à une mobilité individuelle, selon les trois métriques de délai d'acheminement des clés, de consommation en énergie et de taux de transmission de clés.

Les résultats des simulations sont représentés dans les figures 9.5, 9.6, 9.7 et 9.8. La figure 9.5 représente le nombre de clusters générés pas OMCT au cours du temps. La figure 9.6 montre le délai moyen de transmissions de clés de la source vers tous les récepteurs du groupe, en passant par les contrôleurs locaux des clusters. La figure 9.7 présente la consommation en énergie pour la transmission de clés du groupe et la figure 9.8 montre le taux d'acheminement de clés avec les quatre modèles de mobilité.

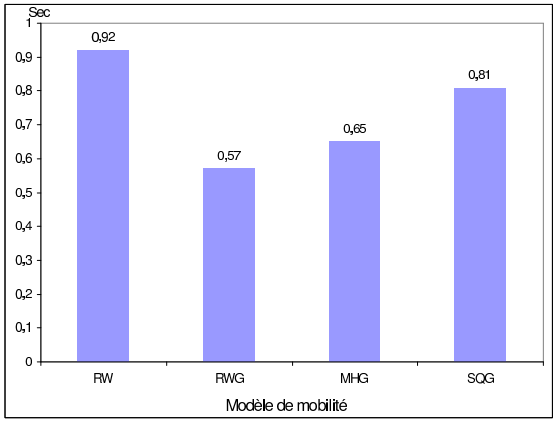


FIG. 9.6 – Délai moyen de transmission de clés

Le modèle *Random Waypoint* détient le délai de transmission de clés le plus grand, 0.92 sec. Ceci est dû à la mobilité individuelle de chaque membre. La transmission de clés peut potentiellement utiliser plusieurs nœuds relais pour arriver de la source aux destinations. La mobilité individuelle rend aussi plus fréquente l'exécution de l'algorithme OMCT pour diviser le groupe en clusters fortement corrélés, car le paramètre de cohésion change d'une façon imprédictible (cf. Figure 9.5). Un autre facteur intervient pour le délai moyen d'acheminement des clés et le taux de fiabilité : c'est l'effet de densité présenté dans [CMS02] pour le modèle *Random Waypoint*. Cet effet tend à centrer tous les nœuds du réseau autour du centre de la surface de simulation, ce qui augmente le degré de congestion du réseau et multiplie les retransmissions.

Nous remarquons aussi que pour le modèle *Random Waypoint*, la consommation de l'énergie requise pour l'acheminement des clés augmente quand le délai de transmission des clés augmente. En effet, plus le nombre moyen de relais d'acheminement entre source et récepteur est grand, plus l'énergie est consommée et plus la latence moyenne de délivrance de clés est grande.

Le modèle *Random Waypoint Group* met en moyenne un délai de 0.57 sec pour la transmission des clés et présente un gain en énergie de 20% et un gain en fiabilité de transmission de clés de 10% par rapport au modèle *Random Waypoint*. Ceci est dû à la nature des scénarii

de mobilité de groupe, où les communications intra-groupe ne nécessitent pas de nœuds relais pour l'acheminement du trafic de clés, minimisant ainsi le délai de transmission, augmentant la fiabilité et optimisant la consommation de l'énergie.

La clusterisation réalisée par OMCT dans le cadre du modèle *Random Waypoint Group* tire profit de cet avantage, en formant des clusters dépendant de la localisation et de la mobilité en collectivité des membres du groupe multicast. Ainsi, les membres d'un même groupe de mobilité appartiennent au même cluster et appartiennent presque toujours à la portée de leur contrôleur local CL. La plupart des clés acheminées sont ainsi entre CLs et membres locaux des clusters. Les pertes de clés sont plus fréquentes pour les communications inter-clusters.

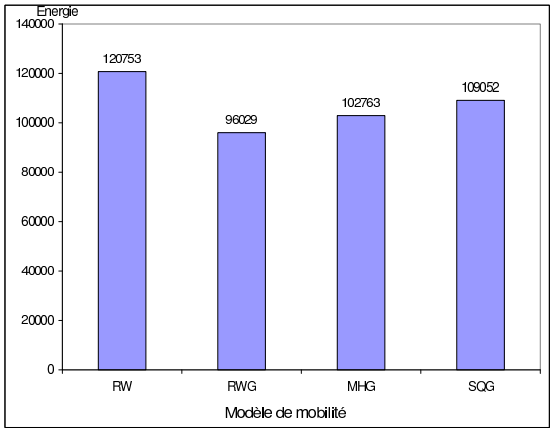


FIG. 9.7 – Consommation de l'énergie

En général, nous ne pouvons pas observer une différence nette entre les deux modèles *Random Waypoint Group* et *Manhattan Group* [CMS02]. Ce fait peut être expliqué par le nombre assez petit des groupes de mobilité (10 groupes de 10 membres) qui tend à rendre ces deux scénarii de mobilité similaires [CMS02]. Cependant, nous remarquons que le modèle *Manhattan Group* détient un délai moyen de délivrance de clés plus grand que celui du modèle *Random Waypoint Group*, consomme plus d'énergie pour la transmission des clés et détient le meilleur taux de fiabilité. Ceci peut être expliqué par le fait que le nombre de clusters formé par OMCT avec le modèle RWG est un peu plus petit que celui réalisé avec MHG, au cours du temps (cf. Figure 9.5).

Le modèle *Sequential Group Mobility* détient le plus grand délai moyen de transmission de clés par rapport aux autres modèles de mobilité de groupe. Il consomme plus d'énergie et est moins fiable que tous les autres modèles. Ceci s'explique par la vitesse réduite que nous utilisons pour notre simulation, pour des raisons de stabilité des résultats [YLN03]. En effet, [CMS02] montre que le modèle *Sequential Group Mobility* se comporte mieux quand la vitesse des nœuds augmente, en terme de fiabilité et de délai moyen de bout en bout de délivrance de clés. Dans ce modèle, les groupes se suivent les uns les autres, ainsi, quand la vitesse augmente, la distance entre les groupes diminue pour aboutir à une forte fiabilité de transmission de données et un délai minimum de bout en bout. Le comportement de OMCT avec le modèle *Sequential Group Mobility* dépend aussi de ce fait ; la faible vitesse des groupes augmente la distance entre les clusters BALADE et de ce fait, la majorité des communications inter-clusters sont perdues. Les communications intra-cluster, entre CL et membres locaux, restent fiables. Le nombre de clusters formé par OMCT dans le cadre de ce modèle de mobilité présente aussi

une variation nette au cours du temps, par rapport aux autres modèles de mobilité de groupe (cf. Figure 9.5).

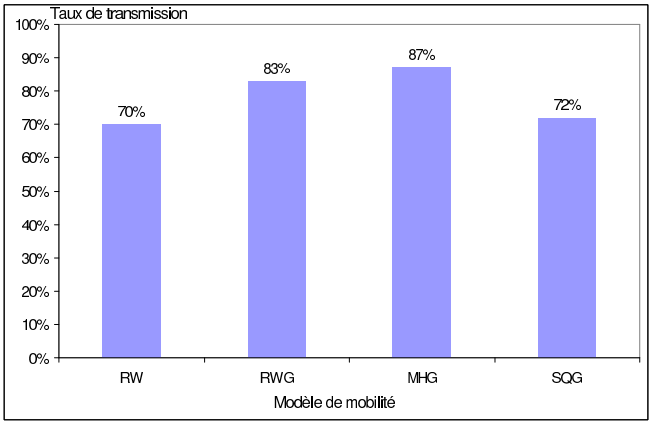


FIG. 9.8 – Taux d'acheminement de clés

Les résultats des simulations que nous avons réalisées montrent une forte dépendance entre le modèle de mobilité utilisé et le comportement de notre algorithme de distribution de clés à clusterisation dynamique OMCT. Nous avons montré également que les performances du couplage BALADE / OMCT sont meilleures avec un modèle de mobilité de groupe.

9.6 Validation de l'intégration des MPRs dans OMCT

L'objectif de ces simulations est de comparer l'algorithme de clusterisation OMCT, avec l'approche intégrant la technique de relais multipoints dans OMCT, selon les métriques de délai d'acheminement des clés, de consommation en énergie et de taux d'acheminement des clés.

Les simulations ont été réalisées sous Linux Mandrake 10.0, avec l'outil de simulation de réseaux NS2 version ns-allinone-2.26. L'environnement de simulation est composé des paramètres suivants :

- surface : 300*300 m.
- nombre de nœuds : 20.
- durée de la simulation : 2000 sec. Durant les premières 1000 secondes, les nœuds se déplacent dans le réseau sans générer du trafic. Ceci permet une stabilité du système de simulation avant la génération du trafic [YLN03].
- couche physique : IEEE 802.11.
- modèle de mobilité : le modèle *random waypoint* avec un temps de pause égal à 20 secondes et une vitesse maximale des nœuds égale à 3 m/s.
- protocole de routage : OLSR
- trafic : seulement le trafic de distribution de clés existe dans la simulation. La source du groupe envoie la TEK aux contrôleurs locaux, qui l'acheminent aux membres de leur cluster respectif.

Les résultats des simulations sont présentés dans la figure 9.9.

Le couplage de OMCT avec la technique des MPRs apporte un profit intéressant en terme

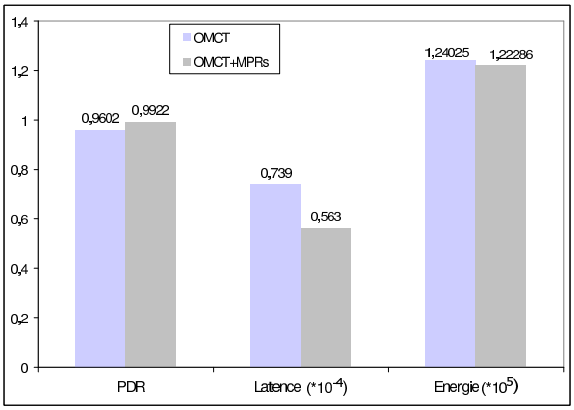


FIG. 9.9 – OMCT Vs OMCT avec MPRs

de taux de transmission de clés, comparé à l'algorithme OMCT. En effet, la clusterisation du groupe est réalisée suivant la connectivité réelle entre les nœuds. La distribution de clés suit l'arbre multicast ainsi construit et fournit un meilleur taux de fiabilité.

La latence moyenne de transmission de clés et la consommation d'énergie sont meilleures avec l'approche combinant OMCT avec la technique des relais multipoints. Ceci s'explique par le fait que le nombre de clusters créés est plus grand avec OMCT. Par conséquence, plus de contrôleurs locaux acheminent les clés de la source aux membres du groupe, ce qui implique une plus grande consommation en énergie et plus d'interférences de signaux.

Il est à noter que la transmission de clés est réalisée en unicast, car la version multicast du protocole OLSR (MOLSR : *Multicast Optimized Link State Routing*) n'est pas encore disponible dans le simulateur NS2. L'utilisation du protocole MOLSR devrait optimiser la latence moyenne, le taux d'acheminement de clés et la consommation de l'énergie de l'approche combinant OMCT avec les MPRs. En effet, la recherche de l'ensemble des MPRs d'un nœud dans MOLSR est calculée de façon à privilégier les nœuds à capacité d'envoyer des messages en multicast, quand c'est possible [LJM+03]. L'élection des contrôleurs locaux serait ainsi effectuée parmi les nœuds MPRs choisis par MOLSR, ayant la capacité d'envoyer des données en multicast.

9.7 Comparaison de BALADE avec d'autres protocoles de gestion de clés dans les MANETs

Dans cette section, nous présentons une comparaison[20] des performances de notre principale contribution (le protocole de gestion de clé de groupe dédié aux MANETs BALADE couplé avec OMCT), avec deux autres protocoles de gestion de clé de groupe dans les MANETs (un de chaque famille selon la taxonomie présentée dans la figure 2.6) : un protocole centralisé GKMPAN [ZSXJ04b] et un protocole distribué DMGSA [KLG06].

[20]Ce travail a été réalisé en collaboration avec Mohamed Bouali, stagiaire en DEA au sein de l'équipe de recherche MADYNES

9.7.1 Environnement de simulation

Les simulations sont réalisées sous Linux Mandrake 10.0, avec le simulateur NS2.26. Leur objectif est de comparer BALADE, DMGSA et GKMPAN selon les trois métriques présentées dans la section 9.5.2 (délai moyen d'acheminement des clés aux membres du groupe, taux d'acheminement des clés et énergie consommée évaluée comme étant l'énergie nécessaire pour la transmission et l'acheminement des messages tout au long des simulations).

Nous définissons notre environnement de simulation comme suit :

- les paramètres de simulation sont le nombre de membres du groupe (30 - 40 - 50) et la surface du réseau ad hoc (500m*500m, 1000m*1000m, 1500m*1500m). Ces deux paramètres définissent la densité des membres du groupe dans le réseau ;
- les scénarii de mobilité sont générées avec le générateur **setdest** ; la vitesse maximale des membres est définie à 10km/h (2.77m/sec), le temps de pause est 20 secondes et la durée des simulations est 2000 sec ;
- chaque configuration des paramètre des simulation est répétée 10 fois ;
- le protocole de routage multicast est MAODV [RP00] ;
- seul le trafic de distribution de clés existe au cours des simulations ; ce trafic commence à l'instant 1000 sec et consiste à distribuer la clé du groupe et à la renouveler toutes les 200 secondes ;
- tous les membres du réseau sont adhérents au groupe multicast dès le début de la simulation.

9.7.2 Présentation des agents des protocoles

Nous avons implanté trois agents correspondants aux protocoles simulés BALADE, DMGSA et GKMPAN (cf. Annexe C) et nous les avons intégrés dans le simulateur NS. L'agent BALADE comporte la simulation du contrôleur global, des contrôleurs locaux et des membres du groupe. Le contrôleur global est responsable de la distribution de la clé de groupe aux contrôleurs locaux, qui l'acheminent à leurs membres de cluster, toutes les 200 secondes. Il est à noter que, pour des raisons d'implantation et d'applicabilité, chaque cluster détient une adresse IP multicast choisie par son contrôleur local. Les membres d'un cluster joignent le groupe multicast formé par le contrôleur local dès leur adhésion à son cluster. La clusterisation dans BALADE, réalisée par OMCT, est implantée de façon à assurer la formation de clusters fortement corrélés, dont tous les membres sont à la portée (à un seul saut) de leurs contrôleurs locaux.

Pour l'agent DMGSA, des chefs de clusters sont élus et assurent la génération de la clé de groupe et sa distribution à leurs membres locaux chaque 200 secondes. Les clusters sont formés à deux sauts de leurs chefs locaux et sont maintenus grâce à des messages envoyés périodiquement (toutes les 10 secondes) par les chefs à leurs membres locaux avec un TTL égal à 2. Un membre n'ayant pas reçu un message de maintenance de cluster auquel il appartient pendant 50 secondes entame une procédure de formation d'un cluster et invite ses voisins à deux sauts à le joindre. Comme dans BALADE, chaque cluster détient une adresse IP multicast choisie et distribuée par le chef.

L'agent GKMPAN simule un chef de groupe qui distribue la clé de chiffrement à tous les membres de son groupe, de façon centralisée. La distribution de clés dans GKMPAN étant réalisée saut par saut, un délai est ajouté au niveau de l'acheminement des paquets de distribution de la clé du groupe par les nœuds intermédiaires. Ce délai correspond au temps

moyen de transmission des clés d'un membre à ses voisins et aux opérations de chiffrement et de déchiffrement des clés lors de ces acheminements.

9.7.3 Résultats et analyses

Les résultats de nos simulations sont présentés dans les figures 9.10, 9.11 et 9.12 (cf. Annexe C). La figure 9.10 comporte une comparaison des délais d'acheminement des clés, pour les trois protocoles de sécurité de groupe (BALADE, DMGSA et GKMPAN) par rapport aux différentes densités des membres dans le réseau. La figure 9.11 montre la consommation de l'énergie des trois protocoles, tout au long de la durée de la simulation. Finalement, la figure 9.12 présente les taux de transmission des clés aux membres du groupe.

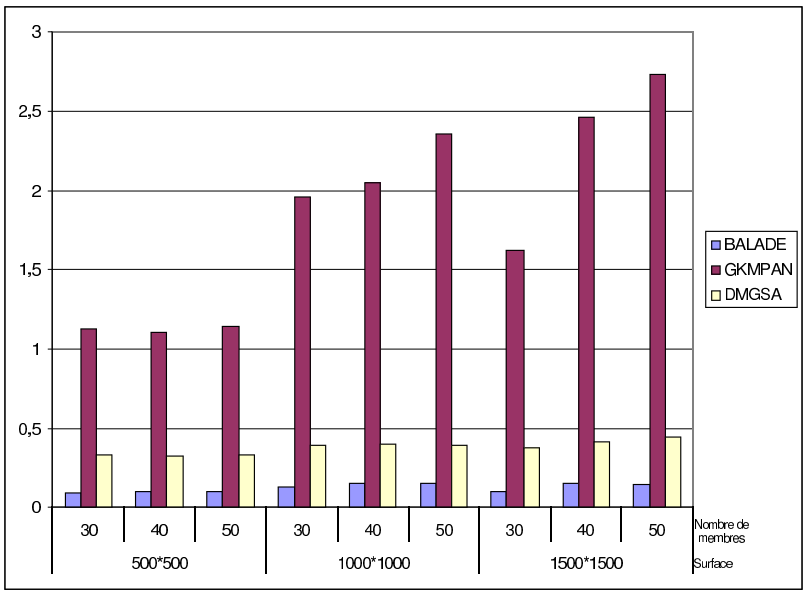


FIG. 9.10 – Délai moyen de transmission des clés (en secondes)

Le protocole de gestion de clé de groupe BALADE détient le meilleur délai moyen d'acheminement des clés (gain évalué jusqu'à 30% sur DMGSA et 90% sur GKMPAN) (cf. Figure 9.10). Ce gain est dû à la clusterisation de BALADE réalisée via l'algorithme OMCT, qui assure une formation de clusters fortement corrélés, dans lesquels tous les membres sont atteignables à un seul saut par leurs contrôleurs locaux. Le délai d'acheminement des clés est ainsi fortement dépendant de la cohésion des clusters autour des contrôleurs locaux.
L'efficacité et la rapidité de la distribution de clés dans BALADE sont suivies par un taux de fiabilité satisfaisant (cf. Figure 9.12). BALADE apporte un gain en terme du taux d'acheminement de paquets variant de 20% à 40% par rapport à DMGSA pour des densités fortes de membres dans le groupe (dans une surface de 500*500m). En effet, la distribution de clés suit les chemins à un seul saut entre les contrôleurs locaux et leurs membres de sous-groupes, minimisant ainsi le taux de pertes et d'interférences. Seule la distribution de la clé du groupe du contrôleur global aux contrôleurs locaux peut suivre des chemins à multi-sauts.
Le gain de BALADE par rapport à DMGSA en terme de fiabilité de la transmission des

clés implique un profit considérable en consommation de l'énergie globale requise pour cette transmission (cf. Figure 9.11). En effet, BALADE minimise les relais responsables de l'acheminement des clés aux autres membres du groupe. Le choix judicieux des contrôleurs locaux par rapport à leur localisation géographique induit ainsi une optimisation de la consommation de l'énergie. Cependant, les simulations réalisées montrent que les performances de BALADE sont meilleures lorsqu'il est déployé avec de fortes densités de membres de groupe dans le réseau ad hoc. Une forte densité de membres limite les communications inter-clusters et par conséquent améliore la fiabilité, le délai et le taux d'acheminement des clés.

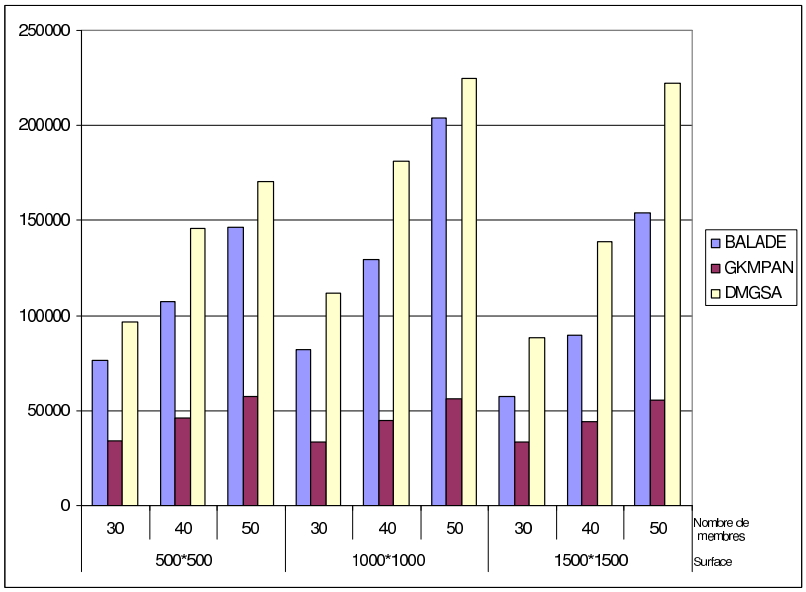


FIG. 9.11 – Consommation de l'énergie

Le protocole de gestion de clé DMGSA assure une clusterisation du groupe multicast de façon distribuée, réalisée grâce à la coopération et la collaboration de tous les membres du groupe. L'élection d'un chef de cluster est ainsi effectuée dans son voisinage à deux sauts et maintenue grâce à des messages qu'il envoie périodiquement pour informer ses membres locaux de sa disponibilité, ainsi que pour inviter d'autres membres à joindre son cluster. Le délai moyen de transmissions de la clé du groupe est directement affecté par la formation des clusters à deux sauts. En effet, en plus des chefs locaux qui acheminent les clés à un saut, des nœuds intermédiaires (les membres locaux dans leur portée) sont également chargés de router ces clés aux membres à deux sauts, adhérents au cluster. Le délai de transmission de clés dans DMGSA (cf. Figure 9.10) est donc nettement supérieur à celui dans BALADE. Cependant, un gain considérable est enregistré par rapport à GKMPAN.
La consommation de l'énergie dans DMGSA est la plus élevée par rapport à BALADE et GKMPAN (cf. Figure 9.11), pour toutes les configurations de densités de membres dans le réseau. Ceci s'explique également par le nombre important de relais qui acheminent les clés aux membres du groupe.
En terme de fiabilité de la transmission des clés, les performances de DMGSA dans le cadre de groupe multicast à forte densité sont nettement inférieures à celles de GKMPAN et BALADE (cf. Figure 9.12). Dans le cadre de groupes multicast à faible densité dans le réseau, les perfor-

mances des trois protocoles de gestion de clés sont pratiquement similaires. Comme exemple, dans la configuration de 50 membres déployés dans une surface de 1500*1500, DMGSA apporte la meilleure fiabilité de la transmission de clés. Ceci est dû à sa clusterisation à deux sauts, qui permet une couverture plus grande des nœuds géographiquement loin du centre de la surface du réseau.

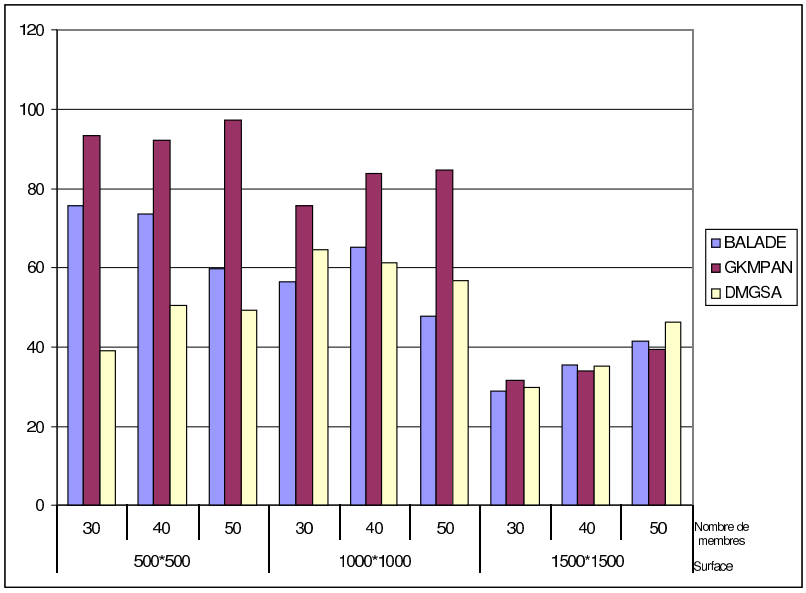


FIG. 9.12 – Taux d'acheminement des clés (%)

GKMPAN est un protocole centralisé de gestion de clé de groupe dans les MANETs. Une seule entité est responsable de la distribution et du renouvellement de la clé aux membres du groupe multicast. La distribution des clés dans GKMPAN est sécurisée par des opérations de chiffrement et de déchiffrement intermédiaires de la clé du groupe (par tous les nœuds qui acheminent les clés à leurs voisins). Le délai moyen d'acheminement de clés est ainsi évalué comme étant la somme de tous les délais causés par les membres intermédiaires lors de la préparation des messages d'acheminement de clés. Ce délai moyen est ainsi très grand (cf. Figure 9.10) et met en cause une accessibilité rapide et efficace au flux de données émis par la source, ainsi qu'une synchronisation entre la source et les membres du groupe pour le chiffrement et le déchiffrement de données.
En terme de consommation de l'énergie et de fiabilité des transmissions des clés (cf. Figure 9.11, 9.12), GKMPAN détient les meilleures performances quand il est déployé au sein de groupes à forte densité dans le réseau, au coût d'un délai très grand d'acheminement de clés. Cependant, il est à noter que les performances de GKMPAN se dégradent considérablement dans le cadre des faibles densités des groupes multicast, en terme de fiabilité d'acheminement des clés. De plus, des opérations d'ajout ou de retrait d'entités dans le groupe seraient très contraignantes dans le cadre de GKMPAN, de par son architecture centralisée, comme cela a été démontré dans la section 9.4 concernant l'approche sans clusterisation (Figures 9.1 et 9.2). Une telle architecture implique des opérations de renouvellement de clés à tous les membres du groupe après chaque événement de Join ou de Leave. Ces processus de renouvellements de

clés ont un impact direct et négatif sur la consommation de l'énergie et sur le taux de fiabilité de la distribution des clés au sein du groupe multicast.

9.8 Conclusion

Dans ce chapitre, nous avons utilisé le simulateur de réseaux NS2 afin de valider et d'évaluer les performances de nos principales contributions et de les comparer avec les travaux existants, selon des métriques que nous avons spécifiées.
Nous avons simulé dans un premier temps notre algorithme de clusterisation dynamique des groupes multicast OMCT et nous l'avons comparé à l'approche sans clusterisation et l'approche avec clusterisation indépendante de la localisation. Nous avons montré que OMCT apporte un gain en termes de délai de transmission de clés et de consommation de l'énergie requise pour la distribution des clés aux membres du groupe.
Nous avons ensuite évalué l'impact du modèle de mobilité (individuelle ou de groupe) sur le comportement et les performances de BALADE couplé avec OMCT, concernant le délai moyen de transmission de clés, la consommation en énergie et le taux d'acheminement des clés. Les résultats des simulations ont montré que les performances de BALADE couplé avec OMCT sont nettement meilleures lorsqu'il est déployé au sein d'un réseau ad hoc suivant un modèle de mobilité de groupe.
Par la suite, nous avons validé l'apport de l'intégration des MPRs dans OMCT. L'utilisation de la technique de relais multipoints dans OMCT, grâce au mécanisme d'intégration de couches protocolaires, optimise la consommation de l'énergie, améliore le délai d'acheminement de paquets et augmente le taux de fiabilité de la distribution de clés. Cependant, cette intégration est contrainte par l'utilisation du protocole de routage OLSR.
La dernière partie de nos simulations avait pour objectif d'évaluer les performances de notre protocole BALADE couplé avec OMCT (sans MPRs) et de les comparer avec deux protocoles de gestion de clé de groupe dans les réseaux ad hoc GKMPAN et DMGSA. Nous avons montré que BALADE couplé avec OMCT apporte le meilleur compromis en termes de délai de transmission de clés, de consommation en énergie et de taux de fiabilité.

Synthèse

Conclusions et perspectives

1 Conclusions

Établir des communications de groupe sécurisées au sein d'un réseau ad hoc est un véritable challenge. En effet, un réseau ad hoc est un environnement hostile, qui apporte plusieurs défis de sécurité, dus à ses caractéristiques et ses spécificités (liens sans fil, capacités limitées, ...). À ces contraintes de sécurité, s'ajoutent les vulnérabilités du modèle IP multicast qui, de par sa nature, élimine toute possibilité d'identification des membres du groupe ou de confidentialité de données. Le déploiement des communications de groupe dans un réseau ad hoc induit également de nouvelles épreuves à prendre en compte lors de la conception d'une architecture de gestion de clés dans les réseaux ad hoc.

La solution la plus appropriée pour assurer des communications de groupe dans les MANETs est l'établissement d'un protocole de gestion de clé de groupe. Ce protocole doit garantir la confidentialité des données multicast en assurant la gestion et la distribution de la clé de chiffrement de données TEK. Le contrôle d'accès au groupe multicast est également assuré car seuls les membres détenant la clé du groupe peuvent accéder au flux multicast émis par la source. Le protocole de gestion de clé de groupe doit également être adapté à la nature et aux caractéristiques des réseaux ad hoc. Nous avons dans un premier temps étudié les différentes approches existantes de gestion de clé de groupe dans les MANETs, en les classant selon leur architecture (centralisée, distribuée ou décentralisée) et également leur prise en compte des contraintes des réseaux ad hoc (communications multi-sauts, support de la mobilité et optimisation de la consommation en énergie). Nous avons ensuite comparé et discuté les protocoles présentés, selon des métriques bien définies (services de sécurité assurés, contraintes et pré-requis, opérations intermédiaires de chiffrement et de déchiffrement, coût de stockage, passage à l'échelle et vulnérabilité). Nous avons montré que l'architecture appropriée pour un protocole de gestion de clé dans les MANETs est l'architecture décentralisée, puisqu'elle réduit le phénomène "1 affecte n" et ne se centralise pas autour d'une seule entité vulnérable et cible d'attaques malicieuses. Cette architecture doit en plus permettre le passage à l'échelle, tout en s'adaptant aux caractéristiques des réseaux ad hoc.
Ayant défini ces motivations pour la conception d'une approche de gestion de clé de groupe dans les MANETs, nous avons proposé deux protocoles à cet effet, un protocole orienté récepteurs membres du groupe multicast dénommé Enhanced BAAL et un protocole orienté sources du groupe, baptisé BALADE. Nous résumons dans le tableau 1 les évaluations de ces contributions, selon les métriques de comparaison définies dans le chapitre 2.

Enhanced BAAL [BCF04] est une adaptation et amélioration du protocole de gestion de clés dans les réseaux filaires BAAL [CCS00] au contexte des réseaux ad hoc. Enhanced BAAL intègre le support hybride et dynamique du protocole AKMP [BBC02] afin d'assurer le passage

à l'échelle et de prendre en compte la dynamicité des membres des groupes multicast. Il utilise également la technique de cryptographie à seuil [ZH99] garantissant l'authentification des membres et de la source du groupe, ainsi que la génération sécurisée des clés de chiffrement. L'architecture décentralisée à TEKs locales adoptée par Enhanced BAAL, le rend orienté récepteurs membres du groupe. Les secrets futur et passé sont garantis au sein de ce protocole. Il est ainsi adapté à la sécurisation des communications de données hautement sensibles. De plus, Enhanced BAAL diminue le phénomène "1 affecte n", rencontré dans BAAL. Le surcoût des opérations de chiffrement et de déchiffrement du flux émis par la source est atténué grâce à la clusterisation dynamique du groupe, qui tient compte de la fréquence d'adhésion et du nombre de membres locaux par cluster.

		Contraintes et pré-aquis	Services de sécurité	Surcoût de calcul	Surcoût de stockage	Surcoût de communications	Vulnérabilités
TEKs locales	Enhanced BAAL	- Cryptographie à seuil - Fonction de clusterisation	-Authentification et contrôle d'accès -Confidentialité des données	-Données multicast déchiffrées et re chiffrées par les CLs -Génération des clés par les nœuds serveurs	-Clés publique et privée -Bufferisation des données pour opération de déchiffrement et de re chiffrement intermédiaires pour les CLs	-Messages de requêtes et de réponses pour la génération des clés par les CLs et le CG -Message de notification au CL parent lors de la clusterisation	- Contrôleur global
TEK commune	BALADE avec OMCT	- Algorithme de clusterisation OMCT -CBIDs	-Authentification et contrôle d'accès -Confidentialité des données	-Déchiffrement et re chiffrement de la TEK par les CLs -Calcul des CBIDs -Algorithme de OMCT en $O(c^2)$, c étant le nombre de membre par cluster	- CBID - Clés locales de clusters KEKs - ACL distribuée par les CIs du groupe *(f * n/k), f : facteur de redondance des données, souhaité selon l'application*	- Signalisation de l'algorithme OMCT: CG : n+k messages, CL : c messages, MG : 1 message,	- Contrôleur global courant

n : *nombre de membres du groupe*
k : *nombre de CLs*
c : *nombre de membres moyen par cluster*

TAB. 1 – Évaluation des contributions selon les métriques de comparaison (cf. Chapitre 2)

BALADE [BCF05a, BCF06a] est un protocole de gestion de clé conçu pour sécuriser les communications de diffusion de groupe, à large échelle dans les MANETs. Le modèle de diffusion des services visés est multi-sources séquentielles (1 vers n séquentiel). Selon ce modèle, à tout instant t, il y a une et une seule source qui émet et une fois qu'elle termine, une autre source prend le relais.
L'avantage majeur de BALADE est que seule la clé TEK est chiffrée et déchiffrée et non plus les données du flux multicast. La clé TEK est renouvelée à chaque unité de données émise par la source, c'est à dire selon la sémantique du flux multicast. Ce renouvellement ainsi fait est orienté sources du groupe et tient compte de la nature de données à sécuriser et de la politique de sécurité instaurée au sein de l'application en question. BALADE améliore également l'accessibilité au flux de données pour un membre qui perd sa connectivité vers son groupe, à cause de problèmes d'énergie ou de déplacements. L'idée est d'utiliser un ticket de

ré-authentification qui apporte un gain en termes de temps d'accessibilité au flux et de nombre de messages transmis dans le réseau.

Nous avons également proposé une approche de clusterisation dynamique pour la distribution de clés de groupe, dans les réseaux ad hoc. Notre approche, basée sur l'algorithme OMCT [BCF05], prend en compte la localisation et la mobilité des nœuds et optimise la consommation de l'énergie et de la bande passante.
Le principe de base de OMCT est de diviser le groupe multicast dynamiquement en des clusters fortement corrélés géographiquement, en tenant compte de la localisation et de la mobilité des membres du groupe, tout en optimisant la consommation de l'énergie et la bande passante ; chaque cluster est géré par un contrôleur local, responsable de la gestion de la sécurité au sein de son cluster. L'algorithme OMCT est intégré dans le protocole de gestion de clé de groupe BALADE, afin d'aboutir à un processus de distribution de clés optimal.
Nous avons étudié un mécanisme d'intégration de couches protocolaires, combinant la technique de relais multipoints utilisée dans le protocole de routage OLSR, avec l'algorithme OMCT. Ce mécanisme ainsi conçu apporte une économie en calcul et en bande passante, tout en assurant un processus de distribution de clé fiable et optimisé.

Nous avons utilisé trois approches complémentaires afin de valider nos principales contributions. La première approche de validation est une vérification formelle de notre protocole de gestion de clé de groupe BALADE combiné avec OMCT [BCC+06]. Pour cela, nous avons opté pour l'outil automatique de validation formelle AVISPA (*Automated Validation of Internet Security Protocols and Applications*), avec son langage de spécification formelle HLPSL (*High-Level Protocol Specification Langage*). Cette approche nous a permis de détecter une faille de sécurité au sein d'un sous-protocole de BALADE, "la ré-intégration au groupe" et d'y remédier.
Ensuite et afin de valider l'applicabilité réelle de BALADE, nous l'avons implanté au sein d'une application de Jukebox coopérative, appelée JDukebox. Cette application consiste en une diffusion multicast de titres MP3 pour un ensemble de nœuds ad hoc. Chaque nœud ayant une liste de titres MP3 la déclare au sein du réseau ; une file d'attente de titres MP3 est ainsi distribuée à tous les nœuds. Plusieurs nœuds du réseau peuvent jouer le rôle de source du groupe multicast garantissant une émission multicast multi-sources séquentielles.
Finalement, nous avons évalué les performances de nos contributions avec le simulateur de réseaux NS2 et nous les avons comparées avec les approches existantes. Nous avons pu montrer à travers les simulations réalisées que la clusterisation selon OMCT apporte une économie en termes de délai moyen de transmissions de clés et de consommation d'énergie. Nous avons évalué également l'impact du modèle de mobilité (individuelle et de groupe) sur les performances de BALADE couplé avec OMCT. Nous avons montré, dans ce sens, que BALADE a de meilleures performances quand il est déployé avec un modèle de mobilité de groupe. La dernière étape de simulation consistait à comparer notre architecture décentralisée BALADE avec deux protocoles de gestion de clé de groupe dans les MANETs, le protocole centralisé GKMPAN et le protocole distribué DMGSA. Les résultats obtenus ont montré que BALADE détient le meilleur compromis en termes de délai moyen et de taux de transmission de clés, ainsi qu'en consommation d'énergie.

2 Perspectives

Comme perspectives à nos travaux, nous prévoyons dans un premier temps d'étendre notre architecture de gestion de clé de groupe BALADE, en définissant un système de gestion de confiance entre les contrôleurs du groupe multicast. Ce système, basé sur la notion de réputation des contrôleurs locaux, permettra d'établir des niveaux de confiance bien définis. La réputation d'un contrôleur local reflète son comportement au sein du réseau ad hoc en interaction avec les autres membres du groupe multicast. Un mécanisme d'évaluation de réputations pourrait être intégré, comme [BH01], [BB02], [MM02], ... À un niveau de confiance, lui correspond un rôle bien spécifique au sein de notre architecture. Par exemple, un contrôleur local qui détient le plus haut niveau de confiance a le droit d'inviter des entités au groupe multicast ou d'exclure des membres qu'il détecte malicieux.
Le système de gestion de confiance ainsi construit, sera ensuite intégré dans le mécanisme de gestion distribuée des listes de contrôle d'accès et de révocation des membres malicieux. Initialement, c'est le contrôleur du groupe CG qui détient les deux listes ACL et RL. Ensuite et au cours de la session du groupe multicast, le CG attribue à chaque contrôleur local un niveau de confiance lui permettant d'accéder au rôle bien spécifique correspondant à son niveau de confiance : pouvoir mettre à jour les deux listes ACL et RL ou bien seulement les consulter.

Nous prévoyons également d'approfondir nos travaux sur l'évaluation de performances de nos protocoles de gestion de clé de groupe dans les MANETs (Enhanced BAAL et BALADE). Notre objectif est d'aboutir à un modèle d'évaluation de performances uniforme, permettant de comparer les différentes approches selon des métriques bien spécifiques. Ce modèle permettra entre autres d'évaluer l'impact de la dynamicité d'adhésion des membres au groupe multicast, d'évaluer le support du passage à l'échelle ainsi que de chiffrer le surcoût de stockage, de calcul et de communications à travers des simulations réelles des protocoles étudiés.
Nous envisageons dans ce cadre d'étudier l'impact de la sécurisation des communication de groupe sur la qualité de service (QoS) fournie par l'application concernée. En effet, selon la politique de sécurité à mettre en œuvre au sein de l'application de groupe et selon la qualité de service à fournir pour les utilisateurs adhérents au groupe, doivent être définis des seuils de tolérance des surcoûts engendrés par l'architecture de sécurisation des communications de groupe mise en place. Par exemple, afin d'améliorer le temps d'accessibilité des membres au flux multicast, le surcoût de communications lors d'une authentification ne doit pas dépasser un nombre défini de messages.

Notre démarche de validation du protocole BALADE couplé avec OMCT a suivi trois étapes essentielles : sa vérification formelle avec l'outil de validation des protocoles de sécurité AVISPA, son implantation au sein d'une application de groupe réelle dans les réseaux ad hoc et l'évaluation de ses performances à travers le simulateur des réseaux NS2. Cette démarche nous a fourni un premier niveau de validation, de par les erreurs de conception ou les failles de sécurité que nous avons pu détecter. Nous prévoyons dans ce cadre d'utiliser une approche complémentaire de validation de notre protocole BALADE couplé avec OMCT, face aux contraintes et aux exigences spécifiques à l'environnement ad hoc et à l'application à sécuriser, à travers l'outil TURTLE [ACLSS04]. Cette démarche permettrait la validation formelle orientée "contrôle" et "contraintes temporelles" de l'applicabilité de BALADE dans les réseaux ad hoc, dans le cadre des applications spécifiques au sein desquelles il est déployé. La méthodologie TURTLE commence par recueillir les exigences temporelles et les contraintes à satisfaire par le système étudié. Un diagramme des cas d'utilisation est ainsi construit, déli-

mitant le périmètre du système et l'isolant de ses acteurs externes. Ce diagramme est ensuite traduit en un diagramme de classes / objets et de diagrammes d'activité décrivant respectivement l'architecture du système et le comportement des objets. La dernière étape de l'approche de validation par TURTLE consiste à traiter pas à pas les cas d'utilisation, en commençant par une situation où le service est supposé parfait et en introduisant ensuite des fonctionnements dégradés telles que les pertes de paquets, les délais importants d'acheminement de messages, l'asynchronisation entre la source et les récepteurs, ... La traçabilité des exigences constitue le résultat de la validation par TURTLE ; elle permet de mettre en relation les exigences initiales avec les résultats du processus de vérification.

Nous nous sommes focalisés durant ces trois années de thèse sur le thème de la sécurité des communications de groupe dans les réseaux ad hoc. Ce thème de recherche se situe dans le cadre d'un domaine plus général qui est la sécurité dans les réseaux spontanés. Les réseaux spontanés permettent à un ensemble de machines hôtes d'être connectées facilement et rapidement entre elles avec un minimum d'infrastructure préalable, voire sans infrastructure. Chaque nœud contribue activement à la vie du réseau soit en collaborant pour acheminer les données à destination, soit en acceptant d'être à la fois client et fournisseur de contenu. Les réseaux pair à pair (P2P) et les réseaux intelligents sont des illustrations de ce concept de spontanéité.
Nous prévoyons dans ce cadre d'adapter BALADE, notre protocole de gestion de clé de groupe dédié aux MANETs, au contexte des réseaux pair à pair et des réseaux intelligents et plus particulièrement des réseaux domestiques, en tenant compte de leurs particularités et de leurs caractéristiques.

Références

Bibliographie

[AA96] K. Almeroth and M. Ammar. Collecting and modelling the join-leave behaviour of multicast group members in the Mbone. In *The Symposium on High Performance Distributed Computing*, pages 209–216, Syracuse NY, 1996.

[ABB+05] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuellar, P.H. Drielsma, P.C. Héam, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santos Santiago, M. Turuani, L. Viganò, and L. Vigneron. The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications. In *17th International Conference on Computer Aided Verification, CAV*, volume 3576 of *Lecture Notes in Computer Science*, pages 281–285, Edinburgh, Scotland, 2005. Springer.

[ACLSS04] L. Apvrille, J.P. Courtiat, C. Lohr, and P. De Saqui-Sannes. TURTLE : A Real-Time UML Profile Supported by a Formal Validation Toolkit. *IEEE Transactions on Software Engineering (TSE)*, 30(7) :473–487, 2004.

[AG00] N. Asokan and P. Ginzboorg. Key-Agreement in Ad-hoc Networks. *Computer Communications*, 23(17) :1627–1637, February 2000.

[ANL01] T. Aura, P. Nikander, and J. Leiwo. DOS-Resistant Authentication with Client Puzzles. *Lecture Notes in Computer Science*, 2133 :170, 2001.

[BB02] S. Buchegger and J. Le Boudec. Nodes Bearing Grudges : Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002. IEEEComputer Society.

[BBC02] H. Bettahar, A. Bouabdallah, and Y. Challal. An Adaptive Key Management Protocol for Secure Multicast. In *11th International Conference on Computer Communications and Networks ICCCN*, Florida USA, October 2002.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Keying Hash Functions for Message Authentication. *RSA CryptoBytes*, 2(1), 1996.

[BH01] L. Buttyan and J. Hubaux. Nuglets : a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, 2001.

[CBB04b] 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), April 2004.

[CCC+04] Y. Chevalier, L. Compagna, J. Cuellar, P. Drielsma, J. Mantovani, S. Modersheim, and L. Vigneron. A High Level Protocol Specification Language for Indus-

trial Security-Sensitive Protocols. In *Workshop on Specification and Automated Processing of Security Requirements (SAPS)*, 2004.

[CCS00] G. Chaddoud, I. Chrisment, and A. Schaff. BAAL : Sécurisation des communications de groupes dynamiques. In *Proceedings of the 8th Colloque Francophone sur l'Ingénierie des Protocoles CFIP'2000*, Toulouse, France, October 2000.

[CH03] T. Chiang and Y. Huang. Group Keys and the Multicast Security in Ad Hoc Networks. In *Proceedings of the 2003 International Conference on Parallel Processing Workshops (ICPP Workshops)*, page 385, 2003.

[Che04] G. Chelius. *Architectures et communications dans les réseaux spontanés sans-fil.* PhD thesis, INSA de Lyon, April 2004.

[CJA+03] T. Clausen, P. Jacquet, C. Adjih, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L.Viennot. Optimized Link State Routing Protocol. RFC 3626, October 2003. Network Working Group.

[CMS02] Juan-Carlos Cano, Pietro Manzoni, and Miguel Sanchez. The Impact of Group Mobility on the Optimization of Mobile Ad hoc Networks Routing Protocols. Technical report, Polytechnic University of Valencia, Spain, 2002.

[CS05] Y. Challal and H. Seba. A taxonomy of Group key Management Protocols : issues and solutions. In *International Conference on Information Security (ICIS), Volume 6*, pages 5–17, 2005.

[Dee91] S. Deering. Multicast Routing in a Datagram Internetwork. Thèse de Doctorat, Standford University, December 1991.

[DEF94] S. Deering, D. Estrin, and D. Farinacci. An Architecture for Wide-area Multicast Routing. In *ACM SIGCOMM*, pages 126–135, August 1994.

[DH76] W. Diffie and M.E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6) :644–654, 1976.

[Die04] M. Dietzfelbinger. Gossiping and broadcasting versus computing functions in networks. *Discrete Appl. Math.*, 137(2) :127–153, 2004.

[DMS99] L. Dondeti, S. Mukherjee, and A. Samal. Secure one-to-many Group Communication Using Dual Encryption. *Computer Communications Journal*, 23(17) :1681–1701, November 1999.

[DY83] D. Dolev and A. Yao. On the Security of Public Key Protocols. *IEEE Transaction on Information Theory*, 29(2) :198–208, 1983.

[Fra02] Frank Stajano. *Security for Ubiquitous Computing.* John Wiley and Sons, February 2002.

[GND+03] V. Gayraud, L. Nuaymi, F. Dupont, S. Gombault, and B. Tharon. La sécurité dans les réseaux sans fil ad hoc. In *Symposium sur la Sécurité des Technologies de l'Information et de la Communication SSTIC*, Rennes France, June 2003.

[HBC01] J. Hubaux, L. Buttyan, and S. Capkun. The Quest for Security in Mobile Ad Hoc Networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Long Beach, CA, 2001.

[HD03] T. Hardjono and L. Dondeti. *Multicast and Group Security.* Computer Security Series. Artech House, Librarie Eyrolles, 2003.

[HFPS99] R. Housley, W. Ford, W. Polk, and D. Solo. RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile, January 1999.

[ITW82] I. Ingemarson, D. Tang, and C. Wong. A Conference Key Distribution System. *IEEE Transactions on Information Theory*, 28(5) :714–720, September 1982.

[KLG06] J. Kong, Y. Lee, and M. Gerla. Distributed Multicast Group Security Architecture for Mobile Ad-hoc Networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Las Vegas, Nevada, USA, April 2006.

[KLNY03] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure Multicast Groups on Ad hoc Networks. In *Proceedings of the 1st ACM workshop on security of ad hoc and sensor networks*, pages 94–102. ACM Press, 2003.

[LBC+01] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris. Capacity of Ad Hoc wireless networks. In *Mobile Computing and Networking*, pages 61–69, 2001.

[Leg03] V. Legrand. Rapport de DEA, Etablissement de la Confiance et Réseaux Ad Hoc. Le Germe de Confiance, EDIIS, Laboratoire CITI, INRIA ARES, Juillet 2003.

[LJM+03] A. Laouiti, P. Jacquet, P. Minet, L. Viennot, T. Clausen, and C. Adjih. Multicast Optimized Link State Routing. Research Report 4721, INRIA, February 2003.

[LL00] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Department of Computer Science, UCLA, 2000.

[LP03] L. Lazos and R. Poovendram. Energy-Aware Secure Multicast Communication in Ad Hoc Networks Using Geographical Location Information. In *IEEE International Conference on Acoustics Speech and Signal Processing*, pages 201–204, 2003.

[Mac67] J.B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, Berkeley, University of California Press, 1967.

[MC02] G. Montenegro and C. Castelluccia. Statistically Unique and Cryptographically Verifiable Identifiers and Addresses. In *ISOC Network and Distributed System Security Symposium (NDSS)*, February 2002.

[Mit97] S. Mittra. IOLUS : A Framework for Scalable Secure Multicasting. In *SIGCOMM*, pages 277–288, 1997.

[MM02] Pietro Michiardi and Refik Molva. Core : a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Proceedings of the IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security*, pages 107–121, Deventer, The Netherlands, The Netherlands, 2002. Kluwer, B.V.

[MM03] P. Michiardi and R. Molva. Ad hoc Network Security. *ST Journal of System Research*, 4(1), March 2003.

[MME04] M. Moharrun, R. Mukkalamala, and M. Eltoweissy. CKDS : An Efficient Combinatorial Key Distribution Scheme for Wireless Ad Hoc Networks. In *IEEE International Conference on Performance, Computing and Communications (IPCCC)*, Arizona, April 2004.

[Moy94] J. Moy. Multicast Routing Extension for OSPF. *Communications of the ACM*, 37(8) :61–66, August 1994.

[MS98] D. Mcgrew and A. Sherman. Key Establishement in Large Dynamic Groups Using One-way Functions Trees, May 1998.

[MSK+93] B. Mah, S. Seshan, K. Keeton, R. Katz, and D. Ferrari. Providing Network Video Service to Mobile Clients. In *Workshop on Workstation Operating Systems*, pages 48–54, 1993.

[NBCF95] N.Davies, G. Blair, K. Cheverst, and A. Friday. Supporting Collaborative Applications in a Heterogeneous Mobile Environment. *Computer Communication on Mobile Computing*, 1995.

[PCTS02] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA Broadcast Authentication Protocol. *RSA Laboratories Cryptobytes*, 5(2), 2002.

[PML+03] R. Pietro, L. Mancini, Y. Law, D. Etalle, and P. Havinga. LKHW : A Directed Diffusion Based Secure Multicast Scheme for Wireless Sensor Networks. In *International Conference on Parallel Processing Workshops (ICPPW)*, Taiwan, October 2003.

[Prü18] H. Prüfer. Neuer Beweis eines satzes über Permutationen. *Archive der Mathematik und physik*, 27, 1918.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications SIGCOMM*, pages 161–172, New York, NY, USA, 2001. ACM Press.

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

[RP00] E. Royer and C. Perkins. Multicast Ad hoc On-Demand Distance Vector (MAODV) routing, IETF Internet Draft : draft-ietf-manet-maodv-00.txt, 2000.

[SA99] F. Stajano and R. Anderson. The Resurrecting Duckling : Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop Proceedings*, pages 172–194, September 1999.

[SK98] J. Sadowsky and V. Kafedziski. On the Correlation and Scattering Functions of the WSSUS Channel for Mobile Communications. *IEEE Transactions On Vehicular Technology*, 47(1), February 1998.

[SR04] T. Samuel and J. Redwine. A Logic for the Exclusion Basis System. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS), Track 9*, Washington, DC, USA, 2004.

[Tea05] AVISPA Team. AVISPA v1.0 User Manual, Automated Validation of Internet Security Protocols and Applications, June 2005.

[Tri03] . Trincherini. Les applications ad hoc. Mémoire de Licence en Informatique de Gestion, 2003.

[VHS01] V. Varadharajan, M. Hitchens, and R. Shankaran. Securing NTDR AD-Hoc Networks. In *IASTED International Conference on Parallel and Distributed Computing and Systems 2001*, pages 593–598, Anaheim California, August 2001.

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

[WNE00] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides. On the Construction of Energy-Efficient Broadcast and Multicast Trees in Wireless Networks. In *INFOCOM 2000*, pages 585–594, Tel-Aviv, Israel, March 26-30 2000.

[WNE01] J.E. Wieselthier, G.D. Nguyen, and A. Ephremides. Algorithms for energy-efficient multicasting in static ad hoc wireless networks. *Mobile Networks and Applications*, 6(3) :251–263, 2001.

[YK02] S. Yi and R. Kravets. Key Management for Heterogeneous Ad Hoc Wireless Networks. Report Research UIUCDCS-R-2002-2290, UILU-ENG-2002-1734, University of Illinois at Urbana-Champaign, Department of Computer Science, 1304 West Springfield Avenue, Urbana, IL 61801-2987 USA, July 2002.

[YLN03] J. Yoon, M. Liu, and B. Noble. Random Waypoint Considered Harmful. In *IEEE INFOCOM*, San Fransisco, California, USA, March 2003.

[ZH99] L. Zhou and J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6) :24–30, 1999.

[Zim95] P. Zimmermann. *The official PGP user's guide.* MIT Press, Cambridge, MA, USA, 1995.

[ZK04] Y. Zhu and T. Kunz. MAODV Implementation for NS-2.26 - Systems and Computing Engineering, Carleton University, Technical Report SCE-04-01, January 2004.

[ZSXJ04a] S. Zhu, S. Setia, S. Xu, and S. Jajodia. GKMPAN : An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks. In *MobiQuitous*, pages 42–51, 2004.

[ZSXJ04b] S. Zhu, S. Setia, S. Xu, and S. Jajodia. GKMPAN : An Efficient Group Rekeying Scheme for Secure Multicast in Ad Hoc Networks Technical report, February 2004.

Publications

Journaux

[BCF06a] M.S. Bouassida, I. Chrisment, and O. Festor. Group Key Management in MANETs. Accepté pour publication. *International Journal of Network Security IJNS*, 2006.

[BCF06b] M.S. Bouassida, I. Chrisment, and O. Festor. Mobility-Awareness in Group Key Management Protocols within MANETs. *Annales des Télécommunications*, 61(9-10), September-October 2006.

Conférences internationales

[BCF04] M.S. Bouassida, I. Chrisment, and O. Festor. An Enhanced Hybrid Key Management Protocol for Secure Multicast in Ad Hoc Networks. In N. Mitrou, K. Kontovasilis, G.N. Rouskas, Iliadis, and L. Merakos, editors, *Networking 2004, Third International IFIP TC6 Networking Conference*, volume 3042 of *Lecture Notes in Computer Science LNCS*, pages 725–742, Athens, Greece, May 9-14 2004. Springer.

[BCF05] M.S. Bouassida, I. Chrisment, and O. Festor. Efficient Clustering for Multicast Key Distribution in MANETs. In R. Boutaba, K. Almeroth, R. Puigjaner, S. Shen, and J.P. Black, editors, *Networking 2005, International IFIP TC6 Networking Conference*, volume 3462 of *Lecture Notes in Computer Science LNCS*, pages 138–153, Waterloo, CANADA, May 2005. Springer.

[BCF06] M.S. Bouassida, I. Chrisment, and O. Festor. Efficient Group Key Management Protocol in MANETs using the Multipoint Relaying Technique. In P. Dini and P. Lorenz, editors, *ICN/ICONS/MCL*, pages 64–75, Ile Maurice, Avril 2006.

Conférences Nationales

[BCC+06] M.S. Bouassida, N. Chridi, I. Chrisment, O. Festor, and L. Vigneron. Automatic Verification of a Key Management Architecture for Hierarchical Group Protocols. In F. Cuppens, H. Debar, D. Boulanger, and A. Gabillon, editors, *Sécurité et Architecture des Réseaux (SAR)*, Biarritz France, Mai 2006.

[BCF04] M.S. Bouassida, I. Chrisment, and O. Festor. Méthodes d'Authentification pour les Communications de Groupe : Taxonomie et évaluation dans un Environnement Ad Hoc. In A. Bouabdallah and A. Serchouni, editors, *SAR'2004, Sécurité et Architectures Réseaux, Third Conference on Security and Network Architecture*, pages 197–208, La Londe, Cote d'Azur, France, June 21-25 2004.

[BCF05a] M.S. Bouassida, I. Chrisment, and O. Festor. BALADE : Diffusion multicast sécurisée d'un flux multimédia multi-sources séquentielles dans un environnement ad hoc. In R. Castanet, editor, *CFIP Ingénierie des Protocoles, Qualité de service, Multimédia et Mobilité*, pages 531–546, Bordeaux FRANCE, March 2005. Hermès-Lavoisier.

[BCF05b] M.S. Bouassida, I. Chrisment, and O. Festor. Prise en compte de la mobilité dans le protocole de gestion de clé de groupe BALADE. In M. Achemlal and M. Maknavicius, editors, *Sécurité et Architecture des réseaux (SAR)*, pages 263–274, 2005.

Rapports de recherche

[BCF06] M.S Bouassida, I. Chrisment, and O. Festor. Validation de BALADE Protocole de gestion de clés de groupe dans les réseaux ad hoc. Rapport de Recherche 5896, INRIA, Avril 2006.

[BLCF04] M.S. Bouassida, A. Lahmadi, I. Chrisment, and O. Festor. Diffusion multicast sécurisée dans un environnement Ad-Hoc (1 vers n séquentiel). Rapport de recherche 5310, INRIA, September 2004.

Livrables de projets

[ABB+05] M. Adib, A. Bouabdallah, M.S. Bouassida, I. Chrisment, H. Ragab, and A. Serhrouchni. Analyse des algorithmes de cryptographie multicast, Rapport de contrat du projet SAFECAST, Mai 2005.

[BBC+04] A. Bouabdallah, M.S. Bouassida, Y. Challal, , and I. Chrisment. État de l'art des protocoles d'authentification dans les communications de groupe, Rapport de contrat du projet SAFECAST, Octobre 2004.

[BBCR05a] A. Bouabdallah, M.S. Bouassida, I. Chrisment, and H. Ragab. Définition d'un protocole de gestion de clés, Rapport de contrat du projet SAFECAST, Mai 2005.

[BBCR05b] A. Bouabdallah, M.S. Bouassida, I. Chrisment, and H. Ragab. État de l'art des protocoles de gestion de clés dans les communications de groupe, Rapport de contrat du projet SAFECAST, Mai 2005.

[BC06a] M.S. Bouassida and I. Chrisment. Identification et mise en œuvre des outils de simulation et de vérification, Rapport de contrat du projet SAFECAST, Juillet 2006.

[BC06b] M.S. Bouassida and I. Chrisment. Validation fonctionnelle de l'architecture SAFECAST, Rapport de contrat du projet SAFECAST, Juillet 2006.

[BCG+04] M.S. Bouassida, I. Chrisment, V. Guyot, V. Legrand, D. Raffo, , and S. Ubeda. Sécurité et réseaux ad hoc, Rapport de contrat du projet SAFARI, Avril 2004.

Séminaires

[Bou03] M.S. Bouassida. Sécurité Multicast et Réseaux Ad Hoc, Présentation à SAR 2003 - Session étudiants, Nancy France, June 2003.

[Bou04] M.S. Bouassida. An Enhanced Hybrid Key Management Protocol for Secure Multicast in Ad Hoc Networks, Présentation à ING 2004 - Session étudiants, Obernai France, June 2004.

Annexes

Annexe A

Algorithme OMCT de clusterisation dynamique

Sommaire

A.1 Version étendue de OMCT

Algorithm 3 Version améliorée de OMCT

Liste_CLs = ϕ **(Étape 1)**
Liste_nœuds = {1, 2, 3, ..., c}
i = 1 ;
j = 2 ;
while (Liste_nœuds $\neq \phi$) **do**
  Trier_Liste_nœuds(i) ; // Trier la liste de membres selon leurs distances par rapport au nœud i **(Étape 2)**
  while $d_{(i,Liste\_noeuds(j)} \leq Max\_Distance$) **do**

    if ($nb\_Liste\_membres\_locaux\_\{i\} < MAX\_THRESHOLD$) **then**

      $Liste\_noeuds = Liste\_noeuds/\{Liste\_noeuds(j)\}$; // Supprimer le nœud Liste_nœuds(j) de la liste des membres
      $Liste\_membres\_locaux\_\{i\} = Liste\_membres\_locaux\_\{i\} \cup$ {Liste_noeuds(j)} ;
      j++ ;
    else
      Break ;
    end if
  end while
  //Le nœud List_nœuds(j+1) ne peut pas être joignable directement par le nœud i, ou le cluster est déjà saturé **(Étape 3')**
  s=1 ;
  for ($l = 2$ to Liste_nœuds(j)) **do**
    if ($d_{(s,Liste\_noeuds(j+1)} > d_{(l,Liste\_noeuds(j+1)}$) **then**
      s=l ;
    end if
  end for
  CL=s ;
  // Le contrôleur élu est le plus proche du nœud j+1 **(Étape 4)**
  i=CL ;
  $Liste\_CLs = List\_CLs \cup \{CL\}$; // Ajouter le contrôleur local à la liste des CLs
  nb_CLs ++ ;
end while(Étape 5)

Version améliorée de OMCT (suite)

// Amélioration de OMCT prenant en considération le nombre de membres par cluster **(Étape 6)**

for ($t = 1$ to nb_CLs) **do**

if ($nb\_Liste\_membres\_locaux\_\{t\} < MIN\_THRESHOLD$) **then**

for ($m = 1$ to $nb\_Liste\_membres\_locaux\_\{t\}$) **do**

for ($z = t$ to nb_CLs) **do**

if ($d_{Liste\_CLs\{z\},m} \leq Max\_Distance$ & $nb\_Liste\_membres\_locaux\_\{z\} < MAX\_THRESHOLD$) **then**

$Liste\_membres\_locaux\_\{t\} = Liste\_membes\_locaux\_\{t\}/\{m\}$;
// Supprimer m de ma liste de membres locaux de t

$Liste\_membres\_locaux\_\{z\} = Liste\_membres\_locaux\_\{z\} \cup \{m\}$;
// Ajouter m à la liste de membres locaux de z
Break ;

end if
end for
end for
end if
end for

A.2 Intégration de la technique de multipoint relais dans OMCT

Algorithm 4 OMCT_Avec_MPRs (Clusterhead)

```
//(Étape 1)
Liste_CLs = {Cluster Head}
Liste_noeud = {1, 2, 3, ..., c} //c est le nombre de membres du groupe
Liste_MPRs = Liste des MPRs du CL
//(Étape 2)
for (i = 1 to Nombre_Liste_MPRs) do
  if (Liste_noeuds ≠ φ) then
    if (i ∈ groupe multicast) && (i a des membres fils appartenant au groupe) then
      Liste_CLs = Liste_CLs ∪ {i} ; // Ajouter i à la liste des contrôleurs locaux
      Liste_noeuds = List_noeuds /{membres du groupe couverts par i} ;
      // Supprimer les membres couverts par i de la liste des membres du groupe
      OMCT_avec_MPRs (i) ;
      // Exécuter récursivement l'algorithme OMCT_avec_MPRs, appliqué à i
    end if
  end if
end for
//(Étape 3)
if (Liste_noeuds ≠ φ) then
  for (j = 1 to Nombre_Liste_noeuds) do
    Calculer le facteur d'accessibilité de j : nombre de membres dans List_noeuds, atteints
    à un seul saut du noeud j
  end for

  while (Liste_noeuds ≠ φ) do
    Choisir le membre S de la liste Liste_noeuds qui déient le plus grand facteur d'acces-
    sibilité
    Liste_LCs = Liste_LCs ∪ {S} ; // ajouter S à la liste des contrôleurs locaux
    Liste_noeuds = Liste_noeuds /{membres du groupe couverts par S} ; // Supprimer
    les membres couverts par S de la liste des membres
  end while
end if
```

Annexe B

Spécification des sous protocoles de BALADE avec HLPSL

Sommaire

B.1 Renouvellement périodique de la TEK

```
%% Renouvellement_TEK

%% Premier Rôle : Le contrôleur global CG

role membre1 (CG,mg0k,CLik: agent,
              IDCG: text,
              KEK-CSG0k : symmetric_key,
              KEK-CCL : symmetric_key,
              Snd, Rcv: channel(dy))
   played_by CG def=

   local State: nat,
         TEK: symmetric_key,
         NumSeq: nat

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> TEK' := new()
           /\ Snd({IDCG.TEK'.NumSeq'}_KEK-CSG0k)
           /\ Snd({IDCG.TEK'.NumSeq'}_KEK-CCL)
           /\ State':=1
           /\ secret(TEK',id1,{CG,mg0k,CLik})
end role

%% Deuxième Rôle : mg0k appartenant à la liste des membres locaux du CG (LML_CG)

role membre2 (CG,mg0k: agent,
              KEK-CSG0k: symetric_key,
              Snd, Rcv: channel(dy))
   played_by mg0k def=

   local State: nat,
         IDCG: message,
         TEK: message,
         NumSeq: message

    const id1: protocol_id
          init State:=0

   transition
     step1. State=0 /\ Rcv({IDCG'.TEK'.NumSeq'}_KEK-CSG0k)
```

```
       =|> State'=1
           /\ secret(TEK',id1,{CG,mg0k})

end role

%% Troisième Rôle : Un contrôleur local CLik appartenant au groupe des CLs (GCL)

role membre3 (CG,CLik,mgik: agent,
              IDCL: text,
              KEK-CCL, KEK-CSGik: symetric_key,
              Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat,
         IDCG: message,
         TEK: message,
         NumSeq: message

   const id1: protocol_id
         init State:=0

   transition
     step1. State=0 /\ Rcv({IDCG'.TEK'.NumSeq'}_KEK-CCL)
       =|> Snd({IDCL.TEK'.NumSeq'}_KEK-CSGik)
         /\ State':=1
         /\ secret(TEK',id1,{CG,CLik,mgik})

end role

%% Quatrième rôle: mgik appartenant à la liste des membres locaux du CLik (LML_CLik)

role membre4 (CLik,mgik: agent,
              kek6csgik: symmetric_key,
              Snd, Rcv: channel(dy))
   played_by mgik def=

   local State: nat,
         IDCL: message,
         TEK: message,
         NumSeq: message

    const id1: protocol_id
          init State:=0

   transition
     step1. State=0 /\ Rcv({IDCL'.TEK'.NumSeq'}_KEK-CSGik)
       =|> State'=1
```

```
           /\ secret(TEK',id1,{CLik,mgik})

end role

%%The role session between the participants

role RenouvellementTEK(SC, RC: channel(dy),
                       CG,mg0k,CLik,mgik : agent,
                       KEK_CSG_0K,KEK-CCL,KEK-CSGik: symmetric_key,
                       IDCG,IDCL: text
           ) def=

   composition
     membre1(CG,mg0k,CLik,IDCG,KEK-CSG0k,KEK-CCL,SC,RC)
     /\ membre2(CG,mg0k,KEK-CSG0k,SC,RC)
     /\ membre3(CG,CLik,mgik,IDCL,KEK-CCL,KEK-CSGik,SC,RC)
     /\ membre4(CLik,mgik,KEK-CSGik,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const CG,mg0k,CLik,mgik : agent,
         KEK_CSG_0K,KEK-CCL,KEK-CSGik: symmetric_key,
         IDCG,IDCL: text

   intruder_knowledge = {CG,mg0k,CLik,mgik}

   composition
      RenouvellementTEK(Snd,Rcv,CG,mg0k,CLik,mgik,KEK_CSG_0K,KEK-CCL,KEK-CSGik,
IDCG,IDCL)

end role

goal
secrecy_of id1
end goal

environment()
```

B.2 Join d'un nouveau membre

```
%% Join

%% Premier Rôle : mgk membre voulant joindre le groupe

role membre1 (mgk,CLik: agent,
           Pub-mgk: public_key,
           imprint: text,
           CBID-mgk: text,
           Snd, Rcv: channel(dy))
   played_by mgk def=

   local State: nat,
         Puzzle-Reply: text,
         Puzzle-Request: message,
         TEK: message,
         IDCL: message,
         Passwd: message,
         KEK-CSGik: message

   const id1,id2: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd(Pub-mgk)
         /\ State':=1
     step2. State=1 /\Rcv(Puzzle-Request)
     =|> Puzzle-Reply' := new()
         /\ Snd(Puzzle-Reply'.Pub-mgk.imprint.{CBID-mgk}_inv(Pub-mgk))
         /\ State'=2
     step3. State=1 /\Rcv({IDCL'.KEK-CSGik'.TEK'.Passwd'}_Pub-mgk)
     =|> State'=2
         /\ secret(TEK,id1,{CLik,mgk})
         /\ secret(KEK-CSG,id2,{CLik,mgk})

end role

%% Deuxième Rôle : CLik contrôleur local du cluster que le nouveau membre
%% veut joindre

role membre2 (mgk,CLik,mgik: agent,
              ancienne-KEK-CSGik: symetric_key,
               IDCL: text,
               Passwd: text,
               TEK: symetric_key,
               Snd, Rcv: channel(dy))
   played_by CLik def=
```

```
   local State: nat,
         KEK-CSGik: symmetric_key,
         Pub-mgk: message,
         Puzzle-Request: text,
         Puzzle-Reply: message,
         imprint: message,
         CBID-mgk: message

   const id1,id2: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv({Pub-mgk')
       =|> Puzzle-Request' := new()
         /\Snd(Puzzle-Request)
         /\State'=1
      step2. State=1 /\ Rcv(Puzzle-Reply.Pub-mgk.imprint.{CBID-mgk}_inv(Pub-mgk))
%% Ce message n'est reçu que si le Puzzle_Reply est correct et que le CBID-mgk
est validé authentique
     =|> KEK-CSGik' := new()
         /\ Snd({IDCL.KEK-CSGik'.TEK.Passwd}_Pub-mgk)
         /\ Snd(IDCL.KEK-CSGik'}_ancienne-KEK-CSGik)
         /\ State'=2
         /\ secret(TEK,id1,{CLik,mgk})
         /\ secret(KEK-CSG,id2,{CLik,mgk,mgik})

end role

%% Troisième Rôle : mgik ancien membre appartenant au cluster géré par CLik

role membre3 (CLik,mgik: agent,
              ancienne-KEK-CSGik: symmetric_key,
              Snd, Rcv: channel(dy))
   played_by mgik def=

   local State: nat,
         IDCL: message,
         KEK-CSGik: message

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv({IDCL'.KEK-CSGik'}_ancienne-KEK-CSGik)
       =|> State':=1
         /\ secret(KEK-CSGik',id1,{CLik,mgik})
```

```
end role

%%The role session between the participants

role Join(SC, RC: channel(dy),
          CG,mgik,mgk : agent,
          KEK_CSG_iK,ancienne-KEK-CSGik,TEK: symmetric_key,
          IDCL,imprint,CBID-mgk,Passwd: text,
          Pub-mgk: Public_key
           ) def=

   composition
     membre1(mgk,CLik,Pub-mgk,imprint,CBID-mgk,SC,RC)
     /\ membre2(mgk,CLik,mgik,ancienne-KEK-CSGik,IDCL,Passwd,TEK,SC,RC)
     /\ membre3(CLik,mgik,ancienne-KEK-CSGik,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const CG,mgik,mgk : agent,
       KEK_CSG_iK,ancienne-KEK-CSGik,TEK: symmetric_key,
         IDCL,imprint,CBID-mgk,Passwd: text,
         Pub-mgk: Public_key

   intruder_knowledge = {mgk,CLik,mgik}

   composition
      Join(Snd,Rcv,CG,CG,mgik,mgk,KEK_CSG_iK,ancienne-KEK-CSGik,TEK,IDCL,imprint,
      CBID-mgk,Passwd,Pub-mgk)

end role

goal
secrecy_of id1
secrecy_of id2
end goal

environment()
```

B.3 Leave d'un membre du groupe

```
%% Leave_membre_du_groupe

%% Premier Rôle : Mgk membre voulant quitter le groupe

role membre1 (Mgk,CLik: agent,
              PubMgk: public_key,
        Imp: text,
        CBIDMgk: text,
        Snd, Rcv: channel(dy))
   played_by Mgk def=

   local State: nat

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd(PubMgk.Imp.{CBIDMgk}_inv(PubMgk))
   /\ State'=1

end role

%% Deuxième Rôle : CLik contrôleur local du cluster que le nouveau membre
%% veut quitter

role membre2 (Mgk,CLik,Mgik: agent,
  IDCL: text,
  PubMgik: public_key,
             Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat,
   KEKCSGik: symmetric_key,
   Imp2: message,
   CBIDMgk2: message,
   PubMgk2: message
   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(PubMgk2'.Imp2'.{CBIDMgk2'}_inv(PubMgk2'))
     %% Ce message n'est reçu que si le Puzzle_Reply est correct et que le
     %%CBIDMgk est validé authentique
     =|>  Snd({IDCL.KEKCSGik'}_PubMgik)
        /\ State':=1
  /\ secret(KEKCSGik',id1,{CLik,Mgik})
```

```
end role

%% Troisième Rôle : Mgik ancien membre appartenant au cluster géré par CLik

role membre3 (CLik,Mgik: agent,
        PubMgik : public_key,
              Snd, Rcv: channel(dy))
   played_by Mgik def=

   local State: nat,
         IDCL3: message,
         KEKCSGik: message

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv({IDCL3'.KEKCSGik'}_PubMgik)
       =|> State':=1
     /\ secret(KEKCSGik',id1,{CLik,Mgik})

end role

%%The role session between the participants

role leave(SC, RC: channel(dy),
           CLik,Mgik,Mgk : agent,
     IDCL,Imp,CBIDMgk: text,
     PubMgk,PubMgik: public_key
           ) def=

   composition
     membre1(Mgk,CLik,PubMgk,Imp,CBIDMgk,SC,RC)
     /\ membre2(Mgk,CLik,Mgik,IDCL,PubMgik,SC,RC)
     /\ membre3(CLik,Mgik,PubMgik,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const clik,mgik,mgk : agent,
   idcl,imp,cbidmgk: text,
   pubmgk,pubmgik: public_key
```

```
   intruder_knowledge = {mgk,clik,mgik}

   composition
      leave(Snd,Rcv,clik,mgik,mgk,idcl,imp,cbidmgk,pubmgk,pubmgik)

end role

goal
secrecy_of id1
end goal

environment()
```

B.4 Leave d'un contrôleur local du groupe

```
%% Leave_contrôleur_local_du_groupe

%% Premier Rôle : CLik membre voulant quitter le groupe

role membre1 (CLik,MGik,CG: agent,
              KEKCSGik,KEKCCL: symmetric_key,
        IDCLik: text,
        Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd({IDCLik}_KEKCSGik)
           /\ Snd(IDCLik}_KEKCCL)
              /\ State'=1

end role

%% Deuxième Rôle : CG contrôleur global du groupe

role membre2 (CLik,CG,CLjk: agent,
  IDCG: text,
  KEKCCL: symmetric_key,
  PubCLjk: public_key,
```

```
              Snd, Rcv: channel(dy))
     played_by CG def=

   local State: nat,
   NEWKEKCCL: symmetric_key,
   IDCLik2: message
   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv((IDCLik2}_KEKCCL))
     =|>  NEWKEKCCL' = new()
        /\Snd({IDCG.NEWKEKCCL'}_PubCLjk)
        /\ State':=1
  /\ secret(NEWKEKCCL',id1,{CG,CLjk})

end role

%% Troisième Rôle : Mgik membre appartenant au cluster géré par CLik

role membre3 (CLik,Mgik: agent,
        KEKCSGik : public_key,
              Snd, Rcv: channel(dy))
   played_by Mgik def=

   local State: nat,
         IDCLik2: message,

   init State:=0

   transition
     step1. State=0 /\ Rcv({IDCLik2}_KEKCSGik)
       =|> State':=1

end role

%% Quatrième Rôle : CLjk contrôleur local appartenant au groupe GCL, j<>i

role membre4 (CLjk,CG: agent,
            PubCLjk: public_key,
              Snd, Rcv: channel(dy))
   played_by CLjk def=

   local State: nat,
         IDCG2: message,
       NEWKEKCCL2: message
```

```
   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv({IDCG2',NEWKEKCCL2'}_PubCLjk)
       =|> State':=1
           /\ secret(NEWKEKCCL2,id1,{CG,CLjk})

end role


%%The role session between the participants

role leaveCL(SC, RC: channel(dy),
             CLik,CLjk,Mgik,CG : agent,
      IDCLik,IDCG: text,
      PubCLjk: public_key,
      KEKCCL,KEKCSGik: symmetric_key
           ) def=

   composition
     membre1(CLik,MGik,CG,KEKCSGik,KEKCCL,IDCLik,SC,RC)
     /\ membre2(CLik,CG,CLjk,IDCG,KEKCCL,PubCLjk,SC,RC)
     /\ membre3(CLik,Mgik,KEKCSGik,SC,RC)
     /\ membre3(CLjk,CG,PubCLjk,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const clik,cljk,mgik,cg : agent,
   idclik,idcg: text,
   pubcljk: public_key,
   kekccl,kekcsgik: symmetric_key

   intruder_knowledge = {mgik,clik,cljk,cg}

   composition
      leaveCL(Snd,Rcv,clik,cljk,mgik,cg,idclik,idcg,pubcljk,kekccl,kekcsgik)

end role

goal
secrecy_of id1
end goal
```

```
environment()
```

B.5 Exclusion d'un membre du groupe

```
%% Exclusion
%% La partie Renouvellement de la TEK ne sera pas vérifiée ici puiqu'elle est
%% similaire au sous protocole de renouvellement périodique de la TEK,
%% validé précedemment

%% Premier Rôle : CLik contrôleur local du membre exclu

role membre1 (CLik,CG,Mgik: agent,
              IDCLik: text,
       PubCLik,PubMgik: public_key,
     Imp:text,
     CBIDCLik: text,
     Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat,
         KEKCSGik: symmetric_key

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> KEKCSGik' := new()
           /\ Snd({IDCLik.KEKCSGik}_PubMgik)
           /\ Snd(IDCLik.PubCLik.Imp.{CBIDCLik}_inv(PubCLik))
           /\ State':=1
     /\ secret(KEKCSGik,id1,{CLik,Mgik})
end role

%% Deuxième Rôle : Mgik appartenant à la liste des membres locaux du CLik

role membre2 (CLik,Mgik: agent,
       PubMgik: public_key,
       Snd, Rcv: channel(dy))
   played_by Mgik def=

   local State: nat,
   IDCLik2: message,
```

```
   KEKCSGik2: message

    const id1: protocol_id
          init State:=0

   transition
     step1. State=0 /\ Rcv({IDCLik2'.KEKCSGik2'}_PubMgik)
       =|> State'=1
           /\ secret(KEKCSGik2,id1,{CLik,Mgik})

end role

%% Troisième Rôle : CG contrôleur global du groupe

role membre3 (CG,CLik: agent,
              Snd, Rcv: channel(dy))
   played_by CG def=

   local State: nat,
         IDCLik3: message,
   PubCLik3: public_key,
   Imp3: message,
   CBIDCLik2: message

   const id1: protocol_id
         init State:=0

   transition
     step1. State=0 /\ Rcv(IDCLik3'.PubCLik3'.Imp3'.
{CBIDCLik2'}_inv(PubCLik3'))
       =|> State':=1
%% Renouvellement de la TEK par le CG

end role

%%The role session between the participants

role exclu(SC, RC: channel(dy),
           CG,CLik,Mgik : agent,
    IDCLik,Imp,CBIDCLik: text,
    PubMgik,PubCLik: public_key
           ) def=

   composition
     membre1(CLik,CG,Mgik,IDCLik,PubCLik,PubMgik,Imp,CBIDCLik,SC,RC)
     /\ membre2(CLik,Mgik,PubMgik,SC,RC)
     /\ membre3(CG,CLik,SC,RC)
```

```
end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const cg,clik,mgik : agent,
   idclik,imp,cbidclik: text,
   pubmgik,pubclik: public_key

   intruder_knowledge = {cg,clik,mgik}

   composition
      exclu(Snd,Rcv,cg,clik,mgik,idclik,imp,cbidclik,pubmgik,pubclik)

end role

goal
secrecy_of id1
end goal

environment()
```

B.6 Envoi périodique des CL_Queries

```
%% CLQUERY

%% Premier Rôle : CLik contrôleur local du groupe

role membre1 (CLik,MGk: agent,
              IDCLik: text,
       PubCLik: public_key,
       Imp: text,
       CBIDCLik: text,
       COORDCLik: text,
       Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat

   init State:=0

   transition
```

```
     step1. State=0 /\ Rcv(start)
       =|> Snd(IDCLik.PubCLik.Imp.{CBIDCLik}_inv(PubCLik).COORDCLik)
           /\ State':=1

end role

%% Deuxième Rôle : MGk membre du groupe à deux sauts du CLik

role membre2 (CLik,MGk: agent,
       Snd, Rcv: channel(dy))
     played_by MGk def=

   local State: nat,
   IDCLik2: message,
   PubCLik2: public_key,
   Imp2: message,
   CBIDCLik2: message,
   COORDCLik2: message

   init State:=0

   transition
     step1. State=0 /\ Rcv(IDCLik2'.PubCLik2'.Imp2'.
{CBIDCLik2'}_inv(PubCLik2').COORDCLik2')
     %%Ce message n'est reçu que si le CBID du Clik est validé authentique
      =|> State':=1

end role

%%The role session between the participants

role query(SC, RC: channel(dy),
          CLik,MGk: agent,
    IDCLik: text,
    PubCLik: public_key,
    Imp: text,
    CBIDCLik: text,
    COORDCLik: text
           ) def=

   composition
     membre1(MGk,CLik,IDCLik,PubCLik,Imp,CBIDCLik,COORDCLik,SC,RC)
     /\ membre2(MGk,CLik,SC,RC)

end role

%%The main role
```

```
role environment() def=
   local Snd, Rcv: channel(dy)
   const clik,mgk: agent,
   idclik: text,
   pubclik: public_key,
   imp: text,
   cbidclik: text,
   coordclik: text

   intruder_knowledge = {clik,mgk}

   composition
      query(Snd,Rcv,clik,mgk,idclik,pubclik,imp,cbidclik,coordclik)

end role

environment()
```

B.7 Gestion des listes ACL et RL

```
%% Demande d'autorisation d'accès

%% First Role: CLik contrôleur local qui demande l'autorisation d'accès
%% d'un membre

role membre1 (CLik,CLrk: agent,
              KEKCCL: symmetric_key,
       IDCLik: text,
       MGnew: text,
       Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat,
         IDCLrk2: message,
       Result2: message

   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd({IDCLik.MGnew}_KEKCCL)
           /\ State':=1
     step2. State=1 /\ Rcv({IDCLrk2'.MGnew.Result2'}_KEKCCL)
```

```
       =|> State':=2

end role

%% The second role: CLrk contrôleur local qui détient l'information ACL

role membre2 (CLik,CLrk: agent,
    KEKCCL: symmetric_key,
 IDCLrk:text,
 Snd, Rcv: channel(dy))
   played_by CLrk def=

   local State: nat,
         Result: text,
         IDCLk2: message,
      MGnew2: message

    const id1: protocol_id
    init State:=0

    transition
     step1. State=0 /\ Rcv({IDCLk2'.MGnew2'}_KEKCCL)
       =|> State'=1
           /\ Result'= new()
           /\ Snd ({IDCLrk.IDCLk2'.MGnew2'.Result'}_KEKCCL)

end role

%%The role session between the two participants

role acl(SC, RC: channel(dy),
        CLik,CLrk: agent,
        KEKCCL: symmetric_key,
  IDCLik,IDCLrk: text,
  MGnew: text
    ) def=

   composition
     membre1(CLik,CLrk,KEKCCL,IDCLik,MGnew,SC,RC)
     /\ membre2(CLik,CLrk,KEKCCL,IDCLrk,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
```

```
   const clik,clrk: agent,
         kekccl: symmetric_key,
   idclik,idclrk: text,
   mgnew: text

   intruder_knowledge = {clik,clrk}

   composition
      acl(Snd,Rcv,clik,clrk,kekccl,idclik,idclrk,mgnew)

end role

environment()
```

B.8 Retransmission de la TEK par les contrôleurs de BALADE

```
%% TEK retransmission

%% First Role: Mgik membre du groupe qui demande à son CLik de lui
%% transmettre la TEK

role membre1 (MGik,CLik: agent,
              IDMGik: text,
       KEKCSGik: symmetric_key,
       Snd, Rcv: channel(dy))
   played_by MGik def=

   local State: nat,
     IDCL2: message,
 TEK2: message,
 NUMseq2: message

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd({IDMGik}_KEKCSGik)
         /\ State':=1
     step2. State=1 /\ Rcv({IDCL2'.TEK2'.NUMseq2'}_KEKCSGik)
       =|> State':=2
         /\ secret(TEK2,id1,{MGik,CLik})

end role
```

```
%% The second role: CLik contrôleur local de MGik

role membre2 (MGik,CLik: agent,
       IDCL,NUMseq: text,
       TEK,KEKCSGik: symmetric_key,
       Snd, Rcv: channel(dy))
    played_by CLik def=

    local State: nat,
          IDMGik2: message

    const id1: protocol_id
    init State:=0

    transition
     step1. State=0 /\ Rcv({IDMGik2'}_KEKCSGik)
       =|> State'=1
         /\ Snd ({IDCL.TEK.NUMseq}_KEKCSGik)
   /\secret(TEK,id1,{MGik,CLik})

end role

%%The role session between the two participants

role tek(SC, RC: channel(dy),
        MGik,CLik: agent,
        IDMGik,IDCL,NUMseq: text,
  KEKCSGik,TEK: symmetric_key
    ) def=

   composition
     membre1(MGik,CLik,IDMGik,KEKCSGik,SC,RC)
     /\ membre2(MGik,CLik,IDCL,NUMseq,TEK,KEKCSGik,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const mgik,clik: agent,
         idmgik,idcl,numseq: text,
   kekcsgik,tek: symmetric_key

   intruder_knowledge = {mgik,clik}
```

```
   composition
      tek(Snd,Rcv,mgik,clik,idmgik,idcl,numseq,kekcsgik,tek)

end role

goal
secrecy_of id1
end goal

environment()
```

B.9 Clusterisation avec OMCT

```
%% Renouvellement_TEK

%% Premier Rôle : Le contrôleur global CG

role membre1 (CG,Mgok,CLik: agent,
  Clusterisation: text,
        KEKCSG0k: symmetric_key,
    PubCGk,PubCLik: public_key,
  Snd, Rcv: channel(dy))
   played_by CG def=

   local State: nat,
     KEKCCL: symmetric_key,
     IDmg0k2: message,
     Coordmg0k2: message,
     LMLCLik: text

   const id1: protocol_id
   init State:=0

   transition
     step1. State=0 /\ Rcv(start)
       =|> Snd({Clusterisation}_KEKCSG0k)
           /\ State':=1
     step2. State=1 /\ Rcv({IDmg0k2'.Coordmg0k2'}_PubCGk)
 =|> State':=2
     /\ KEKCCL' := new()
     /\ LMLCLik' := new()
     /\ Snd({LMLCLik'.KEKCCL'}_PubCLik)
     /\ secret(KEKCCL',id1,{CG,CLik})
```

```
end role

%% Deuxième Rôle : Mgok appartenant à la liste des membres locaux du CG (LML_CG)

role membre2 (CG,Mgok: agent,
  KEKCSG0k: symmetric_key,
  PubCGk: public_key,
  IDmg0k: text,
              Snd, Rcv: channel(dy))
     played_by Mgok def=

   local State: nat,
   Clusterisation2: message,
   Coordmg0k: text

    const id1: protocol_id
    init State:=0

   transition
     step1. State=0 /\ Rcv({Clusterisation2'}_KEKCSG0k)
       =|> State'=1
     /\ Coordmg0k' := new()
     /\ Snd({IDmg0k.Coordmg0k'}_PubCGk)

end role

%% Troisième Rôle : Un contrôleur local CLik élu par le CG

role membre3 (CG,CLik,Mgik: agent,
              PubCLik: public_key,
       Imp,CBIDCLik: text,
       Snd, Rcv: channel(dy))
   played_by CLik def=

   local State: nat,
    LMLCLik2: message,
    KEKCCL2: message,
          IDcluster: text,
    KEKCSGik: symmetric_key,
    Pubmgik: public_key

   const id1,id2: protocol_id
         init State:=0

   transition
     step1. State=0 /\ Rcv({LMLCLik2'.KEKCCL2'}_PubCLik)
         =|>  State':=1
```

```
     /\ secret(KEKCCL2,id1,{CG,CLik})
     /\ IDcluster' := new()
     /\ KEKCSGik' := new()
     /\ Pubmgik' := new()
     /\ Snd({IDcluster'.KEKCSGik'.PubCLik.Imp.{CBIDCLik}_inv(PubCLik)}_Pubmgik')
     /\ secret(KEKCSGik,id2,{CLik,Mgik})

end role

%% Quatrième rôle: Mgik appartenant à la liste des membres locaux du CLik
%% (LML_CLik)

role membre4 (CLik,Mgik: agent,
  PubMgik: public_key,
              Snd, Rcv: channel(dy))
     played_by Mgik def=

   local State: nat,
   IDcluster2: message,
   KEKCSGik2: message,
   PubCLik2: public_key,
   Imp2,CBIDCLik2: text

    const id2: protocol_id
          init State:=0

   transition
     step1. State=0 /\ Rcv({IDcluster2'.KEKCSGik2'.PubCLik2'.Imp2'.
   {CBIDCLik2'}_inv(PubCLik2')}_PubMgik)
       =|> State'=1
           /\ secret(KEKCSGik2,id2,{CLik,Mgik})

end role

%%The role session between the participants

role omct(SC, RC: channel(dy),
         CG,MgOk,CLik,Mgik: agent,
   Clusterisation: text,
   KEKCSGOk: symmetric_key,
     PubCGk,PubCLik,PubMgik: public_key,
         IDmgOk: text,
   Imp,CBIDCLik: text
         ) def=

   composition
     membre1(CG,MgOk,CLik,Clusterisation,KEKCSGOk,PubCGk,PubCLik,SC,RC)
```

```
     /\ membre2(CG,Mg0k,KEKCSG0k,PubCGk,IDmg0k,SC,RC)
     /\ membre3(CG,CLik,Mgik,PubCLik,imp,CBIDCLik,SC,RC)
     /\ membre4(CLik,Mgik,PubMgik,SC,RC)

end role

%%The main role

role environment() def=
   local Snd, Rcv: channel(dy)
   const cg,mg0k,clik,mgik : agent,
         clusterisation: text,
   kekcsg0k: symmetric_key,
     pubcgk,pubclik,pubmgik: public_key,
         idmg0k: text,
   imp,cbidclik: text

   intruder_knowledge = {cg,mg0k,clik,mgik}

   composition
      omct(Snd,Rcv,cg,mg0k,clik,mgik,clusterisation,kekcsg0k,pubcgk,
           pubclik,pubmgik,idmg0k,imp,cbidclik)

end role

goal
secrecy_of id1
secrecy_of id2
end goal

environment()
```

Annexe C

Simulation avec NS2

Sommaire

C.1 Agents NS2 des protocoles simulés

C.1.1 Agent BALADE

```
#ifndef ns_balade_h
#define ns_balade_h
#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"
#define CL_group 234881024  // adresse multicast des contrôleurs locaux
#define t_head 10           // Temps de reclusturisation
#define rayon 2             // rayon (nombre de sauts) de la clusturisation (utilisé
                               dans l'agent AODV)
#define tail_rq_kek 548     // Taille de la requete de demande de la kek
#define tail_rep_kek 720    // Taille de la reponse a la demande de la kek
#define tail_tek 712        // Taille du message de diffusion de la TEK
#define temp_kek 0.5        // Temps necessaire pour générer, coder et décoder la
                              kek
#define temp_renouv_tek  200 // période séparant deux renouvellements
                                successifs de la TEK

class BALADEAgent;

class CL_timer : public TimerHandler {
public:
CL_timer(BALADEAgent *a) : TimerHandler() { a_ = a; }
protected:
virtual void expire(Event *e);
BALADEAgent *a_;
};

class kek_timer : public TimerHandler {
 public:
  kek_timer(BALADEAgent *a) : TimerHandler() { a_ = a; }
 protected:
  virtual void expire(Event *e);
  BALADEAgent *a_;
};

// Timer de renouvellement périodique de la clé de groupe (TEK)
// le message de renouvellement est envoyé du contrôleur global aux contrôleurs locaux
class cl_tek_timer : public TimerHandler {
 public:
  cl_tek_timer(BALADEAgent *a) : TimerHandler() { a_ = a; }
```

```
 protected:
  virtual void expire(Event *e);
  BALADEAgent *a_;
};

// Entête BALADE
struct hdr_BALADE {
        int group;
        char new_chef;
// methodes d'acces a l'entete
static int offset_; //utilise par le gestionnaire d'entetes
inline static int& offset() { return offset_; }
inline static hdr_BALADE* access(const Packet* p) {
return (hdr_BALADE*) p->access(offset_);}
};

// Classe de l'Agent BALADE
class BALADEAgent : public Agent {
 public:
  BALADEAgent();
  int command(int argc, const char*const* argv);
  void recv(Packet*, Handler*);
  void form_cluster (char maintien);
  void rep_kek (void);
  void cl_diff_tek (void);
  int actual_group ;
 protected:
  char cluster;         // Booleen : 1 si le noeud apprtient a un cluster.
                           0 sinon
  char cl;              // Booleen : 1 si le noeud est chef de cluster. 0 sinon
  CL_timer CL_timer_;
  kek_timer kek_timer_;
  cl_tek_timer cl_tek_timer_ ;
};

#endif
```

C.1.2 Agent DMGSA

```
#ifndef ns_dmgsa_h
#define ns_dmgsa_h

#include "agent.h"
#include "tclcl.h"
#include "packet.h"
```

```
#include "address.h"
#include "ip.h"

#define GCKS_group 234881024  // adresse multicast des chefs de clusters (GCKS)
#define t_head 6              // Temps de reclusturisation
#define rayon 3               // rayon (nombre de sauts) de la clusturisation
                                 (utilisé dans l'agent AODV)
#define tail_rq_kek 548       // Taille de la requete de demande de la kek
#define tail_rep_kek 720      // Taille de la reponse a la demande de la kek
#define tail_tek 712          // Taille du message de diffusion de la TEK
#define temp_kek 0.5          // Temps necessaire pour generer, coder et decoder la kek
#define temp_renouv_tek  200  // periode separant deux renouvellement successifs
                                 de la TEK

class DMGSAAgent;

// Timer GCKS utilisé par le chef de cluster pour la diffusion paquet Cluster-Head
class GCKS_timer : public TimerHandler {
public:
GCKS_timer(DMGSAAgent *a) : TimerHandler() { a_ = a; }
protected:
virtual void expire(Event *e);
DMGSAAgent *a_;
};

// Timer de génération de la kek pour l'envoyer aux nouveaux membres de groupe
class kek_timer : public TimerHandler {
 public:
  kek_timer(DMGSAAgent *a) : TimerHandler() { a_ = a; }
 protected:
  virtual void expire(Event *e);
  DMGSAAgent *a_;
};

// Timer de renouvellement periodique de la cle de groupe (TEK)
// le message de renouvellement est envoyé d'un chef de groupe aux autres chefs
class gcks_tek_timer : public TimerHandler {
 public:
  gcks_tek_timer(DMGSAAgent *a) : TimerHandler() { a_ = a; }
 protected:
  virtual void expire(Event *e);
  DMGSAAgent *a_;
};

// Entete Dmgsa
struct hdr_DMGSA {
        int group;
```

```
        char new_chef;
// methodes d'acces a l'entete
static int offset_; //utilise par le gestionnaire d'entetes
inline static int& offset() { return offset_; }
inline static hdr_DMGSA* access(const Packet* p) {
return (hdr_DMGSA*) p->access(offset_);}
};

// Classe de l'Agent DMGSA
class DMGSAAgent : public Agent {
 public:
  DMGSAAgent();
  int command(int argc, const char*const* argv);
  void recv(Packet*, Handler*);
  void form_cluster (char maintien);
  void rep_kek (void);
  void gcks_diff_tek (void);
  int actual_group ;
 protected:
  char cluster;           // Booleen : 1 si le noeud apprtient a un cluster.
                             0 sinon
  char gcks;              // Booleen : 1 si le noeud est chef de cluster. 0 sinon
  GCKS_timer GCKS_timer_;
  kek_timer kek_timer_;
  gcks_tek_timer gcks_tek_timer_ ;
};

#endif
```

C.1.3 Agent GKMPAN

```
#ifndef ns_gkmpan_h
#define ns_gkmpan_h

#include "agent.h"
#include "tclcl.h"
#include "packet.h"
#include "address.h"
#include "ip.h"

#define tail_tek 712     // Taille du message de diffusion de la TEK
#define temp_renouv_tek  200     // periode separant deux renouvellement
                                    successifs de la TEK

class GKMPANAgent;
```

```
// Timer de renouvellement periodique de la cle de groupe (TEK)
// le message de renouvellment est envoye du contrôleur global aux membres du groupe
class TEK_timer : public TimerHandler {
 public:
  TEK_timer(GKMPANAgent *a) : TimerHandler() { a_ = a; }
 protected:
  virtual void expire(Event *e);
  GKMPANAgent *a_;
};

// Entête GKMPAN
struct hdr_GKMPAN {
        // methodes d'acces a l'entete
static int offset_; //utilise par le gestionnaire d'entetes
inline static int& offset() { return offset_; }
inline static hdr_GKMPAN* access(const Packet* p) {
return (hdr_GKMPAN*) p->access(offset_);}
};

// Classe de l'Agent GKMPAN
class GKMPANAgent : public Agent {
 public:
  GKMPANAgent();
  int command(int argc, const char*const* argv);
  void recv(Packet*, Handler*);
  void diff_tek (void);
 protected:
  TEK_timer TEK_timer_ ;
};

#endif
```

C.2 Scripts de simulations et d'analyse des résultats

C.2.1 Script de simulation TCL

```
# TCL script three parameters:  #scenario, #senders, #surface

set opt(stop) 1999.0
set nodes       [lindex $argv 2]
set mobility    1
set pausetime   0
set traffic     cbr
set premier     [lindex $argv 0]
```

```
set deuxieme    [lindex $argv 1]
set surface    [lindex $argv 3]

set ns_ [new Simulator]
set topo [new Topography]
$topo load_flatgrid $surface $surface

set tracefd [open trace/trace$premier$deuxieme w]
$ns_ trace-all $tracefd

set god_ [create-god $nodes]
$ns_ node-config -adhocRouting AODV \
                         -llType LL \
                         -macType Mac/802_11 \
                         -ifqLen 50 \
                         -ifqType Queue/DropTail/PriQueue \
                         -antType Antenna/OmniAntenna \
                         -propType Propagation/TwoRayGround \
                         -phyType Phy/WirelessPhy \
                         -channel [new Channel/WirelessChannel] \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace ON \
                         -macTrace OFF \
                         -movementTrace OFF

for {set i 0} {$i < $nodes} {incr i} {
    set node_($i) [$ns_ node]
    $node_($i) random-motion 0;
}

for {set i 0} {$i < $nodes} {incr i} {
        set p($i) [new Agent/BALADE]
        $ns_ attach-agent $node_($i) $p($i)
}

$ns_ at 800 "$p(0) FORM_CLUSTER"

puts "Loading scenarios file..."
source "scenarios/scen$premier$deuxieme"

$ns_ at $opt(stop) "$ns_ halt"

puts "Starting Simulation ..."
$ns_ run
```

C.2.2 Script d'analayse des traces

```
#!/usr/local/bin/perl
#$totalReceivers = 10;
#$totalNodes = 50;
$totalPid = 20000;

($traceFile,$totalNodes) = @ARGV;
open (MYFILE, $traceFile) || die "cannot open the trace file\n";

$i = 0;
while($i < $totalPid){
  $send[$i] = 0;
  $sendTime[$i] = 0;
  $j = 0;
  while($j<$totalNodes){
    $recv[$i][$j] = 0;
    $latency[$i][$j] = 0;
    $j++;
  }
  $i++;
}

while (<MYFILE>)
{
  $current_time = findTime($_);
  if (m/TEK/ && m/^s/ && m/AGT/) {
    $id = findPacketId($_);
    $send[$id] = 1;
    $sendTime[$id] = $current_time;
  }
  elsif(m/TEK/ && m/^r/ && m/RTR/){
    $id = findPacketId($_);
    $node = findNodeId($_);
    if ($recv[$id][$node] == 0){
      $recv[$id][$node] = 1;
      $latency[$id][$node] = $current_time - $sendTime[$id];
    }
  }
}

#print "almost done!\n";
close (MYFILE);

$totalSend = 0;
$totalRecv = 0;
```

```
$totalLatency = 0;
$i = 0;
while ($i < $totalPid){
  if ($send[$i] == 1){
    $totalSend++;
    $j = 1;
    while ($j < $totalNodes){
      if ($recv[$i][$j] == 1){
        $totalRecv++;
        $totalLatency += $latency[$i][$j];
      }
      $j++;
    }
  }
  $i++;
}

#print $totalNodes, "\t" ; # noeuds
#print $totalRecv,  "\t" ; # total recus

print "nb ";
print $totalNodes, " ";
print $totalRecv/(5*($totalNodes-1)), " " ; # PDR
print $totalLatency/$totalRecv, " ; \n"; # Latence

sub findPacketId {
@fields = split(/ /, $_[0]);
$pid = $fields[6];
if ($fields[4] != "") {$pid = $fileds[5];}
return $pid;
}

sub findTime {
@fields = split(/ /, $_[0]);
$theTime = $fields[1];
return $theTime;
}

sub findNodeId {
  $startPo = index($_[0], "_")+1;
  $endPo = index($_[0], "_", $startPo);
  $len = $endPo - $startPo;
  $node = substr($_[0], $startPo, $len);
  return $node;
}
```

C.2.3 Tableaux de résultats des simulations

BALADE

Surface	Nombre de membres	Latence (sec)	PDR (%)	Energie (messages transmis)
		Moyenne de 10 simulations		
500*500	30	0.0874663615	75.7241379310	71266.3
	40	0.09824451682	73.5384615384	107241.7
	50	0.0971491842	59.7235621521	146224.1
1000*1000	30	0.1317006353	56.3949843260	81807.6
	40	0.1487102553	65.0582750582	129529.7
	50	0.1519970875	47.9591836734	204006.7
1500*1500	30	0.0987081029	28.9341692789	57169.3
	40	0.1520040201	35.4965034965	89558
	50	0.1422341084	41.5343228200	153944.4

FIG. C.1 – Résultats de simulation de BALADE

DMGSA

Surface	Nombre de membres	Latence (sec)	PDR (%)	Energie (messages transmis)
		Moyenne de 10 simulations		
500*500	30	0.3322041670	39.1034482758	96571.4
	40	0.3256686155	50.6153846153	146131.6
	50	0.3319164007	49.5102040816	170682.7
1000*1000	30	0.3977047318	64.4514106532	111787.8
	40	0.4028266982	61.3053613056	181381.8
	50	0.3915519978	56.5491651272	224619.1
1500*1500	30	0.3784654502	29.7178683351	88591.4
	40	0.4140159153	35.1794871794	138743.1
	50	0.4502149463	46.321892344	222026.6

FIG. C.2 – Résultats de simulation de DMGSA

GKMPAN

Surface	Nombre de membres	Latence (sec)	PDR (%)	Energie (messages transmis)
		Moyenne de 10 simulations		
500*500	30	1.1268771655	93.3103448275	34137
	40	1.1026115626	92.1282051282	45771.5
	50	1.1402484995	97.2857142857	57540.65
1000*1000	30	1.9608002252	75.7931034482	33310.85
	40	2.0542066496	83.5897435897	44737.15
	50	2.3527524697	84.6122448979	56255.95
1500*1500	30	1.6208990694	31.5172413793	33220.55
	40	2.4654872192	34.0512820512	44036.7
	50	2.7345445475	39.4285714285	55415.75

FIG. C.3 – Résultats de simulation de GKMPAN

Glossaire

1 affecte n	Le renouvellement de la clé du groupe TEK affecte tous le membres du groupe
1 n'est pas égal à n	Le renouvellement de la clé du groupe n'est pas effectué pour tous les membres du groupe mais individuellement
ACL	(*Access Control List*) Liste de contrôle d'accès
Ad Hoc	Un réseau ad hoc est une collection de nœuds mobiles, dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans aucune infrastructure fixe ni administation centralisée
Authentification	Service de sécurité qui permet à un nœud de s'assurer de l'identité des nœuds avec lesquels il communique
AVISPA	(*Automated Validation of Internet Security Protocols and Application*) Outil automatique de vérification des protocoles de sécurité dans Internet
BALADE	Protocole de gestion de clés de groupe dans les réseaux ad hoc
CBID	(*Crypto Based Identifiers*) Identificateur cryptogaphique unique
CG	Contrôleur global
CKDS	*Combinatorial Key Distribution Scheme*
CL	Contrôleur local
CL-ATSE	(*Constraint Logic Based Attack Searcher*) Back-end utilisé par AVISPA pour valider des protocoles de sécurité, basé sur la méthode de résolution de contraintes

Confidentialité	Service de sécurité qui assure dans le contexte du multicast que seuls les membres du groupe peuvent accéder au flux de données envoyé par la source
Contrôle d'accès	Service de sécurité qui assure que seuls les membres autorisés à joindre le groupe peuvent y adhérer
CSG_i	Clé du sous groupe i
Déni de service	Attaque malicieuse visant à compromettre la disponibilité d'une ou plusieurs entités dans le réseau
DEP	*Dual Encryption Protocol*
DH	*Diffie Hellmann*
Disponibilité	Service de sécurité qui assure qu'un serveur ou une entité spécifique est toujours disponible dans le réseau, contre toute attaque de déni de service
DMGSA	*Distributed Multicast Group Security Architecture*
GCL	Groupe des contrôleurs locaux
GDH	*Group Diffie Hellmann*
GKMP	(*Group Key Management Protocol*) Protocole de gestion de clé de groupe
GKMPAN	*Group Key Management Protocol in Ad Hoc Networks*
HLPSL	(*High Level Specification Langage*) Langage formel de spécification des protocoles de sécurité
Intégrité	Service de sécurité qui assure que les données n'ont pas été modifiées au cours de leur acheminement dans le réseau
KEK	(*Key Encryption Key*) Clé de chiffrement de clé
KEK_CCL	Clé de chiffrement de clé partégé par tous les contrôleurs locaux
KEK_CSG_i	Clé de chiffrement de clé partégé par tous les membres du sous groupe i

LKH	*Logical Key Hierarchy*
LKHW	*A Directed Diffusion-Based Secure Multicast Scheme for Wireless Sensor Networks*
LML	Liste des membres participants
MPR	Relais Multipoint
Non-répudiation	Service de sécurité qui assure qu'une source ne peut pas nier avoir envoyé un message dans le réseau
NS2	(*Network Simulator*) Simulateur de réseaux en libre source
OFT	*One-way function Trees*
OMCT	(*Optimized Multicast Clustering Tree*) Algorithme de clusterisation dynamique dans BALADE
PKI	(*Public Key Infrastructure*) Infrastructure à clés publiques
RL	(*Revocation List*) Liste de révocation
Secret futur	Un membre ayant quitté le groupe multicast, ne doit plus être capable de déchiffrer le flux mutlticast après son départ (*Forward Secrecy*)
Secret passé	une entité qui adhère au groupe ne doit pas être capable d'accéder au flux de données émis avant son arrivée (*Backward Secrecy*)
TEK	(*Traffic Encryption Key*) Clé de chiffrement de données
Transmission Multicast	Communication d'une source vers plusieurs récepteurs (1 vers n), ou de plusieurs sources vers plusieurs récepteurs (n vers n), pour lesquelles l'acheminement des données suit un arbre multicast, et ne s'effectue pas en point à point entre une source et tout récepteur individuellement

Index

Résumé

La transmission multicast est un mécanisme efficace de communication pour des applications, telles que les vidéos conférences, les jeux interactifs et la distribution de logiciels orientés groupe. L'avantage principal des communications multicast est d'optimiser la consommation des ressources de réseaux, principalement en réduisant la consommation de la bande passante et des ressources des routeurs.

En parallèle avec le développement des services multicast dans l'Internet, la dernière décennie a vu l'émergence des réseaux ad hoc, grâce à l'apparition de nouvelles technologies sans fil et de nouvelles normes (par exemple 802.11, Wimax, ...). Un réseau ad hoc est une collection dynamique de dispositifs, autonomes, reliés sans aucune infrastructure fixe, qui peut être fortement mobile.

La combinaison d'un environnement ad hoc avec des services multicast à déployer, a induit de nouveaux défis envers l'infrastructure de sécurité requise pour un large déploiement des communications multicast dans un tel environnement. C'est dans ce cadre que nous avons défini deux protocoles de gestion de clé de groupe dans les MANETs, un protocole orienté récepteurs, appelé Enhanced BAAL et un protocole orienté sources, dénommé BALADE. Nous avons également proposé OMCT : un algorithme de clusterisation dynamique du groupe multicast, que nous avons intégré dans BALADE. Ce couplage assure une distribution de clés efficace et rapide, tenant compte de la mobilité et de la localisation des nœuds, tout en optimisant la consommation de l'énergie et de la bande passante. L'applicabilité de BALADE couplé avec OMCT est démontrée à travers sa validation formelle avec l'outil AVISPA, son implantation dans le cadre d'une application de Jukebox coopérative, et l'évaluation de ses performances avec l'outil de simulation de réseaux NS2.

Mots-clés: Sécurité, Multicast, Ad Hoc

Abstract

Multicast communication is an efficient mechanism, adapted to several applications such as video conferences, software distributions, ... The principal advantage of this type of communications is the optimisation of network resources, mainly by reducing the bandwidth and the routers resources consumption.

In parallel to the development of the multicast services within Internet, the last years saw the emergence of the ad hoc networks, due to the appearance of new wireless technologies and new standards (like 802.11, Wimax, ...). An ad hoc network is a dynamic collection of autonomous entities, without any fixed infrastructure, and which can be highly mobile.

The combination of MANETs with the multicast services, induces new challenges towards the required security infrastructure, to ensure a reliable deployment of the multicast communications within this wireless environment. In this context we defined two group key management protocols in MANETs, a receivers-oriented protocol called Enhanced BAAL and a sources-oriented protocol named BALADE. We proposed also OMCT : a dynamic clustering algorithm of multicast group that we integrated in BALADE. This coupling ensures an efficient and fast keys distribution, taking into account the mobility and the localization of nodes, while optimizing the energy and bandwidth consumptions. The applicability of BALADE coupled with OMCT is shown through its formal validation with the AVISPA tool, its implementation within a cooperative jukebox application, and the evaluation of its performances under the network simulator NS2.

Keywords: Security, Multicast, Ad Hoc