UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN GÉNIE ÉLECTRIQUE ET GÉNIE INFORMATIQUE

PAR
SAMIR KHALI

DÉVELOPPEMENT D'UNE TECHNIQUE DE DISTRIBUTION DE CLÉS DE CRYPTAGE POUR LES APPLICATIONS MULTICAST SUR LES RÉSEAUX SANS FIL AD HOC

AOÛT 2008

RÉSUMÉ

Les réseaux sans fil ad hoc ont connu une phase de progression très rapide due à leur facilité et leur rapidité de mise en œuvre. Ces réseaux sont assez perméables et ouverts aux intrusions. De plus, ils sont caractérisés par une topologie dynamique, une bande passante limitée, des contraintes énergétiques, une sécurité physique limitée, l'absence d'infrastructure et un rayon de portée limitée. Le multicast est la communication d'un seul message d'un émetteur vers plusieurs récepteurs. Il permet, entre autres d'assurer, l'économie des ressources et de faciliter la communication en groupe ou d'offrir des services de groupe. On observe ainsi depuis quelques années une prolifération d'applications permettant des communications entre groupes d'individus partageant souvent des intérêts communs : on peut citer l'enseignement en ligne, les jeux en ligne, la téléconférence, les services de vidéo en ligne, etc. La plupart des services de groupe cité ci-dessus s'offrent de plus en plus sur des périphériques sans fil. Mais, il se pose d'importants problèmes de sécurité d'abord parce que l'interface de communication radio est facilement accessible, donc très vulnérable aux perturbations et aux attaques extrêmement complexes. Le projet de mémoire consiste à étudier le côté sécurité de ces réseaux avec l'étude des nouvelles approches de distribution des clés basée sur la technique de cryptage à courbes elliptiques. Le travail consiste à développer une architecture de distribution de clés qui, tout en tenant compte des exigences inhérentes à la structure et aux propriétés des réseaux mobiles ad hoc, assure un meilleur recouvrement lors de la perte de connectivité d'un membre du groupe.

REMERCIEMENTS

Je remercie, vivement, pour leur sérieux, mes professeurs Ahmed Chériti et Boucif Amar Bensaber qui ont accepté de travailler avec moi sur ce sujet. J'exprime ma grande gratitude pour leur encadrement de haute qualité et leurs conseils bénéfiques. Je présente surtout ma sincère reconnaissance à M. Boucif Amar Bensaber pour sa disponibilité et ses réponses à toutes mes questions.

Qu'il me soit permis de remercier tous les évaluateurs de ce mémoire, les professeurs M. Adel Omar Dahmane, M. Ahmed Chériti et M. Ismaïl Biskri qui ont accepté de prendre de leur temps pour lire, évaluer et suggérer les corrections nécessaires de ce mémoire.

Je déclare mes très grandes gratitudes à mon père et à ma mère qui m'ont encouragé pendant ce travail et pendant tout mon parcours scolaire. Qu'ils trouvent ici mon fidèle remerciement pour tous les sacrifices qu'ils ont fait pour moi et je leur dis, je vous aime tellement.

Je remercie tous les membres de ma famille (petits et grands) ainsi que tous mes amis pour leur soutien moral et encouragement qui m'ont été d'une importance capitale dans l'élaboration de ce travail.

Table des matières

RÉSUMÉ.....ii
REMERCIEMENTS.....iii
Table des matières.....iv
Liste des tableaux.....x
Liste des figures.....xi

CHAPITRE 1- INTRODUCTION.....1

CHAPITRE 2- Le MULTICAST.....3
2.1 Introduction.....3
2.2 Définition.....4
2.3 La création du groupe Multicast.....5
2.4 Description des entités.....6
2.5 L'utilisation des communications Multicast.....7
2.5.1 Quelques motivations de l'utilisation du Multicast.....7
2.5.2 Les applications du Multicast.....8
2.6 IP Multicast.....9
2.6.1 Introduction.....9
2.6.2 Définition.....10
2.6.3 IP Unicast vs IP Multicast.....10

2.6.4 IGMP (Internet Group Management protocol).....11

2.6.5 MBONE.....13

2.7 Conclusion.....14

CHAPITRE 3- Les réseaux mobiles AD HOC.....16

3.1 Généralité.....16

3.2 Les des réseaux ad hoc.....17

3.3 Modèles de communication des nœuds dans les réseaux ad hoc.....18

3.4 Caractéristiques des réseaux ad hoc.....19

3.5 Les réseaux ad hoc grand publique.....20

3.6 Les Problèmes et contraintes spécifiques aux réseaux ad hoc.....21

3.7 Les réseaux ad hoc multicast 22

3.7.1 L'impacte de la mobilité sur le routage au sein des réseaux ad hoc.....23

3.8 Conclusion.....26

CHAPITRE 4- Approches de gestion de clés pour les communications multicast les réseaux Ad hoc.....27

4.1 Introduction.....27

4.2 Les services de sécurité et la gestion des clés.....27

4.3 Approche de gestion de clés de groupe.....30

4.3.1 Approches contributives.....31

4.3.1.1 Diffusion.....31

4.3.1.1.1 Secure Lock.....31

4.3.1.1.2 Secure Multicast Protocol.....32

4.3.1.2 Approches en arbre de clés centralisé.....33

4.3.1.2.1 Approche de Wallner.....34

4.3.1.2.2 Autres approches.....35

4.3.1.2.3 Inconvénients.....35

4.3.2 Approches semi-distribuées.....36

4.3.2.1 Approche SMKD.....36

4.3.2.2 Iolus.....36

4.3.2.2.1 Les ajouts et les départs.....37

4.3.3 Approches distributives.....38

4.3.3.1 Approches basées sur l'algorithme de Diffie-Hillman.....39

4.3.3.1.1 L'algorithme de Diffie-Hillman.....39

4.3.3.1.2 L'approche de Fiat et Naor.....41

4.3.3.1.3 L'approche de Ingemarson.....42

4.3.3.1.4 Les approches de la suite CLIQUES.....43

4.3.3.1.4.1 L'approche GDH.1.....43

4.3.3.1.4.2 L'approche GDH.2.....45

4.3.3.1.4.3 L'approche GDH.3.....46

4.3.3.2 Approches en arbre.....48

4.3.3.2.1 L'approche STR.....48

4.3.3.2.2 L'approche TGDH.....52

4.4 Conclusion.....55

CHAPITRE 5- L'approche GAKAP: Group Activity BAsed Key Agreement Protocol.....57

5.1 Introduction.....57

5.2 Les courbes elliptique au sein de la cryptographie.....58

5.2.1 Définitions.....59

5.2.1.1 Courbe elliptique.....59

5.2.1.2 Corps finis.....59

5.2.2 Description mathématique de la cryptographie à courbes elliptiques 59

5.2.2.1 Les opérations des courbes elliptiques.....60

5.2.2.2 Les courbes elliptiques sur les corps $GF(2^m)$.....61

5.2.2.3 L'ordre de la courbe.....62

5.2.3 L'algorithme des courbes elliptiques.....63

5.2.4 ECDH: Diffie-Hillman à base des courbes elliptiques.....64

5.2.5 La signature digitale DSA.....67

5.2.5.1 Génération des clés.....67

5.2.5.2 La signature du message.....68

5.2.5.3 La vérification du message signé.....68

5.2.6 La signature digitale ECDSA.....68

5.2.6.1 La préparation et la génération des clés.....69

5.2.6.2 La signature du message.....69

5.2.6.3 La vérification du message signé.....69

5.3 Choix de la courbe elliptique.....70

5.4 Définitions des éléments.....70

5.4.1 Notations.....72

5.4.2 Types de paquets.....72

5.4.2.1 Description de chaque paquet.....72

5.4.2.2 Clé glissée.....79

5.4.2.3 Clé cachée.....80

5.4.2.4 Fonction à sens unique.....80

5.4.2.5 Le stockage.....80

5.4.2.6 La clé Mise à jour.....80

5.4.2.7 La constante d'activité α.....80

5.5 Les opérations sur le groupe.....81

5.5.1 Élection du sponsor.....81

5.5.2 Initialisation.....82

5.5.3 Les événements au sein du groupe.....83

5.5.3.1 L'ajout simple.....83

5.5.3.2 Départ simple.....85

5.5.3.3 Ajouts multiples.....86

5.5.3.4 Départs multiples.....87

5.5.3.5 Fusion des nœuds.....89

5.6 Conclusion.....89

CHAPITRE 6- Les étapes de simulation de l'approche GAKAP.....91

6.1 Introduction.....91

6.2 Les outils utilisés.....91

6.2.1 Les librairies Openssl.....91

6.2.2 Les simulateurs réseau.....92

6.2.2.1 NS-2 (*Network Simulator*).....92

6.2.2.2 NS-2 dans l'application 93

6.3 Les simulations.....93

6.3.1 Scénario et métriques.....93

6.3.1.1 Scénario.....93

6.3.1.2 Métriques.....96

6.3.2 Les étapes d'une simulation réseau.....96

6.3.3 Implémentation du programme de simulation.....96

6.3.3.1 L'implémentation et les problèmes rencontrés.....96

6.3.3.2 Visualisation des simulations.....97

6.3.3.3 Événements au niveau de l'arbre.....98

6.3.3.3.1 L'élection du sponsor.....98

6.3.3.3.2 L'initialisation.....101

6.3.3.4 Événements au niveau des nœuds.....104

6.3.3.4.1 L'ajout.....105

6.3.3.4.2 Le départ.....106

6.4 Conclusion.....108

Chapitre 7 - Conclusion.....110

Bibliographie.....113

Liste des tableaux

Tableau 1 Exemples d'applications multicast.....9

Tableau 2 Résumé des symboles utilisés.....24

Tableau 3 Sommaire des paramètres Diffie-Hillman.....40

Tableau 4 Notations.....72

Liste des figures

Figure 1	Communication 1-à-N	4
Figure 2	Communication N-à-M	5
Figure 3	Communication N-à-1	5
Figure 4	Transfert Unicast	10
Figure 5	Transfert Multicast	11
Figure 6	L'emplacement des routeurs MBone	14
Figure 7	Décomposition des réseaux sans fil	16
Figure 8	Communication directe entre deux nœuds du même réseau	18
Figure 9	Communication par routage ou en mode ad hoc	19
Figure 10	Position de la gestion des clés dans le système de sécurisation	29
Figure 11	Approches de gestion des clés de groupe	30
Figure 12	Architecture générale de l'arbre binaire des clés	34
Figure 13	Protocole de Wallner	35
Figure 14	Architecture de l'approche Iolus	37
Figure 15	L'anneau logique CKDS	42
Figure 16	L'approche GDH.1	44
Figure 17	L'approche GDH.2	46
Figure 18	L'approche GDH.3	47
Figure 19	L'arbre STR	49
Figure 20	Mise à jour de l'arbre après l'ajout d'un nouveau nœud (M5)	50
Figure 21	Mise à jour de l'arbre après le départ d'un nœud (M5)	51

Figure 22 Création de l'arbre de clé avec TGDH....52

Figure 23 L'arbre de clés après l'ajout....53

Figure 24 L'arbre de clés après le départ....54

Figure 25 Addition des deux points P et Q....61

Figure 26 Doublage des deux points P et Q....62

Figure 27 Ordre du choix du sponsor....81

Figure 28 Phase d'initialisation de l'arbre des clés GAKAP....83

Figure 29 Événement d'ajout simple dans l'approche GAKAP....84

Figure 30 Événement du départ simple dans l'approche GAKAP....85

Figure 31 Événement des ajouts multiples dans l'approche GAKAP....87

Figure 32 Événement des départs multiples dans l'approche GAKAP....88

Figure 33 NAM Outil de visualisation de NS-2....97

Figure 34 Explication du NAM....98

Figure 35 Ajout d'un nœud....106

Figure 36 Départ d'un nœud....108

Chapitre 1 - INTRODUCTION

Tous au long de l'existence de l'homme, ce dernier a inventé des méthodes et des techniques pour communiquer les informations, surtout les alertes du danger. Passant par les cris, les tam-tams, la fumée chez les Indiens, la téléphonie et arrivant aux réseaux sans fil qui sont en développement phénoménal.

Avec la rapidité du développement de la technologie sans fil, de nouvelles exigences ont vu le jour, comme le besoin d'être connecté en permanence avec les autres et la communication de grands flux d'information d'une manière instantanée et efficace. Les réseaux sans fil s'avèrent une solution convenable à ces besoins. Mais, vu l'ouverture de ce genre de réseaux sur des environnements non sûrs, leur sécurisation est un côté primordial dans leur bon fonctionnement.

Dans le ce mémoire, il sera donc question de traiter le côté sécurité des échanges de groupe au sein d'un nouveau type des réseaux sans fil qui est les réseaux ad hoc.

Pour assurer la sécurisation des applications et des communications au sein des réseaux sans fil, plusieurs approches de distribution et de gestion des clés de cryptage ont vue le jour jusqu'à date, ces dernières sont regroupées en trois sous-groupes : contributives, semi-distribuées et distributives. Malgré la robustesse de ces approches, elles représentent des inconvénients majeurs qui se présentent dans l'exploitation des ressources du réseau et des machines. D'un côté par la lourdeur des calculs effectués lors de l'exécution des algorithmes et d'un autre côté par la grande consommation de la bande passante qui est

limitée dans le cas des réseaux sans fil. Pour palier à ces incconvénients, ce travail de recherche introduit une approche de distribution et de gestion des clés de cryptage connu sous le GAKAP et qui est basée sur les courbes elliptiques. La description et la simulation est la donc la principale contribution de ce mémoire.

Le mémoire sera divisé de la façon suivante :

Dans le deuxième chapitre, nous allons traiter des communications de groupe ou multicast, leurs principes, les étapes de création du groupe et les quelques motivations d'utilisation de ce type de communications.

Le troisième chapitre sera consacré aux réseaux sans fil et précisément les réseaux ad hoc, leur définition, leur principe et leur structure ainsi que quelques exemples de leurs applications.

Les approches et les techniques existantes de distribution des clés de cryptages seront évoquées dans le quatrième chapitre, avec une étude comparative de tous les types.

Le cinquième chapitre sera consacré à notre technique GAKAP, nous allons voir dans la première partie, les paquets utilisés lors des échanges entre les participants dans la session multicast du groupe, dans la deuxième partie, nous allons voir en détail les courbes elliptiques et tout ce qui concerne la cryptographie à base d'eux.

Au niveau de l'avant-dernier chapitre, nous allons discuter la méthodologie suivie pour amener les simulations, ainsi que les résultats obtenus. Pour arriver à la fin à faire une conclusion générale du mémoire.

Chapitre 2 - Le MULTICAST

2.1 Introduction

Le monde a vécu, ces dernières années l'apparition d'une grande variété d'applications qui ont conduit à un nouveau besoin d'être connecté en tout temps à un réseau, quel que soit l'endroit où l'on se trouve. De nouveaux réseaux sans fil sont apparus: réseaux de téléphonie, de diffusion haut débit, locale ou grande distance par l'intermédiaire de communications sans fil, parmi eux, il y a les réseaux ad hoc qui ont vu le jour avec l'apparence des dispositifs informatiques mobiles comme les ordinateurs portables (Lap Top), le développement de ces réseaux sans fil a nécessité des avances technologiques au niveau matériel et logiciel et des techniques de codage et de confidentialité. Ainsi avec le besoin de communiquer les informations à plusieurs utilisateurs du même réseau (les vidéos-conférences, la diffusion de la télévision par Internet), on a eu la naissance du mode Multicast qui permet l'envoi des informations en même temps à plusieurs unités connectées au même réseau.

Communication Multicast : C'est le terme employé pour décrire une communication où l'information est envoyée en même temps d'un ordinateur à un groupe d'ordinateurs identifiés comme entité virtuelle par une même adresse ou un même nom.

Dans le cadre de ce mémoire, nous avons adapté certaines structures de données et algorithmes développés au sein du laboratoire LAMIA (Université du Québec à Trois-Rivières UQTR) par l'équipe de recherche sur la sécurité des communications Multicast.

2-2 Définition

Le Multicast est un mécanisme qui permet à une (ou plusieurs) source (s) de diffuser de l'information à un groupe prédéfini de récepteurs de façon optimale. Au lieu de générer une copie des données pour chaque membre du groupe, la source n'envoie qu'une seule copie et les unités intermédiaires du réseau (routeurs, par exemple) se chargent de créer des duplicata des paquets et les transmettent au reste du groupe quand cela est nécessaire. L'identification du groupe se fait par une adresse IP unique appelée *adresse IP Multicast*. Les applications Multicast sont regroupées en trois catégories:

- **Les applications 1-à-N** (figure 1): elles sont définies par la présence d'une seule entité source qui émet une masse de données à plusieurs récepteurs.

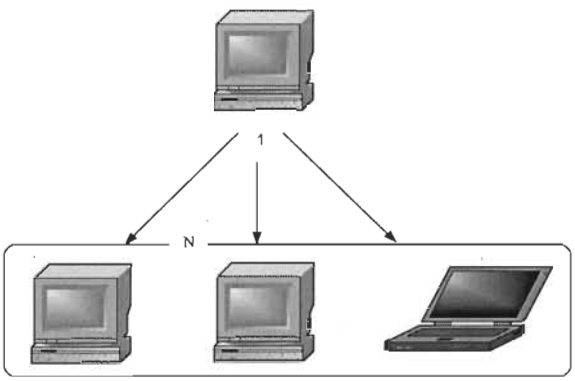


Figure 1 : Communication 1-à-N

- **Les applications N-à-M** (figure 2): pour les applications de cette catégorie, tous les récepteurs peuvent avoir le rôle d'émetteur.

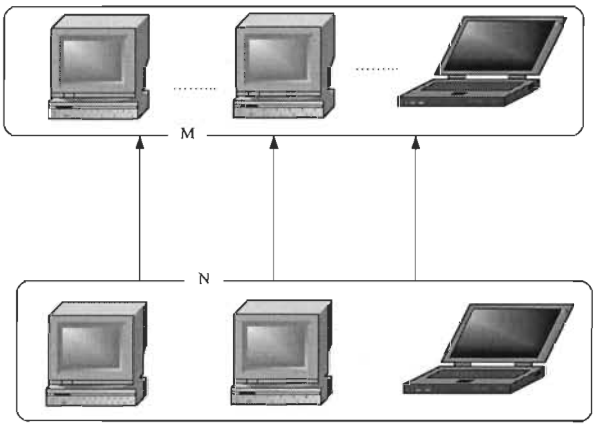


Figure 2 : Communication N-à-M

- **Les applications N-à-1** (figure 3): il s'agit des applications où plusieurs sources envoient des données à un seul récepteur.

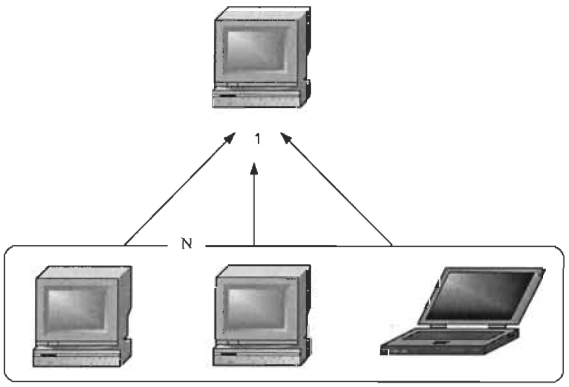


Figure 3 : Communication N-à-1

2.3 Création du groupe Multicast.

Pour créer un groupe Multicast, il faut passer par cinq principales étapes :

- Initialiser le groupe de participants : pour ce faire, il faut définir une entité qu'on appelle l'**initiateur** du groupe. C'est le responsable principal qui définit les paramètres de sécurité et qui crée la liste de contrôle d'accès du groupe.

- Choisir un contrôleur de groupe : l'initiateur choisit une machine comme **Contrôleur** de groupe qui s'occupera de la gestion des événements d'ajout et de départ des participants. Ce contrôleur a le privilège d'accepter ou de rejeter la demande d'un participant de s'ajouter au groupe. Ainsi, il peut supprimer des participants, étant donné qu'il a accès à la base de données de ces derniers.
- Sélectionner les participants : les autres éléments dans le groupe, mis à part l'initiateur et le contrôleur sont appelés les participants du groupe. Ils jouent le rôle d'émetteurs et/ou de récepteurs.
- Distribuer la liste des participants : suite à la vérification des participants par le contrôleur, la liste est transmise par ce dernier.
- Créer la clé du groupe : la clé du groupe sert à décrypter les messages envoyés par l'utilisateur en mode multicast. Cette clé est créée par le contrôleur et envoyée à tous les participants de la session en cours.

2.4 Description des entités

Une session Multicast est toujours débutée par l'initiateur qui détermine les paramètres nécessaires à la diffusion multicast. On retrouve parmi ces paramètres:

- les contraintes (attributs) de la sécurité;
- le nombre de participants possible ainsi que leurs caractéristiques;
- les particularités des connexions (la bande passante…);
- la qualité de service (le débit de transmission…);

- Émetteur (l'initiateur dans la majorité des cas) : tout participant à la session autorisé à envoyer les paquets Multicast vers les autres machines de la même session.

- Récepteur : tout participant à la session à qui l'initiateur a donné l'autorisation de recevoir les paquets transmis au sein du groupe (en général tous les membres du groupe).
- Clé : une clé est un code attribué par l'initiateur pour déchiffrer les paquets échangés au sein du même groupe. Il existe différents types de clés, elles peuvent être publiques comme elles peuvent être privées.
- Message : l'information échangée entre les membres du groupe [1].
- Diffusion : envoie du même message par un membre à tous les membres du groupe simultanément [1].

2.5 L'utilisation des communications Multicast

2.5.1 Quelques motivations de l'utilisation du Multicast

Plusieurs raisons peuvent motiver l'utilisation des techniques de communication de groupe, et en particulier du service de routage Multicast.

La réduction des coûts de transmission est la première motivation de l'utilisation du multicast. Avec cette technique, il est possible de diffuser de l'information à un très grand nombre de participants sans avoir besoin de beaucoup de bande passante.

La deuxième raison est *le passage à l'échelle, en fonction du nombre de participants.* Chaque information n'est diffusée qu'une seule fois et simultanément à tous les récepteurs. L'appartenance de plusieurs récepteurs à une même session ne doit pas causer l'épuisement des ressources du réseau.

2.5.2 Les applications du Multicast

Il y a une multitude d'applications du multicast, par exemple [2]:

- Le streaming audio/vidéo : Pour transmettre une grande quantité de données multimédia à un nombre important d'utilisateurs, le multicast est adopté comme technique de transmission efficace.
- La diffusion de contenu "à la demande" : Les serveurs diffusent des données populaires (vidéoclip, logiciels, mise à jour…) et peuvent utiliser le Multicast afin de transmettre d'une manière continue les données que les utilisateurs pourront récupérer à la demande.
- La diffusion de contenu en mode ``push'' : Il s'agit d'une application de diffusion où c'est la source qui prend l'initiative et non les récepteurs. Le contenu est constitué d'un ou plusieurs fichiers.

Il existe d'autres utilisations. Citons à titre d'exemple, la diffusion de données issues des marchés financiers, les outils de visioconférence, de travail coopératif, les jeux multi-utilisateurs, etc... [2].

Ainsi, on peut classer ces applications comme suit:

Transmission des fichiers multimédias	
Temps réel	**Non-temps réel**
Transmission vidéo Vidéoconférence Diffusion audio par Internet Événements multimédias	Réplication Livraison des données sur demande
Transmission des données	
Temps réel	**Non-temps réel**
Côtes de la bourse Nouvelles Jeux en ligne Cache web	Acheminement des données Réplication de la base de données

Tableau 1: Exemples d'applications multicast

2.6 IP Multicast [3]

2.6.1 Introduction

Le Multicast IP, c'est-à-dire la multidiffusion IP, est une technique de haute qualité inventée par le groupe de travail IETF (**I**nternet **E**ngineering **T**ask **F**orce) et utilisée pour diffuser en continu des données avec une haute qualité que du service. Le multicast, vu son efficacité, a pris la place de l'unicast vu que ce dernier ne répond pas aux demandes croissantes du streaming.

2.6.2 Définition

La multidiffusion IP est la transmission d'un datagramme IP à « un groupe d'accueil », c'est-à-dire un ensemble de plusieurs hôtes identifiés par une adresse IP de destination unique. Le datagramme multicast est livré à tous les membres du groupe destination avec la même qualité [3].

2.6.3 IP Unicast vs IP Multicast

Dans le mode Unicast (figure 4), le transfert d'un paquet à un groupe se fait séparément pour chacun des récepteurs. Ce qui cause l'encombrement des connexions réseau. Par contre, le mode multicast (figure 5) permet aux sources de diffuser les mêmes données à plusieurs utilisateurs à la fois. Cette technique offre des avantages considérables. Le réseau est beaucoup moins encombré et permet d´importantes économies puisque les investissements matériels requis sont moindres.

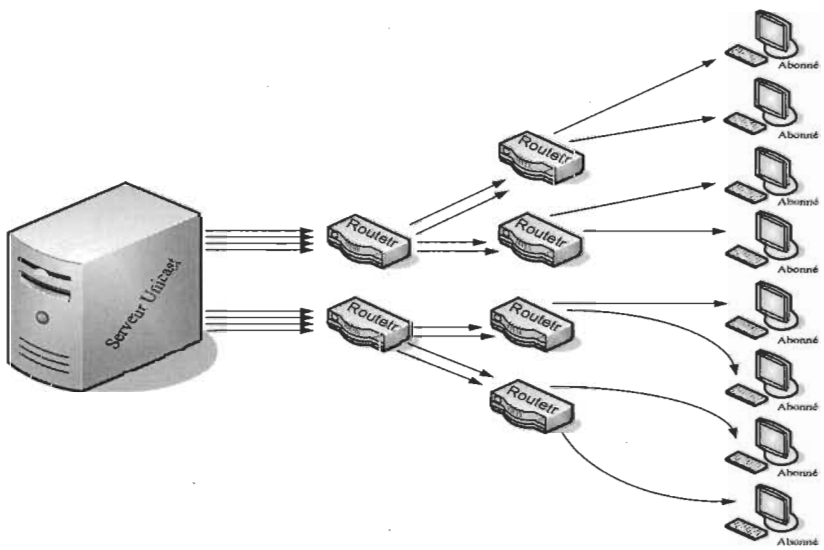


Figure 4 : Transfert Unicast

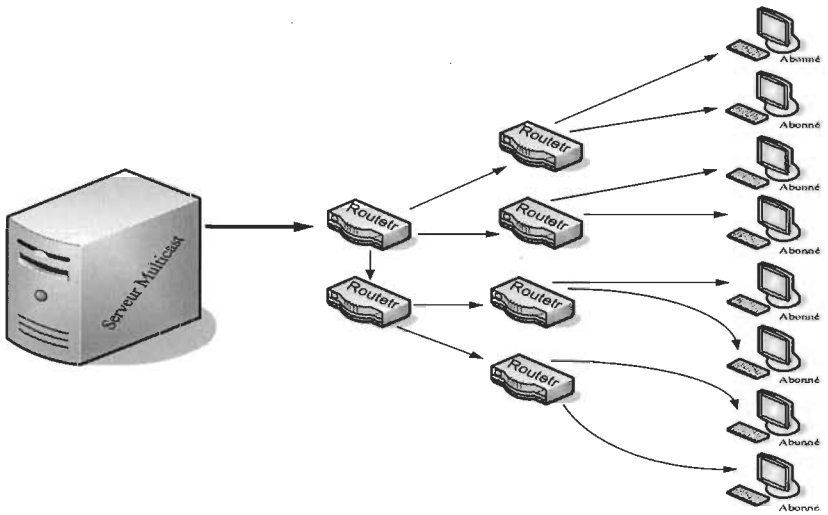


Figure 5 : Transfert Multicast

2.6.4 IGMP (*Internet Group Management protocol*)

IGMP [4] est un protocole qui appartient à la famille des protocoles de gestion de groupe sur Internet. Il gère les demandes des machines qui veulent quitter ou joindre un groupe multicast. Il permet des communications entre les hôtes d'un sous réseau multicast relié à un routeur multicast. Les demandes sont soit spontanées soit après requête du routeur. Pour cela, l'hôte diffuse un datagramme IGMP destiné aux routeurs multicast locaux. Il existe trois versions du protocole IGMP : IGMPv1, IGMPv2, IGMPv3 [5] :

IGMPv1 [5]: La première version de IGMP est évidemment la plus simple, et celle qui comporte le plus d'inconvénients. Par contre, IGMPv1 est considérée comme la base de l'élaboration de IGMPv2 et IGMPv3.

Dans IGMPv1, les hôtes qui désirent se joindre à un groupe multicast échangent avec les routeurs multicast deux types de messages :

- *IGMP membership report* : Message envoyé par les hôtes désireux joindre le groupe mutlicast et pris en compte par le routeur de sortie DR (*Designated Router*).

-*IGMP membership query* : régulièrement, le DR interroge les machines qui sont encore connectées.

La sécurité est l'inconvénient principal de cette méthode. Les hôtes ne sont pas capables de faire le filtrage de la source d'un trafic donné.

IGMPv2 [5]: Cette version a le même concept que IGMPv1 sauf que de deux types de messages ont été ajoutés:

- *Group-Specific Query* : Il permet de fixer précisément le groupe duquel on désire recevoir des données.

Un champ supplémentaire appelé "*Query Interval Response Time* " [4] est introduit dans les messages de type *Query*. Ce champ n'est intégré que dans les requêtes d'adhésion, il précise la durée maximale autorisée avant d'envoyer un rapport du répondant. Dans tous les autres messages, il est mis à zéro par l'émetteur et ignoré par les récepteurs.

- *Leave Group Message* : permet à un hôte de signaler au routeur concerné son départ du groupe pour qu'il ne lui envoie pas de messages. En contrepartie, le DR envoie le message "***Group-Specific Query***" pour vérifier s'il existe encore des hôtes désireux recevoir le trafic destiné au groupe. Dans le cas où aucun hôte ne répond, le DR déduit qu'il n'y a plus d'hôte faisant partie du groupe considéré.

IGMPv3 [6]: représente une révision principale des deux versions précédentes du protocole. Il permet à des hôtes d'indiquer la liste des routeurs desquels ils veulent recevoir le trafic. Le trafic des autres routeurs non désignés sera bloqué à l'intérieur du réseau. Il permet également à des hôtes de bloquer à l'intérieur du réseau les paquets parvenant des sources qui ont envoyé le trafic non désiré. C'est ce que l'on appelle le filtrage. Les améliorations apportées par IGMPv3 concernent la sécurisation.

Il existe de modes pour effectuer ce filtrage:

- *Mode Inclusion* : les paquets Multicast d'un groupe ne sont acceptés que si les sources font partie d'un ensemble prédéfini et identifié par le ***DR***.
- *Mode Exclusion* : tous les paquets multicast d'un groupe sont acceptés sauf ceux provenant de sources dont la liste est transmise par l'hôte au ***DR***

Les deux modes se font par le biais d'un message "*Version 3 membership report*" envoyé par le ***DR*** aux hôtes.

Vu ses performances, aujourd'hui IGMPv3 est adopté par de nombreuses entreprises telles Cisco, RealNetworks, Microsoft, et Nortel.

2.6.5 MBONE

MBone (**M**ulticasting **B**ack**Bone**) [7] [8], est une conséquence des deux premières expériences menées par Audiocast et IETF (**I**nternet **E**ngineering **T**ask **F**orce) en 1992 dans laquelle les phases audio et vidéo de leurs réunions sont diffusées en mode multicast vers les participants autour du monde. L'idée était de construire un banc d'essai d'IP multicast semi-permanent pour porter les transmissions de l'IETF et soutenir la liaison continue entre les réunions.

Le MBONE est un réseau virtuel. Il est posé sur des parties physiques d'Internet pour soutenir le cheminement des paquets multicast avant même que la fonction ne soit intégrée dans beaucoup de routeurs qui sont placés presque partout dans le monde. Le réseau se compose d'hôtes qui peuvent directement soutenir le multicast, reliés entre eux par des liens virtuels appelés "tunnel". Chaque hôte interconnecte un ensemble de sous-réseaux. Les paquets multicast circulant dans le réseau sont encapsulés dans des paquets IP (IP dans IP) qui incluent un champ IP positionné à 4. L'encapsulation des paquets sert à les rendre inaccessible que par les routeurs destination. Ces routeurs sont présents dans toutes les régions de la planète (figure 6).

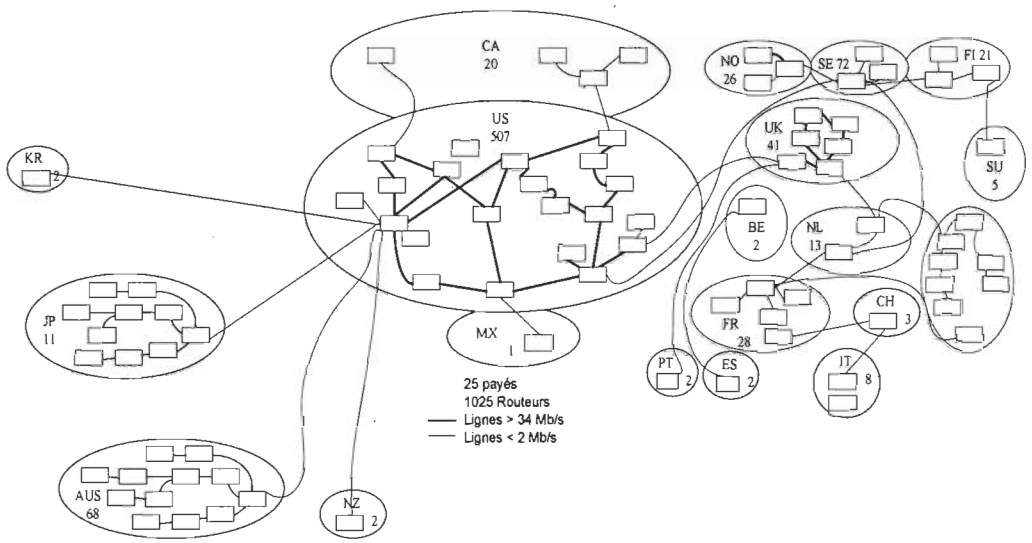


Figure 6 : L'emplacement des Routeurs MBone

2.7 Conclusion

Dans ce chapitre, nous avons exposé une présentation des principes fondamentaux du Multicast même s'il existe des particularités pour chaque type de réseaux surtout dans les réseaux filaires qui sont les plus utilisés à ce jour.

Dans le troisième chapitre, nous ferons un tour d'horizon des réseaux sans fil a*d hoc* ainsi que les problèmes liés à ce type de réseaux. Dans le quatrième chapitre, nous verrons les différentes approches classiques de distribution de clé de cryptage suivies lors des différents événements d'un groupe multicast. Par la suite, nous introduirons à travers un survol les techniques de cryptage utilisant les courbes elliptiques. Enfin, dans les chapitres 5 et 6, nous détaillerons notre méthode qui est basée sur la technique de cryptographie à courbes elliptiques et décrirons la méthodologie choisie et les résultats obtenus.

Chapitre 3 - Les réseaux mobiles AD HOC

3.1 Généralité

Les réseaux sans fil ont connu un grand succès surtout après l'apparition des produits à grand public basés sur la norme IEEE 802.11. Les débits importants obtenus de nos jours avec les réseaux sans fil ont permis l'exécution des applications complexes exigeant une bonne bande passante de communication. Il existe deux types de réseaux mobiles sans fil (figure 7): *les réseaux avec infrastructure* qui se basent sur le concept de la communication cellulaire où un point d'accès fixe est nécessaire, et les réseaux *ad hoc* qui ne nécessitent *aucune infrastructure*.

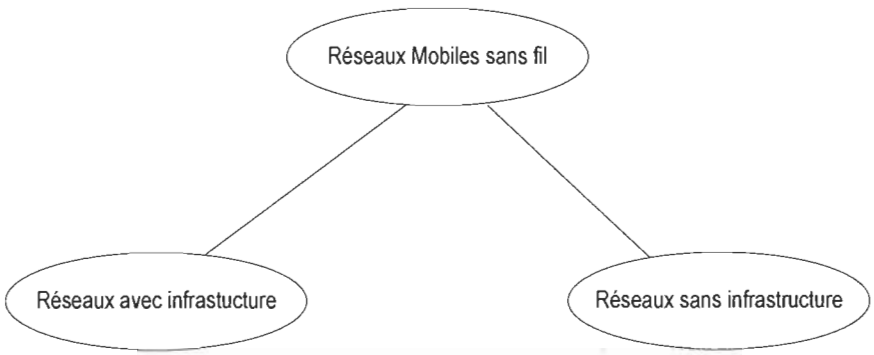


Figure 7 : Décomposition des réseaux sans fil

Les réseaux mobiles offrent une grande flexibilité d'utilisation. Ils permettent la connexion des zones dont le câblage est inabordable.

Les réseaux avec infrastructure (réseaux GSM par exemple) utilisent une importante infrastructure soit matérielle soit logistique. Par contre, les réseaux sans infrastructure (ad

hoc par exemple) se présentent sous forme d'un réseau temporaire composé d'entités mobiles interconnectées entre elles par le biais d'un faisceau hertzien sans avoir une administration centrale un tout point fixe d'accès (routeur par exemple).

Les réseaux ad hoc auxquels nous nous sommes intéressés sont ceux décrits et étudiés par le groupe de travail Mobile ad hoc NETworks *(MANET)* de l'***Internet Engineering Task Force*** (IETF). Il s'agit de réseaux sans fil mobiles et sans infrastructure fixe, utilisant généralement le médium radio, et où chaque nœud peut combiner les rôles de client et de routeur.

3.2 Les réseaux ad hoc

Un réseau ***ad hoc*** ou aussi ***MANET*** pour ***Mobile Ad hoc Network*** est un groupe de machines équipées d'une interface de communication sans fil. Puisque les nœuds sont mobiles et le réseau construit ne bénéficie d'aucune infrastructure préexistante ou d'appoint (station de base, borne, relais...), la topologie du réseau peut changer rapidement et de manière imprévisible. Les réseaux ad hoc sont décentralisés, toutes les activités du réseau comme la gestion et la diffusion des messages systèmes sont exécutées par les nœuds eux-mêmes. Ainsi, dans un réseau ad hoc, les nœuds peuvent joindre et quitter le réseau de façon totalement dynamique sans devoir en aviser les autres nœuds.

Un réseau ad hoc est un réseau point à point sans fil qui permet la communication entre deux entités, qui sont à la portée radio l'un de l'autre.

Comme déjà mentionné ci-dessus, si les faisceaux hertziens ne permettent pas d'établir un lien direct entre deux nœuds au sein du réseau ad hoc (l'émetteur et le récepteur sont

trop loin l'un de l'autre). La mise en œuvre d'un routage multisauts est nécessaire afin d'acheminer les paquets de données jusqu'à leur destination finale.

La mise en œuvre des algorithmes de routage, le calcul des routes et la sécurité sont les grands défis des réseaux ad hoc vu la dynamique de l'environnement et le manque d'entité centralisée. Ce qui nécessite l'implémentation d'algorithmes de cryptage et distribués de routage assez robustes, mais moins gourmands en ressources (Lightweight).

3.3 Modèles de communication des nœuds dans les réseaux ad hoc [9]

La liaison entre les mobiles (nœuds) se fait de deux manières, directes ou par routage:

- On dit que la communication est directe, si les nœuds peuvent s'échanger les données et cela ne peut se produire que si les stations sont suffisamment proches l'une de l'autre (figure 8).

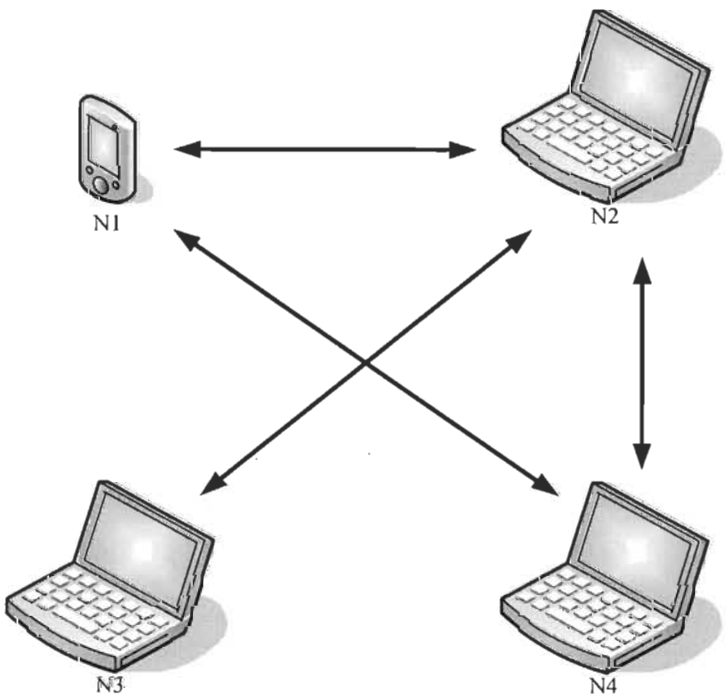


Figure 8 : Connexion directe entre deux nœuds du même réseau

Pour que la communication passe entre deux nœuds trop éloignés l'un de l'autre, il est nécessaire de faire appel à des nœuds dits intermédiaires qui se chargeront de l'acheminement des données. Pour cela, il est primordial d'avoir des nœuds qui peuvent rejoindre d'autres nœuds qui sont à leur portée et qui sont capables de construire des routes vers des nœuds éloignés : C'est le rôle du protocole de routage (figure 9).

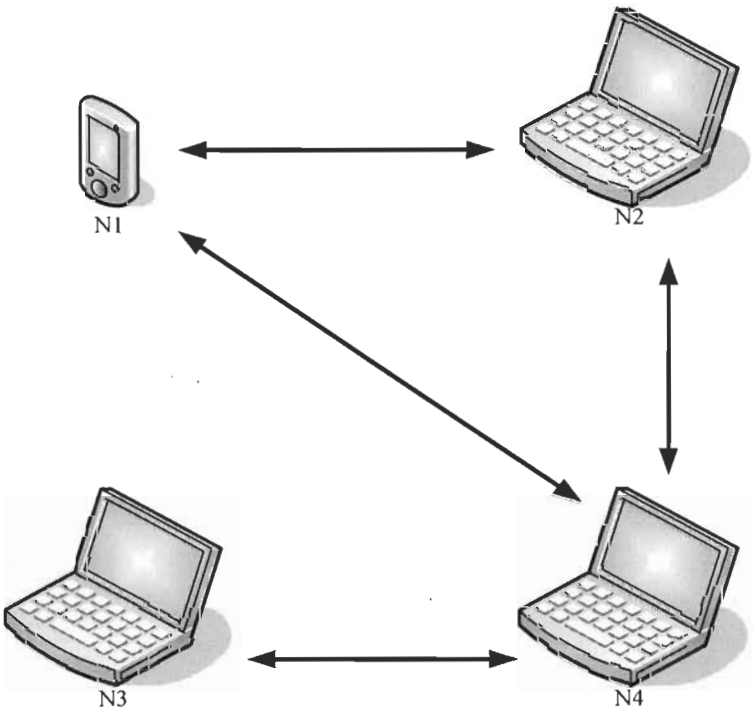


Figure 9: Communication par routage ou en mode ad hoc

3.4 Caractéristiques des réseaux ad hoc

Les réseaux ad hoc ont plusieurs caractéristiques particulières [10]:

- Topologies dynamiques : Les nœuds sont libres de se déplacer librement, ce qui amène des changements rapides et aléatoires de la topologie du réseau.
- Limitation de la bande passante avec une liaison à débit variable.

- Source d'énergie limitée: Une partie des nœuds mobiles est équipée généralement d'une batterie qui est une source limitée d'énergie.
- Utilisation limitée des autres ressources (CPU, mémoire, support de stockage, bande passante).
- Limitation de la sécurité physique: Le manque d'infrastructure au sein des réseaux sans fil mobiles cause une grande sensibilité aux menaces physiques. Toutes les liaisons utilisées sont des ondes hertziennes qui sont à la portée de tous les nœuds.
- Limitation géographique, car le déploiement de ces réseaux se fait sur des zones à rayon limité.
- La liberté des nœuds au sein des réseaux rend la connectivité très variable.
- Grande spontanéité, vu que les nœuds peuvent quitter ou se joindre au groupe sans préavis.

3.5 Les réseaux ad hoc grand public

Les recherches sur les réseaux ad hoc ont débuté au sein des laboratoires de l'armée américaine au milieu des années 60. Le but derrière ces premières recherches a été le développement d'un système pour connecter rapidement des zones difficiles à atteindre (zone sinistrée, par exemple).

Quelques années plus tard, ont commencé à apparaître des applications commerciales servant à la transmission des données, en voici quelques-unes de ces applications utilisant les réseaux ad hoc grand public [11] :

- Extension de la portée des points d'accès à l'Internet sans fil dans les environnements non desservis par les fournisseurs.
- Renseignements sur la localisation: Les réseaux de diffusion d'informations sur les autoroutes.
- Les réseaux de secours et les services d'urgence: des réseaux mises en place d'urgence dans le cas des catastrophes majeures, pour mener rapidement les opérations de sauvetage.
- Éducation : La couverture des salles de cours ainsi que l'animation des conférences virtuelles.
- Étiquettes intelligentes : La localisation facile de la marchandise.
- Réseaux de capteurs : Ce genre de réseaux est présent dans les applications environnementales, par exemple.

3.6 Les Problèmes et les contraintes des réseaux ad hoc

Vu les rôles multiples que chaque nœud participant joue dans un réseau ad hoc, la grande dynamique des nœuds, la limite d'énergie et la faible sécurité sont les grandes sources de problèmes des réseaux ad hoc:

- **La mobilité** [12]: La mobilité et la dynamique des nœuds conduisent à des changements fréquents de la topologie des réseaux. Cette propriété nécessite l'utilisation d'algorithmes de routage totalement distribués.
- **L'énergie** [13]: Vu le manque d'infrastructure, le manque des points de raccordement entre les membres ainsi que la topologie dynamique, certains nœuds participants jouent le rôle de routeurs pour assurer le routage du grand flux de

paquets circulants sur le réseau et l'acheminement des clés de cryptage, dans le cas où la position de ces nœuds est stratégique; ce qui peut causer une grande consommation de l'énergie et donc l'épuisement rapide de la source (la batterie).

Avec la liberté des nœuds à se déplacer dans le réseau, l'échange des paquets de contrôle grandit au fur et à mesure. Plus la topologie change vite et profondément, plus la charge du trafic du contrôle sera importante. Ces nœuds peuvent rapidement épuiser leurs énergies.

- **La faible sécurité** : La sécurisation du trafic de contrôle, la sécurisation des informations et la gestion du routage au sein des réseaux ad hoc représentent un grand défi qui est causé par le manque de l'infrastructure fixe [13] [12]. Dans le cas des réseaux ad hoc, les problèmes de sécurisation sont tellement complexes que l'ajout de nouveaux nœuds mobiles au réseau nécessite un contrôle strict afin de détecter et d'arrêter les nœuds "intrus" qui détourneraient ou perturberaient le fonctionnement même du routage.

3.7 Les réseaux ad hoc multicast

Actuellement, la diffusion de données vers plusieurs destinations est un scénario très avantageux dans les contextes d'utilisation des réseaux en général [14]. Tel que présenté au chapitre précédent, les réseaux à diffusion offrent, généralement, le concept de groupe de multidiffusion ou *le multicast*, de sorte qu'un émetteur puisse considérer l'ensemble de ses destinations comme une seule entité. Il impose alors au protocole de multidiffusion de distribuer des données à tous les membres du groupe considéré.

Comme nous nous intéressons à l'étude de la faisabilité de ce genre d'applications dans les réseaux sans fil, deux fonctionnalités nous paraissent primordiales : assurer des communications Multicast et assurer certains critères de qualité de service et de sécurité.

Les méthodes de gestion des clés multicast dans les réseaux filaires ne répondent pas complètement aux exigences des réseaux sans fil mobiles. Donc, il faut les adapter pour répondre aux exigences de ses derniers [14].

3.7.1 Impact de la Mobilité sur le routage au sein des réseaux ad hoc

Caractériser les modèles en termes de mobilité et de connectivité peut aider à expliquer l'impact de ces deux dernières sur l'exécution du routage. Des chercheurs comme, Pandey et Zappala [15], Bai, Sadogopan et Helmy [16] ont mis en application ces deux caractéristiques [17] tout en fixant un certain nombre de paramètres pour arriver à la fin à mesurer l'effet de la mobilité sur les modèles de routage dans les réseaux sans fil.

Pour chacun de ces modèles, ces auteurs ont employé une gamme de vitesses, y compris le cas où tous les nœuds sont statiques. Ceci examine le cas typique où un groupe de personnes se réunissent spontanément dans un espace de réunion dans les lieux du travail par exemple. Ils ont établi les relations mathématiques ci-dessous.

Symbole	Interprétation
$\mathcal{N}$	Nombre total des nœuds dans un même réseau
R	Le rayon de portée radio
$\mathcal{V}_i(t)$ $\mathcal{V}_j(t)$	La vitesse du nœud i à l'instant t La vitesse du nœud j à l'instant t
$\mathcal{D}_{i,j}(t)$	La distance entre les nœuds i et j à l'instant t
RD	$\frac{v_i(\vec{t}).v_j(\vec{t})}{\lvert v_i(\vec{t}) * v_j(\vec{t})\rvert}$ (La direction relative entre les deux nœuds)
SR	$\frac{\min(v_i(\vec{t}), v_j(\vec{t}))}{\max(v_i(\vec{t}), v_j(\vec{t}))}$ (Le rapport de vitesse entre les deux nœuds)
$\mathcal{P}_{i,j}$	$\begin{cases} 1, D_{i,j}(t) < R \\ 0, D_{i,j}(t) > R \end{cases}$

Tableau 2 : Résumé de symboles utilisés [18]

Pour tous les modèles, ces chercheurs ont essayé d'éviter des tours pointus et des changements soudains de vitesse en employant des vecteurs d'accélération et de décélération [17].

De cette manière, ils ont trouvé qu'une dépendance spatiale ainsi que le nombre moyen de changements de liens peut différencier d'un modèle de mobilité à l'autre, chose qui aide à expliquer l'exécution du routage multicast. Ceci est défini par :

- *La dépendance spatiale* : La dépendance spatiale [15] caractérise le degré auquel deux nœuds se déplacent dans une direction semblable avec une vitesse semblable. Pour deux nœuds, ***i*** et ***j*** au temps ***t***, la dépendance spatiale est définie comme suit:

$$D_{spacial}(i,j,t) = RD\left(v_i(\vec{t}), v_j(\vec{t})\right) * SR\left(v_i(\vec{t}), v_j(\vec{t})\right)$$

Soient RD la direction et SR le rapport de vitesse comme défini dans le tableau 3. Plus les nœuds sont proches, plus la dépendance spatiale est importante. Pour que ce rapport soit applicable, la condition suivante s'applique également [16] :

$$D_{i,j}(t) > C_1 * R \rightarrow D_{spacial}(i,j,t) = 0$$

C_1 est une constante qui a été fixée à 2.

- *Le nombre de changements de lien :* C'est le nombre de changements de lien pendant une simulation [16]. Un changement de lien est défini comme événement quand deux nœuds (relevant de la portée radio) n'ont pu communiquer directement entre eux.
- *La densité du Voisinage :* C'est Le nombre de nœuds à la portée radio d'un nœud donné. Pour un nœud ***i***, la densité du voisinage est définie comme :

$$N_i = \sum_{j=1}^{N} P_{i,j}$$

Où ***p*** est déterminé si deux nœuds sont dans la portée radio l'un de l'autre.

La valeur d'accessibilité: Le nombre de nœuds qui sont accessibles via la transmission par le biais du réseau ad hoc. Nous mesurons cette valeur en utilisant un algorithme récursif [18].

3.8 Conclusion

Les réseaux sans fil permettent de relier très facilement des utilisateurs entre eux. De plus, l'installation de tels réseaux ne demande pas de lourds aménagements des infrastructures existantes comme c'est le cas avec les réseaux filaires, de plus dans le cas des réseaux ad hoc aucune infrastructure n'est demandée. En contrepartie, des problèmes relatifs aux transmissions radio et/ou à la mobilité se posent à nous. Cette dernière est un facteur clé au niveau de la qualité des services (transmission des paquets), de la bonne exploitation de la bande passante du réseau, ainsi que de la sécurisation qui reste un grand défi pour ce genre de réseaux. La mobilité des nœuds peut être la cause de la déconnexion fréquente, et ce phénomène est d'autant plus important lors des communications de groupe au sein des réseaux sans fil ad hoc. Ce qui peut perturber l'opération de distribution des clés de groupe multicast.

Dans le chapitre 4, nous présenterons les différentes approches de distribution des clés de cryptage pour les communications multicast dans les réseaux filaires et sans fil. Enfin, nous présenterons une comparaison de ces approches.

Chapitre 4 - Approches de gestion de clés pour les communications multicast sur les réseaux ad hoc

4.1 Introduction

La confidentialité prend une place très importante dans le cadre des applications en multicast où le nombre de récepteurs est presque indéterminé. Seuls les membres d'un groupe prédéfini doivent avoir la possibilité d'accéder aux données en clair. Le meilleur moyen pour protéger ces données est l'utilisation des algorithmes de chiffrement et de déchiffrement. Par conséquent, pour qu'un membre d'un certain groupe puisse avoir accès aux données en clair destinées à ce groupe, il doit recevoir une (ou plusieurs) clé (s) de déchiffrement.

Tout au long de ce chapitre et à travers la littérature, nous allons faire une étude comparative des principales approches de distribution et de gestion des clés de cryptage utilisées dans les communications de groupe.

4.2 Les services de sécurité et gestion des clés

La modification et la destruction non autorisées des données, sont les grandes menaces pour les systèmes ouverts vers l'extérieur [19] comme le cas des réseaux ad hoc. Pour contrer ces menaces, on fait appel à des services de sécurité qui utilisent des mécanismes basés sur le chiffrement à une ou plusieurs clés [20]. Ces mécanismes sont:

La non-répudiation [20]: Elle protège les participants contre les envois et les réceptions démentis.

L'intégrité [20]: Elle sert à vérifier l'exhaustivité des données.

Le contrôle d'accès [20]: Il ne donne l'autorisation d'accès aux données qu'aux utilisateurs adhérant.

L'authentification [20] : Le mécanisme se divise en deux types, soit l'authentification d'entité et l'authentification de l'origine. En ce qui concerne l'authentification d'entité, chaque nœud s'assure de l'identité de l'autre nœud avec qui il va communiquer, tandis que dans l'authentification de l'origine, le nœud destination doit s'assurer qu'il reçoit bien les données du nœud source.

Ces services de sécurité assurent la ***confidentialité*** des informations [20], élément primordial pour créer une session privée d'échange, circulant entre les nœuds participants.

La majorité des services mentionnés précédemment sont assurés grâce aux mécanismes de la cryptographie [20]. En effet, la cryptographie [21] est l'ensemble de technique qui sert à rendre les données confidentielles en les *chiffrant* au niveau du nœud source et en les *déchiffrant* au niveau du nœud destination. Les deux extrémités doivent se mettre d'accord sur une ***clé*** de cryptage.

Nous distinguons deux types d'algorithmes de codage/encodage : Les algorithmes *symétriques* [21] dans lesquels les deux parties de la communication disposent d'une même clé de cryptage, et les algorithmes *asymétriques* [21] où on distingue l'utilisation de deux clés, l'une publique connue par tous les participants de la session et l'autre privée qui est personnelle pour chaque participant.

Pour assurer l'authentification, un troisième élément digne de confiance appelé le contrôleur de groupe peut entrer en jeu. Au niveau de ce contrôleur se fait la génération, la sauvegarde, la certification et la diffusion des clés.

La figure 11 représente l'emplacement de tous les services qui entre dans le mécanisme de sécurité des communications. Le service de gestion des clés [22] peut être considéré comme la base de la sécurité.

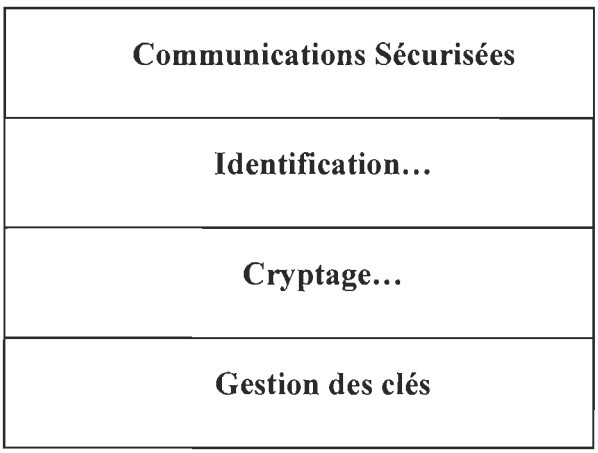


Figure 10 : Position de la gestion des clés dans les systèmes de sécurisation

Comme on peut le voir dans [23] et [11], l'établissement et la diffusion de la clé de groupe est le grand handicap de la sécurité des communications de groupe.

4.3 Approche de gestion des clés de groupe

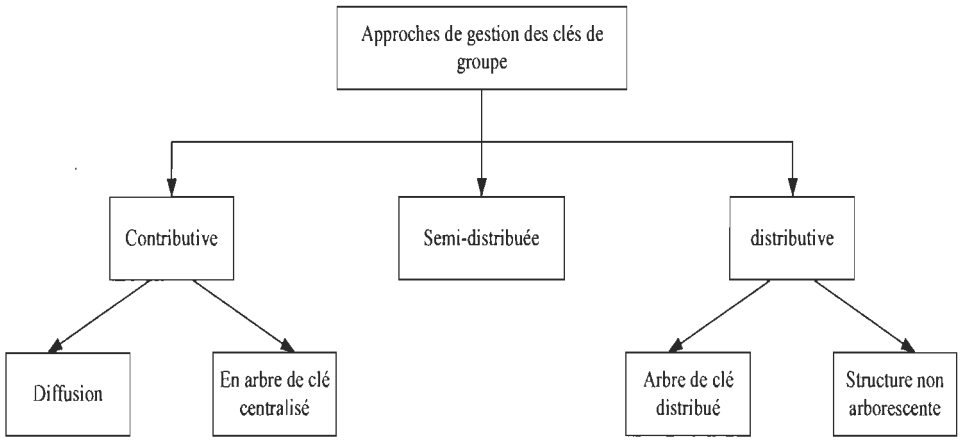


Figure 11 : Approches de gestion des clés de groupe

Comme nous avons déjà mentionné, la clé du groupe est une information qui ne doit être connue que par les participants à la session et ils doivent la garder secrète. La clé de groupe peut être établie soit, d'une manière **distributive** [21], où le contrôleur de groupe assure en toute sécurité la génération et la distribution d'une clé secrète à tous les participants à la session, soit d'une manière ***contributive*** [24], où la clé est générée avec la contribution de tous les membres du groupe. La figure 11 présente les différentes approches de gestion ou de distribution des clés lors d'une communication multicast.

4.3.1 Approches contributives

4.3.1.1 Diffusion

4.3.1.1.1 Secure lock

Le principe de la méthode S***ecure lock*** [25] est l'établissement de la clé du groupe avec une seule diffusion produite du contrôleur du groupe et selon les auteurs G.H. Chiou et W.T. Chen [25], la méthode utilise le théorème des restes chinois:

Théorème [25] :

Si R_1, ..., R_n sont des entiers positifs supérieurs à 2,

Pour $L=N_1 * N_2 * N_3 * ... * N_n$:

$$X \equiv R_1 \bmod N_1$$
$$\dots$$
$$X \equiv R_i \bmod N_i$$
$$\dots$$
$$X \equiv R_n \bmod N_n$$

Une seule solution est admise et qui est:

$$X=\left(\sum_{i=1}^{n}\left(L/N_i\right) * R_i * f_i\right) \bmod L$$

avec $X_i = X/R_i$ et $L_i = X_i^{-1} \bmod R_i$ pour $1 \leq i \leq n$.

- Lors de l'ajout d'un nouveau membre, une clé secrète lui est attribuée par le contrôleur du groupe accompagné d'un nombre R_i positif.
- Lors de la diffusion d'un message par le contrôleur du groupe:
    1. Le contrôleur produit un nombre R pour le crypter;

2. R à son tour est chiffrée par la clé secrète r_i de chaque participant;
3. Pour arriver enfin à obtenir des clés chiffrées $R_{i\ (i=1\ldots n)}$ $(R_i=\{R\}r_i)$;
4. Trouver X :

$$X=R_1\ (mod\ N_1)$$

...

$$X=R_n\ (mod\ N_n)$$

5. Diffuser X;
6. À la fin les participants arrivent à décrypter le message à l'aide de la clé R récupérée.

Malgré la puissance de cette méthode, elle a un grand inconvénient qui est celui de la lourdeur des calculs effectués au niveau du contrôleur à l'aide du théorème des restes chinois.

4.3.1.1.2 Secure Multicast Protocol

Nous présentons aussi une autre méthode qui est simple et efficace : la méthode ***Secure Multicast Protocol*** [26] où la transmission de données passe par trois étapes:

→ **L'étape d'envoie**: Dans cette étape, l'expéditeur construit un message de données qui contient trois composantes : La première contient l'identification de l'expéditeur (*suid*) et une identification de message (*msgid*), la deuxième contient les données chiffrées avec une clé du message K_{msg} que l'expéditeur produit de façon aléatoire et la troisième composante du message contient la clé K_{msg} chiffrée avec la clé K_{suid} établie entre le contrôleur du groupe et le membre expéditeur.

Chacune des trois composantes est assemblée dans le message de données envoyé sur un canal multicast sécurisé.

→ **L'étape de vérification**: Lors de la réception du message de données de l'expéditeur, le contrôleur du groupe lit la liste courante d'adhésion pour vérifier que l'expéditeur est en effet un membre valide du groupe.

→ **L'étape de réception**: Lors de cette phase, le récepteur reçoit le message de données et le déchiffre avec la clé K_{msg}.

Le traitement des messages prend un temps assez élevé [26]. D'ailleurs, c'est le seul et le principal handicap de cette méthode.

4.3.1.2 Approches en arbre de clés centralisé

Dans l'approche en arbre de clé centralisé, l'arbre des clés est géré par le contrôleur du groupe [27].

Les nœuds (participants du groupe) sont associés aux feuilles et portent chacun une KEK (Key-Encryption-keys) qui est personnelle à chaque nœud. Ce dernier reçoit aussi les KEKs de tous les participants allant jusqu'à la racine ainsi que la clé du groupe qui est en même temps la clé de la racine.

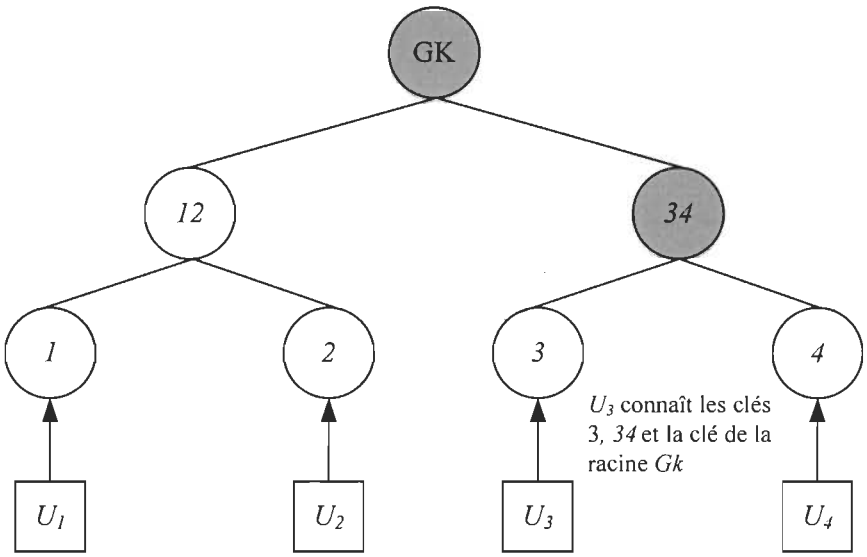


Figure 12 : Architecture générale de l'arbre binaire de clés

4.3.1.2.1 Approche de Wallner

Wallner et coll. [27] et Caronni et coll. [28] sont les premiers utilisateurs du principe de l'arbre centralié des clés de groupe. Leur méthode consiste à créer un arbre binaire des clés (KEKs) à l'aide du GTEK (*Group Traffic Encryption Key).*

Lors d'une opération d'ajout ou de départ d'un participant au niveau du groupe, le contrôleur du groupe met à jour les clés KEKs des nœuds qui se trouvent sur le chemin du nœud en question jusqu'à la racine en procédant comme suit [28]:

- La reproduction des clés des nœuds qui se trouvent sur le chemin jusqu'à la clé de la racine GK (*Group Key*).
- La constitution d'un message de taille $2log_2n$ contenant toutes les nouvelles clés après les avoir chiffrés de nouveau avec des KEKs.

Le contrôleur du groupe peut employer les KEKs pour exclure un simple participant ou un groupe de participants.

Dans l'exemple de la figure 13, nous illustrons l'ajout d'un nouvel arrivant qui est u_5.

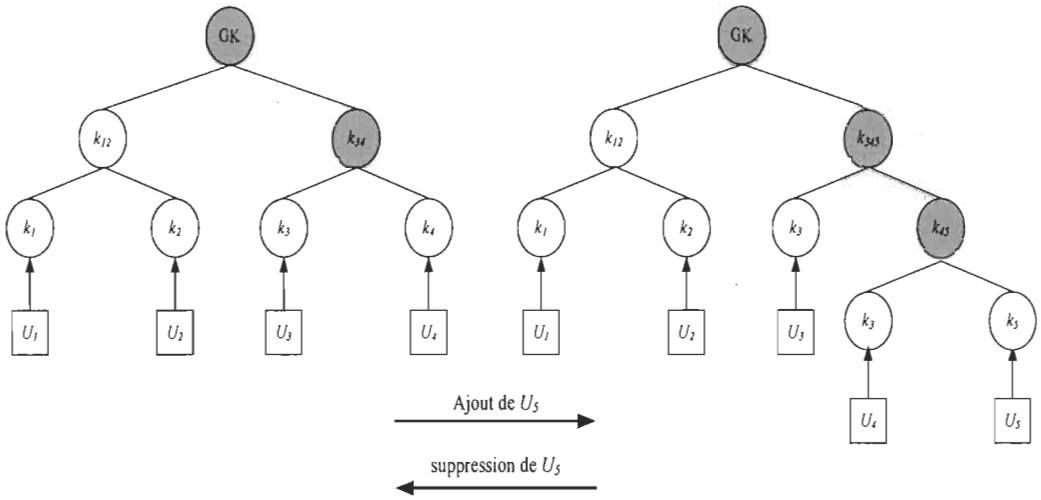


Figure 13 : Approche de Wallner [27]

4.3.1.2.2 Autres approches

Nous trouvons, aussi, dans la littérature, d'autres approches à arbre de clé centralisé, nous pouvons citer à titre d'exemple; l'approche ELK (*Efficient Large Group Key distribution*) développée Perrig et call. [29], l'approche CFKM (Centralized Flat Key Management) développée par Waldvogel et call. [30].

4.3.1.2.3 Inconvénients

Les grands inconvénients de ce genre d'approches sur lesquels plusieurs auteurs se sont mis d'accord sont:

- À cause de leur mauvaise convivialité aux grands réseaux, les approches à arbre de clés centralisées ne répondent pas au facteur d'échelle.
- Le contrôleur de groupe qui joue le rôle du serveur central, peut constituer une cible facile d'attaques.

4.3.2 Approches semi-distribuées

Dans les approches semi-distribuées, la gestion de clés de groupe n'est plus dédiée au contrôleur du groupe tout seul, mais à plusieurs unités qui sont soit dans ou en dehors du groupe.

4.3.2.1 Approche SMKD

L'approche **SMKD** (*Scalable Multicast Key Distribution*) conçue par Ballardie [31], se compte parmi les premiers travaux de recherche dans les approches semi-distribuées.

Ce protocole est basé sur le routage multicast CBT (Core Based Trees) [32] où un arbre de clés concentré autour du *Core* principal (le nœud père) et des Cores secondaires est construit. Le Core principal crée l'ACL (*Access Control List* où Liste de Contrôle d'accès) tout en procédant comme suit [31]:

- Il génère une clé de session GTEK (*Group Tree Encryption Key*)
- Il génère une clé de décryptage de clés GKEK (*Group Key-Encryption-Keys*).
- Par après, le Core principal diffuse ACL, GTEK et GKEK à tous les nœuds appartenant à l'arbre.

4.3.2.2 Approche Iolus

Dans l'approche **Iolus** [33] le groupe multicast est organisé d'une façon hiérarchique en un certain nombre de sous-groupes indépendants. La gestion de ces sous-groupes est confiée aux GSA (Group Security Agents) ou contrôleurs de sous-groupes. Par contre, la gestion de tout le groupe se fait au niveau de la racine de l'arbre GSC (group Security Controller). Les contrôleurs des sous-groupes GSA peuvent jouer le rôle des GSS (Group

Security Skill) servant de proxy au contrôleur principal (figure 14). Les GSA sont groupés par niveau dans une hiérarchie de distribution. Chaque GSA est responsable des transmissions de toute information aux membres de son sous-groupe.

En principe, à la création du groupe par le GSC, ce dernier fixe une politique de sécurité; les GSI et les membres de leurs sous-groupes doivent respecter les conditions de sécurité. Vu la relative indépendance de chaque sous-groupe, l'ajout d'un membre au groupe ou son départ se fait au niveau du sous-groupe local, en possédant une clé de trafic propre à lui et il n'y a pas de clé globale pour tout le groupe. Vu que le GSC n'est en principe qu'un contrôleur du sous-groupe de la racine, seule la clé de chaque sous-groupe doit être changée.

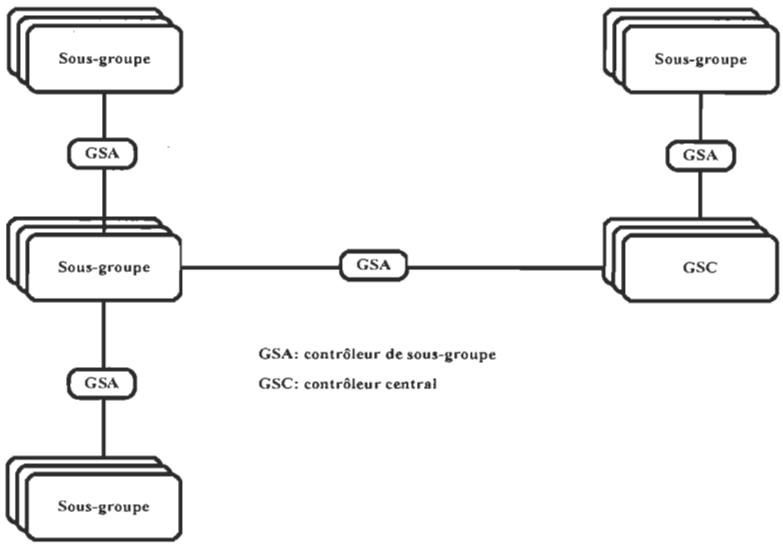


Figure 14 : Architecture de l'approche Iolus

4.3.2.2.1 Les ajouts et les départs

Nous commençons par l'ajout d'un nœud au groupe. En effet, ce dernier localise le GSA adéquat et lui communique une demande d'ajout à l'aide d'un canal unicast. Une fois que le GSA reçoit la demande de l'ajout [33]:

- Il vérifie sa base de données et décide s'il approuve ou non cette demande.
- Si la demande est approuvée, il produit une clé secrète K_{GSA} et il ne la partage qu'avec le nouveau nœud en lui envoyant cette dernière via un canal privé.
- Il génère une nouvelle clé de sous-groupe et l'envoie aux anciens membres.
- La liste d'accès du groupe ACL sera elle aussi envoyée dans le cas ou le nouveau membre est un GSA.

En ce qui concerne les départs des membres, ces derniers se produisent de deux manières [33]:

- Le membre souhaite quitter volontairement le sous-groupe. Dans ce cas, il envoie une requête de départ au GSA en utilisant un canal sécurisé. Le GSA génère une nouvelle clé du sous-groupe, le chiffre avec la clé K_{GSA} qu'il partage avec chaque nœud.
- Le GSA veut expulser un membre du sous-groupe il envoie une notification directement à celui-ci.

Dans les deux cas, la clé du sous-groupe doit être changée en éliminant la participation du membre partant.

On constate que Iolus est assez puissant vu l'organisation des membres en sous-groupes ce qui facilite la mise à l'échelle. Ainsi, lorsqu'un événement d'ajout ou de départ d'un membre se produit, cela n'affecte que les clés du sous-groupe auquel il appartient. Par contre, l'inconvénient principal de Iolus réside dans la possibilité de la non-appartenance des contrôleurs GSA au groupe, ainsi que le temps de chiffrement peut être long.

Dans la littérature, on constate l'existence d'autres approches semi-distribuées sauf qu'elles tournent toutes autour du même principe que Iolus, chose qui nous laisse les considérer comme des améliorations.

KHIP (*Keyed HIerarchical multicast Protocol*) est un exemple d'approche qui mélange entre les principes de SMKD et ceux de Iolus, il est proposé par shields et Garcia-Luna-Aceves [34]. **DEP** (*Dual Encryption Protocol*) proposé par Dondetti et coll. [35] et qui est une pure extension et amélioration d'Iolus. Aussi il y a Molva et Pannetrat [36] qui ont proposé une autre amélioration d'Iolus en ajoutant l'utilisation de la technique *Reversible Cipher Sequences* qui sert à limiter l'accès des nœuds aux données systèmes (tout en gardant le même principe de Iolus).

4.3.3 Approches distributives

Dans ces approches, la gestion et la distribution de la clé de groupe est une tâche confiée à un ou des contrôleurs de groupe.

Nous distinguons deux types des approches distributives: d'une part les approches arborescentes (hiérarchiques) et d'autre part les approches distribuées (non centralisées). La majorité des approches distribuées sont des extensions du protocole de Diffie-Hellman [37].

4.3.3.1 L'algorithme de Diffie-Hillman et les approches basées sur lui

4.3.3.1.1 L'algorithme de Diffie-Hillman

Les paramètres présentés dans le tableau suivant vont être cités tout au long de la description des approches basées sur Diffie-Hillman [37]:

Notations	Descriptions
n	nombre de participants
M_i	ième membre du groupe
N_i	valeur aléatoire générée par le membre M_i
S, T	sous-ensembles de $\{N_1, ...N_n\}$
K_n	clé partagée par les n membres
$\pi(S)$	produit de tous les éléments de S
g, p	paramètres Diffie-Hillman

Tableau 3 : Sommaire des paramètres de l'algorithme de Diffie-Hillman

L'algorithme de Diffie-Hillman [37] est une nouvelle méthode de cryptage basée sur l'utilisation d'une clé secrète entre deux entités. Cette méthode s'appelle l'*approche à clé publique*. Cette dernière permet de remédier aux problèmes d'échanges de clé rencontrés par les méthodes à clé symétrique. La méthode Diffie-Hillman permet à deux participants **A** et **B** d'établir et de partager une seule clé confidentielle, en procédant comme suit [37]:

Les deux participants se mettent d'accord publiquement sur les paramètres de Diffe-Hillman, ***g*** (le générateur > 2) et ***p*** (≪ ***g***), ***p*** et ***g*** sont des nombres premiers.

- Chaque nœud génère d'une façon confidentielle et aléatoire sa clé privée;

  x ≪ ***p-1*** est la clé privée de **A**

  y ≪ ***p-1*** est la clé privée de **B**.

- Chaque participant procède séparément au calcul d'une clé publique:

  A calcule $y_1 = g^x \bmod p$ et l'envoie à B

B calcule $y_2=g^y \bmod p$ et l'envoie à A

A calcule $k = (z_1)^x \bmod p$

B calcule $k'= (z_2)^y \bmod p$

La valeur de $K=g^{xy} \bmod p$ qui est considérée comme la clé symétrique est le secret partagé par **A** et **B**.

4.3.3.1.2 L'approche de Fiat et Naor

Dans l'approche de Fiat et Naor [38], les chercheurs ont adapté les principes de l'algorithme de Diffie-Hellman pour être utilisé dans des groupes [38].

Dans ce protocole, le système est initialisé par un nœud T_c digne de confiance en suivant la procédure suivante [38] :

→ Il choisit P et Q deux nombres premiers.

→ Il diffuse le produit $N= P*Q$.

→ Il génère aléatoirement et d'une manière secrète le paramètre Diffie-Hellman g.

→ Le membre M_i assigne la clé $g_i = g^{P_i}$.

→ T_c communique nombre premier aléatoire p_i et le nombre p_j déjà généré pour un membre M_j, (avec i, j ∈ ∪, ($i \neq j$)).

Une fois que chaque membre reçoit tous les messages diffusés par tous les nœuds participants, il calcule individuellement la clé du groupe par la formule suivante :

$$g_i^{\prod_{j \epsilon T-\{i\}} p_j} mod\ N$$

4.3.3.1.3 L'approche de Ingemarson

Dans l'approche de Ingemarson [39], qui est carrément différente de celle de Fiat et Naor, nous définissons, tout d'abord, un anneau logique CKDS (*Conference Key Distribution System*) de l'ordre j pour avoir un système de tel sort que la clé de conférence soit :

$$K^{(j)} \triangleq g^{S(j)(\Omega)} \bmod p$$

La clé de groupe est simultanément distribuée à tous les participants M_i de la façon suivante (figure 16) [39]:

- Comme déjà mentionné, les stations sont reliées dans un anneau, de sorte que, le participant M_i envoie toujours des messages au participant M_{i+1} et M_{i-1} jusqu'au participant 0.
- Chaque participant M_i passe la clé du groupe qu'il a reçu du nœud qui le précède M_{i-1} au nœud qui le suit M_{i+1}, après qu'il l'élève à la puissance N_i.
- La clé du groupe est calculée par les membres du groupe au bout de *(n-1)* tours.

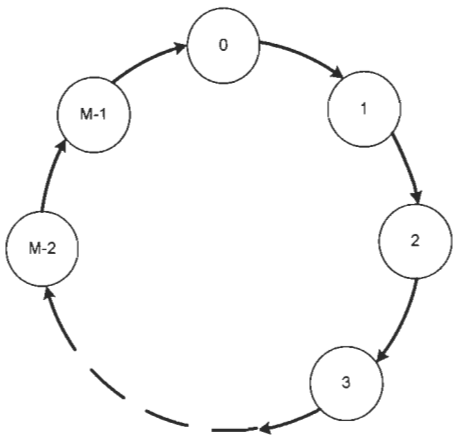


Figure 15 : L'anneau logique CKDS

Durant un tour, la communication des messages d'un membre M_i au membre qui le suit se fait comme suite [39]:

Tour 1 : M_i envoie g^{Ni}
Tour 2 : M_i reçoit $g^{N_{i-1}}$ et envoie $g^{N_i N_{i-1}}$
...
Tour n : M_i reçoit $g^{N_{i+1}...N_n N_1 ... N_{i-1}}$;
Arrivant à la fin à calculer la clé $K_n = g^{N_1 . N_2 ... N_n}$

Inconvénient de l'approche de Ingemarson:

La ré-exécution de toutes les opérations du protocole à chaque tour, rend l'approche inadaptée aux mises à jour fréquentes des clés [40].

4.3.3.1.4 Les approches de la suite CLIQUES

Trois autres extensions de Diffie-Hellman portant le nom GDH (*Generic Group Diffie Hellman Agreement*) ont été proposées par Steiner et coll. [40]. GDH.1, GDH.2 et GDH.3 sont utilisées dans le contexte de groupe.

Dans GDH, plusieurs valeurs intermédiaires entrent en jeux. Ce qui permet d'éviter la ré-exécution de toutes les opérations du protocole lors d'une mise à jour de clés comme dans le cas de l'approche de Ingemarson [39].

4.3.3.1.4.1 L'approche GDH.1

Comme illustrer dans la figure 17, l'exécution de l'approche GDH.1 passe par deux étapes :

→ Étape du *flux ascendant*: Cette phase qui est initialisée par le premier membre qui assemble les participations de tous les membres du groupe [41]:

1. Chaque membre M_i reçoit du membre qui le précède $M_{(i-1)}$ de (i-1) valeurs (g^{N_1}, $g^{N_1 N_2}$, …, $g^{N_1 N_2 N_{i-1}}$).

2. La dernière valeur de la suite est élevée à la puissance N_i par les membres M_i, (g^{N_1}, $g^{N_1 N_2}$, …, $g^{N_1 N_2 \ldots N_{i-1}}$, $g^{N_1 N_2 \ldots N_i}$). Cette dernière collection de valeurs est transmise d'un membre au membre qui le suit.

L'étape du flux ascendant tient à sa fin lorsque le flux atteint le dernier membre.

→ Étape du *flux descendant* : Dans cette étape initialisée par le dernier membre du groupe, chaque membre M_i effectue i exponentiations servant dans :

- Le calcule de la clé K_n et de *(i-1);*
- L'obtention des valeurs intermédiaires dont les autres membres ont besoin pour calculer la clé du groupe.

Sur la figure 17, nous présentons les étapes de l'approche GDH.1, qui sont comme suit [41]:

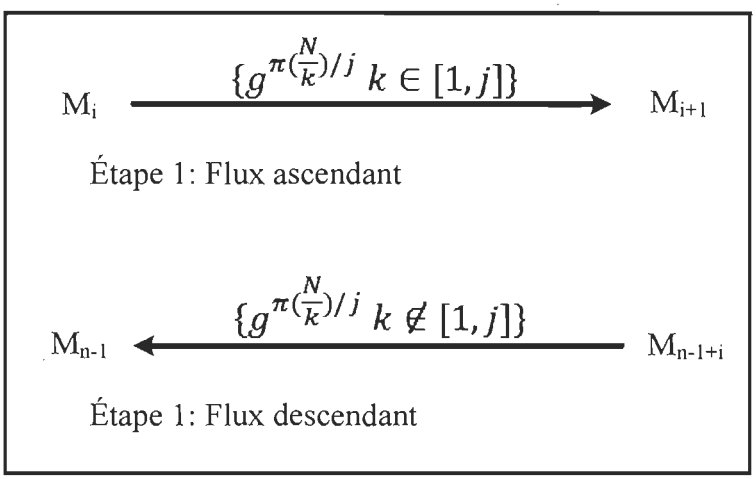


Figure 16 : L'approche GDH.1

1. Premièrement, le membre M_i commence par élever toutes les clés qu'il a reçues à la puissance N_i.

2. M_n élimine la clé K_n de la collection, nomme g^{N_n} comme premier élément et envoie toutes les informations au membre qui le précède.

3. Chaque membre M_i retire la dernière valeur ($K_n=g^{N_1 * N_2 * \ldots N_n}$) de la suite des valeurs après les avoir élevé à la puissance N_i.

4. M_i diffuse à tous les membres du groupe les dernières valeurs intermédiaires obtenues $\{g^{\pi (N/k \in [1, j])/j \in [1, i]}\}$.

À cause des (i+1) exponentiations effectuées au niveau de chaque membre, l'approche GDH.1 comporte 2(n-1) tours.

4.3.3.1.4.2 L'approche GDH.2

L'approche GDH.2 n'est qu'une amélioration de sa cousine GDH.1 et l'amélioration qui porte est une réduction au niveau du nombre de tours à n [40]. Comme pour le GDH.1, l'étape du flux ascendant est encore employée pour rassembler les participations de tous les membres du groupe, sauf que chaque membre doit composer deux types de valeurs : une première valeur intermédiaire et une seconde valeur cardinale.

Pour mieux comprendre le contexte du GDH.2 et sa différence du GDH.1, regardons l'exemple cité dans [41] :

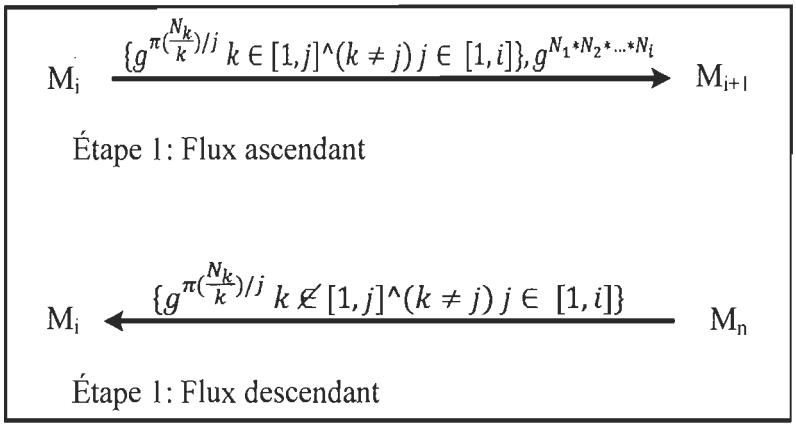


Figure 17 : L'approche GDH.2

- Le participant M_4 reçoit $\{g^{N_1N_2N_3}, g^{N_1N_2}, g^{N_1N_3}, g^{N_3N_2}\}$ de la part de M_3,
- Envoie $\{g^{N_1N_2N_3N_4}, g^{N_1N_2N_3}, g^{N_1N_2N_4}, g^{N_1N_3N_4}, g^{N_2N_3N_4}\}$,
- La valeur cardinale de cet exemple est $g^{N_1N_2N_3N_4}$,
- avant que le flux ascendant atteigne M_n, la valeur cardinale devient $g^{N_1\ldots N_{n-1}}$. M_n est ainsi le premier membre de groupe qui calcule la clé K_n et aussi le membre ayant l'index le plus élevé, calcule la dernière série de valeur intermédiaire.
- Chacun des membres calcule la clé correspondante à son numéro.

4.3.1.4.3 L'approche GDH.3

Dans l'approche GDH.3 [40], le nombre des calculs effectués au niveau de chaque membre est réduit par rapport aux approches GDH.1 et GDH.2. Ainsi que la taille des messages qui circulent au sein du groupe ne change pas avec le changement de la taille de ce dernier.

L'approche GDH.3 s'exécute en quatre étapes, regardons l'exemple cité dans [40]:

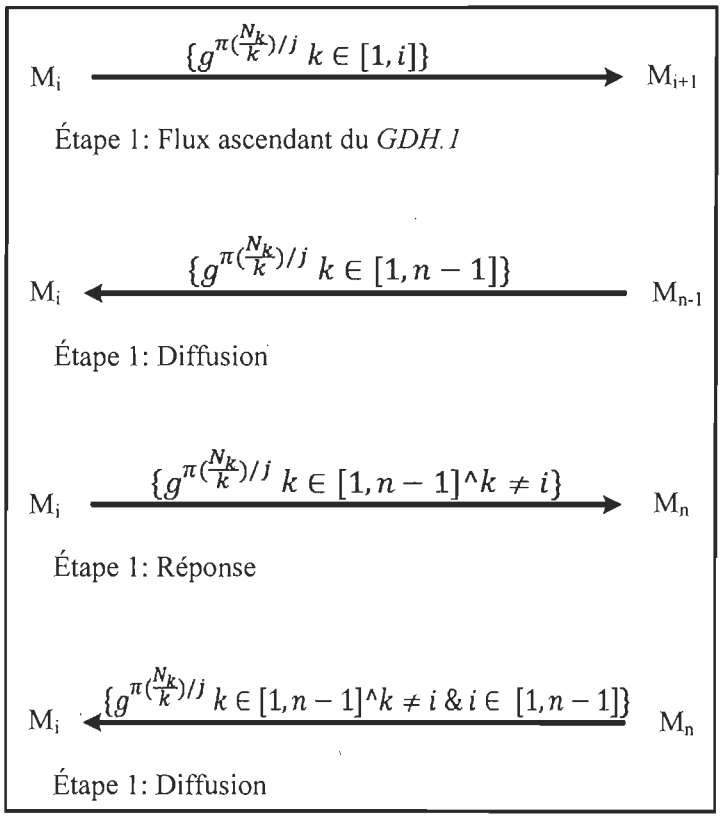


Figure 18 : L'approche GDH.3

- La première étape n'est que le flux ascendant du protocole GDH.1.
- Dans la deuxième étape, l'avant-dernier membre envoie à tous les participants la valeur $g^{N_1 \ldots N_{n-1}}$ atteint du flux ascendant.
- Dans la troisième étape, chaque membre enlève sa participation et envoie le reste au membre qui le suit.
- Dans la quatrième étape, il y a le calcul de la clé du groupe.

Au fond, le protocole GDH.3 nécessite l'exécution de *n+1* tours.

Pour les trois approches, GDH.1, GDH.2 et GDH.3, les ajouts et les départs se font de la même manière.

La ré-exécution complète de toutes les opérations du protocole lors de l'ajout et le départ d'un membre, a été évité avec les approches de la suite clique [40], car dans ces dernières il y a l'intégration du contrôleur des opérations.

4.3.3.2 Les approches en arbre

Les approches de cette catégorie sont considérées comme des extensions en arbre du protocole Diffie-Hellman [37].

4.3.3.2.1 L'approche STR

STR [44] *Skinny TRee* est la première approche contributive de gestion de clés qui utilise une structure en arbre. Ce protocole a été développé par Steer et coll. [42], en effet il est basé sur l'échange de Diffie-Hellman et assume la formation d'un groupe statique. Par contre, Kim et coll. [43] ont amélioré ce protocole pour qu'il soit utilisé dans les réseaux à groupe. Nous pouvons ajouter que le STR vient optimiser les performances des GDH.

Dans le STR, chaque membre peut agir comme sponsor en fonction de sa position dans l'arbre.

Dans l'exemple de la figure 19 [43], les participants sont présentés sous forme de feuilles et le nombre ***n*** est égal à la profondeur de l'arbre.

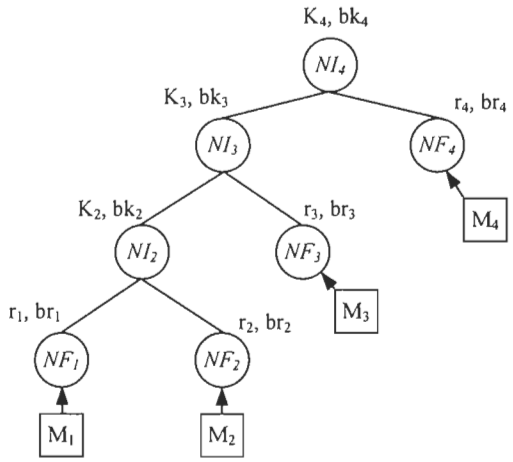


Figure 19 : L'arbre STR

→ NF_i $(1 \leq i \leq n)$ est la position qui identifie chaque feuille dans l'arbre.

→ NI_j $(2 \leq j \leq n)$ est l'identification de chaque nœud et en même temps sa position dans l'arbre.

→ r_i représente une clé que chaque participant associe à la feuille qu'il choisi d'une manière aléatoire,

→ $br_i = g^{r_i} \bmod p$ étant la clé aléatoire, (g et p les paramètres de Diffie-Hellman [37]).

À chaque nœud on associe une clé secrète x_i et une clé publique $bx_j = g^{x_j} \bmod p$

La clé x_i est calculée comme suite [43]:

$$K_i = (bx_{i-1})^{r_i} \bmod p = (br_i)^{k_{i-1}} \bmod p = g^{r_i x_{i-1}} \bmod p$$

La clé du groupe est $x_n = g^{r_n g^{r_{n-1}} \ldots g^{r_1 r_2}}$.

Chaque membre du groupe doit connaître sa propre clé publique ainsi que celles de tous les membres.

Les clés des nœuds internes sont calculées par le premier nœud NF_1 en suivant les étapes suivantes [43]:

$$x_2 = (br_2)^{r_1} \bmod p = g^{r_1 r_2} \bmod p \qquad bx_2 = g^{x_2} \bmod p$$
$$x_3 = (br_3)^{x_2} \bmod p \qquad bx_3 = g^{x_3} \bmod p$$
$$\ldots \qquad \ldots$$
$$x_n = (br_n)^{x_{n-1}} \bmod p \qquad bx_i = g^{x_i} \bmod p \text{ avec } (2 \leq i \leq n-1)$$

À la fin, NF_1 diffuse à tous les membres toutes les clés publiques bx_i.

Chaque membre peut par la suite calculer toutes les clés. Donc, la mise à jour de l'arbre des clés est faite par chaque membre. Ce qui nécessite l'élection d'un Sponsor du groupe qui est le membre le plus proche de la racine de l'arbre.

Durant l'opération de l'ajout d'un nouveau membre au groupe (figure 20) [43]:

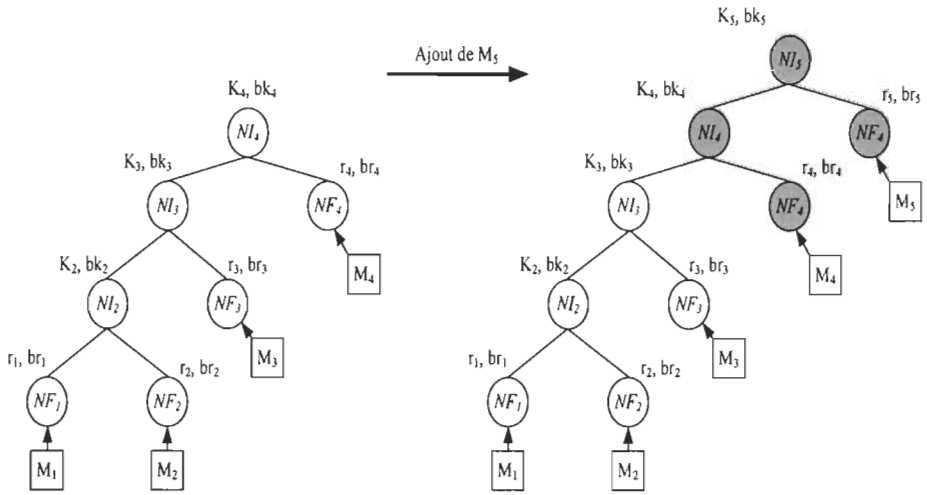


Figure 20 : Mise à jour de l'arbre après l'ajout d'un nouveau nœud (M_5)

- Le nouveau membre diffuse au groupe une demande d'ajout au groupe.

- Le sponsor calcule la clé publique du groupe bk_n.

- Le sponsor envoie au nouveau membre une copie du nouvel arbre des clés publiques.

- Un nouveau nœud racine est élu suite à l'incrémentation des indices de tous les nœuds du groupe. L'ancienne racine devient la feuille M_{n+1}.

- À la fin de l'opération, chacun des membres calcule la nouvelle clé du groupe.

Durant l'opération du départ d'un membre du groupe (figure 21) [43]:

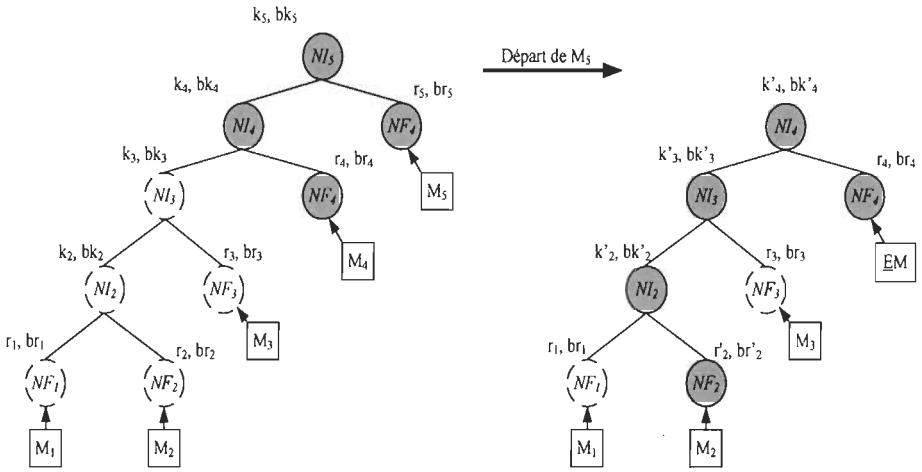


Figure 21 : Mise à jour de l'arbre après le départ d'un nœud (M_5)

- Le rôle du sponsor du groupe sera occupé par le nœud se trouvant en dessous du nœud partant.

- Le nouveau sponsor recalcule toutes les clés jusqu'à la racine et génère une nouvelle clé secrète.

- Le sponsor diffuse à tous les participants le nouvel arbre contenant les nouvelles clés publiques.

- Tous les nœuds du groupe mettent individuellement à jour l'arbre des clés.

- À la fin, les nœuds participants calculent la nouvelle clé du groupe.

4.3.3.2.2 L'approche TGDH

Développée par Perrig et coll. [44] et revue par Baras et Striki [45], l'approche TGDH (Tree-based Group Diffie-Hellman) a pour but de minimiser le nombre de clés calculées de n clés comme dans le cas de STR à $log_2 n$ clés, tout en gardant le même principe de l'arbre binaire.

D'après Perrig et coll. [44] ainsi que Baras et Striki [45], l'arbre de clés est construit de manière ascendante et selon les ajouts des membres, la racine est au niveau 0 et les feuilles les plus profondes sont au niveau h qui représente la profondeur de l'arbre (figure 22).

Les nœuds de l'arbre sont étiquetés $<l,i>$ et à chaque clé privée $K_{<l,i>}$, on associe une clé cachée $BK_{<l,i>} = f(K_{<l,i>}) = g^{K}{}_{<l,v>} \bmod p$.

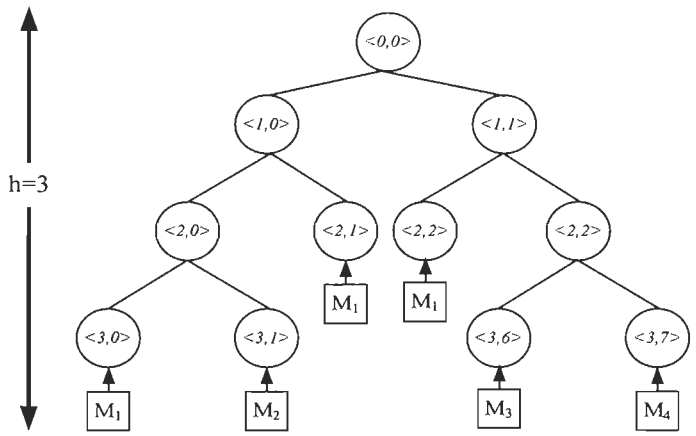


Figure 22 : Création de l'arbre de clé en TGDH

Pour l'ajout ou le départ d'un nœud du groupe, il faut tout d'abord élire un sponsor de groupe [45].

Dans le cas d'un ajout [45] (figure 23):

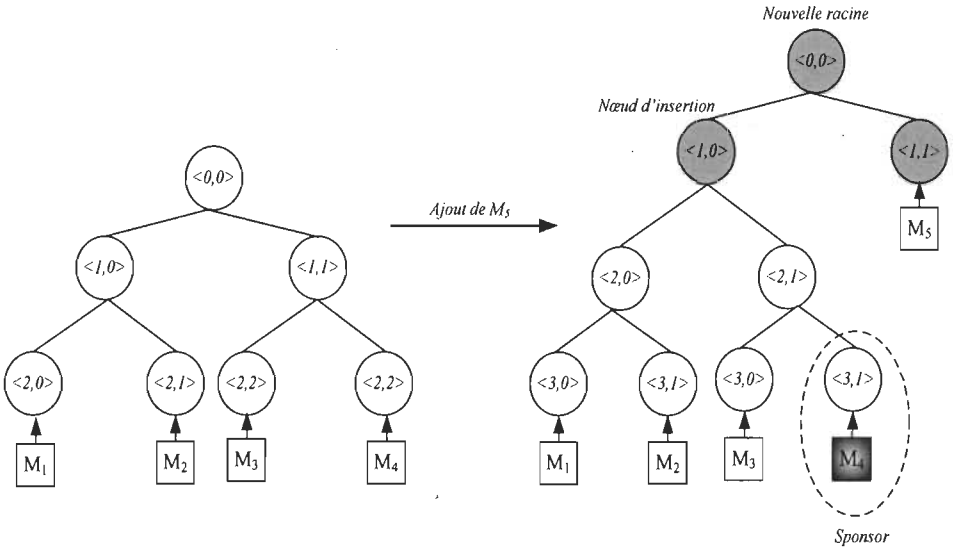


Figure 23 : L'arbre de clés après l'ajout

- Diffusion par le nouveau membre de la demande d'ajout avec la clé publique.
- Détermination du point d'insertion du nouvel arrivant dans l'arbre par les anciens membres.
- Création d'un nœud intermédiaire qui est le nœud le plus à droite du nœud d'insertion.
- Le sponsor calcule les nouvelles clés cachées (vu sa connaissance de toutes les clés publiques nécessaires) et diffuse le nouvel arbre après sa mise à jour.
- Chaque nœud du groupe calcule la clé du groupe après réception du nouvel arbre.

Dans le cas du départ d'un membre ou de son exclusion de l'arbre [45] (figure 24):

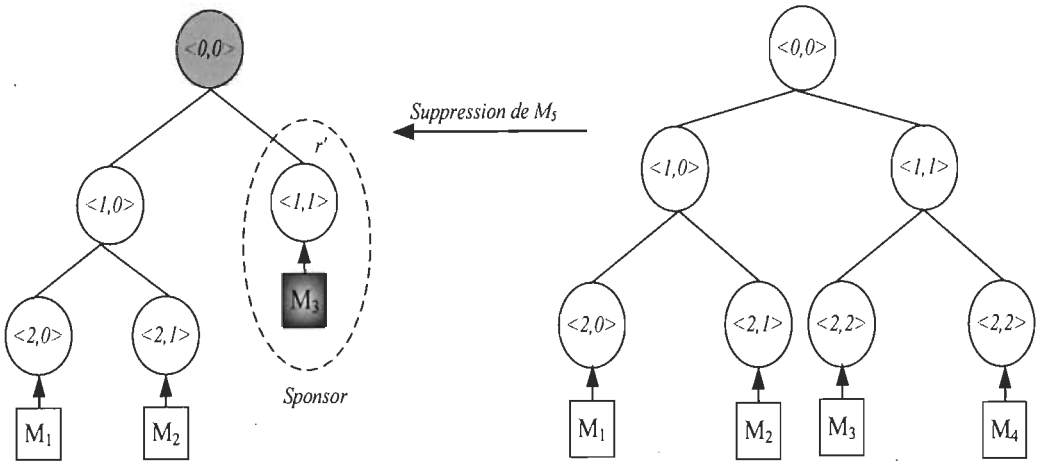


Figure 24 : L'arbre de clés après le départ

- Le nœud partant informe les autres membres de son départ.
- Chaque membre met à jour l'arbre de clé.
- Le nœud frère remplace le nœud père.
- Le sponsor génère une nouvelle clé cachée.
- Toutes les clés sur le chemin en montant à la racine sont recalculées par le sponsor.
- De nouvelles clés publiques sont diffusées.
- Tous les membres calculent (à l'aide des clés publiques) la nouvelle clé du groupe.

Avantages et inconvénients de l'approche TGDH [45]:

Malgré sa force, qui se présente dans le coût logarithmique des échanges entre membres (qui ne nécessitent pas un canal sécurisé) lors des événements, l'approche TGDH

représente un handicap majeur, dans le cas de la défaillance du sponsor qui est le point central du contrôle du groupe.

4.4 Conclusion

Au niveau de ce chapitre, il a été question de faire une étude comparative des différents types d'approches de gestion des clés de groupe existantes.

L'ouverture des réseaux ad hoc sans fil sur des environnements non sûrs, le manque d'infrastructure, le grand dynamisme des nœuds et la limitation des paramètres liés aux dispositifs mobiles (la bande passante, les sources d'alimentation ainsi que leur capacité de stockage) rendent difficile l'opération de gestion des clés du groupe.

Nous avons constaté que les approches de gestion et de distribution des clés sont réparties en trois catégories :

- Les approches *contributives* avec les deux sous catégories; par diffusion ou en arbre de clés centralisées. Dans ce genre d'approche, la gestion des clés est confiée au contrôleur du groupe, ce qui rend le groupe vulnérable aux attaques des intrus.

- Les approches *semi-distribuées* où la gestion des clés de groupe est confiée à plusieurs nœuds qui peuvent ne pas faire partie du groupe, chose qui peut être considérée comme un inconvénient majeur.

- Les approches *distributives* peuvent être classées en deux grandes sous catégories : celles qui sont arborescentes utilisant un arbre binaire de clé et celles non arborescentes. La majorité de ces dernières sont considérées comme des extensions de l'algorithme de Diffie-Hillman.

Néanmoins, les approches contributives en arbre sont plus conviviales dans les communications de groupe au sein des réseaux ad hoc vu le meilleur coût des échanges entre les membres du groupe.

Dans le chapitre 5, nous allons présenter notre approche de gestion de clés pour les communications multicast au sein des réseaux ad hoc. Dans le fond, c'est une approche contributive qui utilise un arbre binaire de clé. Dans cette approche nous avons utilisé la théorie des courbes elliptiques, vu sa fiabilité dans le domaine de la cryptographie.

Chapitre 5 - L'approche GAKAP : *G*roup *A*ctivity *B*ased *k*ey *A*greement *P*rotocol

5.1 Introduction

Parmi les approches abordées dans le chapitre 3, nous avons vu les approches de distribution par accord de clés au niveau des communications du groupe. En particulier, nous nous sommes intéressés à l'approche TGDH qui est considérée comme l'extension directe de l'algorithme de Diffie-Hillman, que nous avons constaté la plus avantageuse.

Malgré la force de TGDH par rapport aux autres approches, au niveau des exponentiations, nous avons constaté qu'il n'est pas compatible avec les réseaux mobiles. Ce cas est surtout attribué aux réseaux ad hoc vu leur grande dynamique, la limite des ressources de leurs équipements (alimentation, espace de stockage,...), ainsi que la liberté des participants du groupe d'entrer ou de quitter le groupe en tout temps. Ce qui cause une grande consommation de la bande passante lors de l'attribution des clés.

Dans ce chapitre, il sera question de présenter une technique de distribution de clés en arbre ayant la particularité d'être complètement équilibré. Cette technique appelée GAKAP [64] est basée sur le fameux algorithme de Diffie-Hillman, avec l'intégration des techniques de cryptage à base des courbes elliptiques. Dans notre cas, cette approche sera appliquée dans un environnement ad hoc, tout en prenant en considération les propriétés fondamentales de ce genre de réseaux (grande mobilité, opérations presque illimitées au sein du groupe, ressources limitées...).

Comme déjà mentionné au début, de ce mémoire, le but de ce travail est d'améliorer la méthode de distribution et de gestion des clés de cryptage GAKAP (*G*roup *A*ctivity *B*ased *k*ey *A*greement *P*rotocol) [64] qui a été développée au sein du laboratoire LAMIA (*La*boratoire de *M*athématiques et *I*nformatique *A*ppliquée) de l'UQTR par l'équipe de recherche sur la sécurité des communications multicast, en utilisant la technique de cryptographie basée sur les courbes elliptiques.

5.2 Les courbes elliptiques au sein de la cryptographie

La cryptographie à base des courbes elliptiques [46] s'est finalement avérée être une construction mathématique intéressante dans un cryptosystème à clé publique bien établi. Ce genre de cryptographie est déjà inclus dans de nombreuses normes. En fait, ECC (*Elliptic Curve Cryptography*) est maintenant une technique usuelle. Elle, il a résisté à de longues séries d'attaques; ce qui la marque comme un cryptosystème mûr et robuste.

La cryptographie par courbe elliptique (ECC) est une approche de cryptographie à clé publique, basée sur la structure algébrique des courbes elliptiques à corps finis. L'utilisation des courbes elliptiques dans la cryptographie a été suggérée la première fois par Vainqueur S. Miller [47] en 1985.

ECC a un certain nombre d'avantages par rapport à d'autres cryptosystèmes à clé publique tels que RSA. En particulier, pour un niveau indiqué de sécurité, la taille des clés cryptographiques et les opérandes impliqués dans le calcul des cryptosystèmes à courbes elliptiques sont normalement beaucoup plus courts que dans d'autres crypto-systèmes. Ceci est dû à la puissance informatique disponible pour la cryptanalyse qui grandit et cette différence devient de plus en plus plus apparente [46]. C'est un état avantageux

particulièrement pour des applications où les ressources telles que la mémoire et/ou la puissance de calcul sont limitées.

5.2.1 Définitions

5.2.1.1 Courbe elliptique

Du point de vue mathématique, une courbe elliptique est une solution réglée à l'équation polynôme bivariante de degré total 3: $f(x,y) = 4x^3 + 27y^2 = 0$ [48]. Les points sur la courbe constituent un groupe commutatif. Le problème du logarithme discret dans les courbes elliptiques rend l'utilisation de ces dernières particulièrement attrayantes dans les applications cryptographiques [48], [22].

5.2.1.2 Corps finis

Dans algèbre abstraite, un corps fini est un champ qui contient, seulement, de façon finie, beaucoup d'éléments. Les corps finis sont utilisés dans les théories des nombres, la géométrie algébrique, les théorèmes de codage et de cryptographie.

Les constantes **a, b** de l'équation de la courbe elliptique sont choisies tels que, $\mathbf{4a^3 + 27b^2 \neq 0}$, p > 3 ou p est un nombre premier, et $a, b \in GF(q)$ [48].

5.2.2 Description mathématique de la cryptographie à courbes elliptiques

ECC est basé sur le problème discret d'un logarithme constitué par les points d'une courbe elliptique à corps fini. La base essentielle de sécurité des cryptosystèmes à courbes elliptiques se fonde sur l'absence (supposée) d'un algorithme subexponentiel pour résoudre les courbes cryptographiques de logarithme discret [46] [49].

En termes de formules mathématiques, les courbes elliptiques utilisées dans la cryptographie sont définies sur un corps fini ***K*** par l'équation de Weierstrassβ suivante [50]:

$$E/_K : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

5.2.2.1 Les opérations des courbes elliptiques

L'ensemble de points *(x, y)*$\in$ *K*K* et le point à l'infini ∂, constituent un groupe abélien ***E(K)*** (avec $\forall P \in E(K), P + \partial = P$) [49].

Si Q = -P (P= (x_1, y_1) et Q = (x_2, y_2) $\in$ $E(K)$), l'inverse de P= (x_1,y_1) est –P=(x_1, -y_1 – a_1x_1 – a_3), alors $P + Q = \partial$. La somme P+Q = (x_3, y_3) est calculée comme suit [49] :

$$x_3 = \gamma^2 + a_1\gamma - a_2 - x_1 - x_2$$

$$y_3 = -(\gamma + a_1)x_3 - \mu - a_3$$

avec:

$$\begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & P \neq Q \\ \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3} & P = Q \end{cases}$$

et : $\mu = y_1 - \gamma x_1$

Nous constatons l'emploi de deux principales familles des courbes elliptiques dans la cryptographie, la définition de la courbe est basée sur l'excédent bas du corps GF. En particulier, pour *(K)* $\neq$ *2,3*, l'équation générale de Weierstrassβ [50] peut être simplifiée comme suit :

$$E/K:\ y^2 = x^3 + ax + b \qquad (avec\ 4a^2 + 27b^2 \not\equiv 0\ mod\ p)$$

Admettant que $a_1 = a_2 = a_3 = 0$, $a_4 = a$ et $a_6 = b$, dans l'équation générale de Weierstrassβ, la somme de $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ ($P \neq Q$) est donnée par $P+Q = (x_3, y_3)$ où : $x_3 = \gamma^2 - x_1 - x_2$ et $y_3 = \gamma(x_1 - x_3) - y_1$, avec $\gamma = \frac{(y_1 - y_2)}{(x_1 + x_2)}$ dans le cas où $P \neq Q$, mais dans le cas où $P = Q$, $\gamma = \frac{(3x_1^2 + a)}{(2y_1)}$.

5.2.2.2 Les courbes elliptiques sur les corps *GF (2^m)*

Pour K=2, l'équation non singulière des courbes elliptiques est donnée par : $E/K: y^2 + xy = x^3 + ax^2 + b$, avec $b \neq 0$, l'ensemble des couples (x_i, y_j) appartenant au produit $GF(2^m) x GF(2^m)$ satisfait l'équation.

La somme des deux points $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ $(P \neq Q)$ est donné par $P+Q = (x_3, y_3)$ où $x_3 = \gamma^2 + \gamma + a + x_1 + x_2$, $y_3 = \gamma(x_1 + x_3) + x_3 - y_1$, avec $\gamma = (y_1 - y_2)/(x_1 - x_2)$.

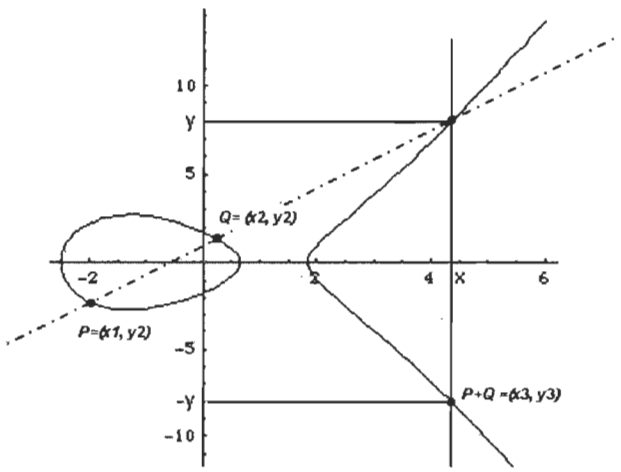


Figure 25: Addition des deux points Pet Q

Par contre, si $P=Q$, il s'agit dans ce cas du doublage des deux points, qui est donné comme suit :

$$P+Q = (x_3, y_3),\ \gamma = x_1 + (y_1/x_2),\ x_3 = \gamma^2 + \gamma + a \text{ et } y_3 = x_1^2 + (\gamma + 1)x^3.$$

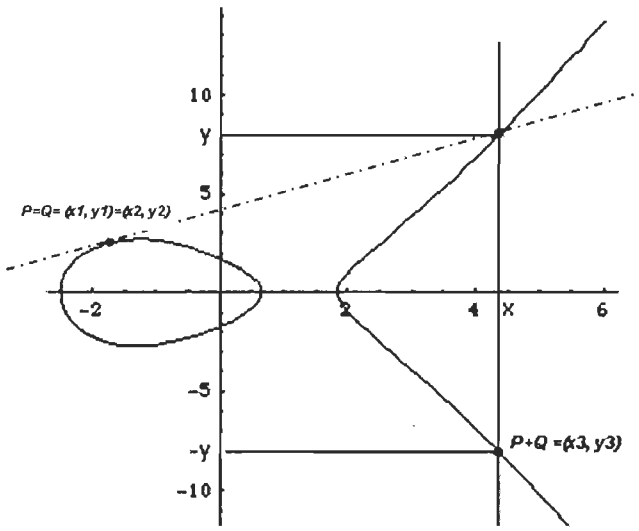


Figure 26: Doublage des deux points Pet Q

5.2.2.3 L'ordre de la courbe

L'ordre de la courbe ***#E/K*** est le nombre de points *Ė/k*. Pour les courbes générales, l'ordre de la courbe peut être calculé dans le polynôme par l'algorithme de Schoof's [51], et plus efficacement par la technique de Satoh's [52] pour des courbes de petites caractéristiques.

Une multiplication scalaire sur la courbe est obtenue par $kP = P+P+P+...+P$ (k étant le nombre de tours), après avoir fixé un point P = (x,y) sur la même courbe.

L'ordre du point P est n, qui est le plus petit nombre entier tel que, $nP = \partial$. Notons que, si i et j sont des nombres entiers, $iP=jP$ si et seulement si $i = j(modn)$.

En fixant un point P de l'ordre n et un point Q, le problème ECDL (*elliptic curve discrete logarithm*) consiste à trouver un nombre entier k, tels que $Q=kP$, le cas échéant. La complexité de résolution du problème ECDL est la base des cryptosystèmes à courbes elliptiques. Il y a plusieurs algorithmes pour résoudre ce problème, tel que l'algorithme de la méthode Pollard-ρ [60], mais, aucun de ces algorithmes n'est subexponentiel pour les courbes convenablement choisies.

Pour les courbes elliptiques générales, il y a des catégories dans lesquelles les algorithmes subexponentiels existent, rendant possibles des attaques spéciales. En particulier, l'attaque de Menezes/Okamato/Vanstone (MOV) [53], l'attaque de Smart/Araki-Satoh/Semaev [54], et l'attaque de Weil [55], toutes fonctionnent avec des hypothèses particulières énoncées sur les courbes attaquées.

Ces résultats mathématiques sont fondamentaux du fait qu'ils ont affecté les choix de conception pour des protocoles et des normes utilisés dans des applications réelles. Par exemple, des courbes super singulières ne sont pas employées en raison du résultat de Menezes, d'Okamato, et de Vanstone [53].

5.2.3 L'algorithme des courbes elliptiques

Dans cette section, nous abordons le problème du calcul arithmétique des courbes elliptiques. L'opération de base, pour n'importe quel cryptosystème à base des courbes elliptiques, est la multiplication scalaire d'un paramètre de sécurité et d'un point d'exécution. La manière la plus simple de la mettre en application est d'écrire le kP [52] comme:

$$kP = \left(\sum_{i=0}^{l-1} k_i 2^i\right) \quad P = (k_{l-1}2^{l-1}P) + \cdots + (k_1 2P) + (k_0 P)$$

$$= 2(2(\dots 2(2(k_{l-1}P) + k_{l-2}P) + \cdots) + k_1P) + k_0P$$

Ainsi que d'employer l'algorithme de la méthode binaire suivant:

P, k = $(K_{l-1}, K_{l-2}, K_{l-3}, \dots, K_0)_2$ est connu, nous devons calculer $Q = kP$.

1. R := ∂
2. pour j :=l-1 jusqu'à 0 faire
3. R := 2R
4. si (k_j =1) alors R :=R+P
5. fin pour
6. retourner R

L'algorithme de la méthode binaire, comme la plupart des méthodes alternatives, ramène la multiplication scalaire aux additions (R : = R+P) et doublages (R :=2R) sur la courbe. Alternativement, ceux-ci sont réduits aux additions, aux multiplications et aux inversions sur les corps finis fondamentaux GF(p) et GF (2^m).

Il est important de noter que x_r et y_r, représentent les coordonnées du point *R* sur la courbe. En employant d'autres représentations, il est possible d'obtenir différentes formules pour des additions et des doublages, tout en utilisant les mêmes opérations arithmétiques de base, telle que, l'addition, la multiplication, l'inversion et la soustraction.

5.2.4 ECDH : Diffie-Hellman à base des courbes elliptiques.

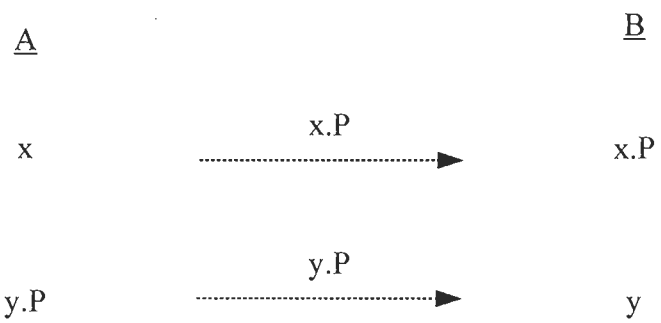
Nous décrivons l'échange des clés par l'approche Diffie-Hellman à base des courbes elliptiques ECDH (*Elliptic curve Diffie-Hellman*) qui utilise un groupe additif des points

d'une courbe elliptique [52], alors que l'échange classique des clés Diffie-Hellman est basé sur un groupe multiplicatif *mod p*.

Il est plus simple et plus facile de comprendre l'arrangement, tout en se concentrant sur le rôle du problème ECDL dans une vraie application. ECDH opère comme suit : [52]

- Deux parties ***A*** et ***B*** se mettent d'accord sur une clé partagée.
- Comme ensemble de paramètres de domaine, ***A*** et ***B*** définissent une courbe elliptique ***E*** spécifique et un point générateur ***P***.
- **A** produit une valeur secrète aléatoire ***x*** et envoie le résultat de la multiplication scalaire ***x.P*** à **B**.
- **B** produit une valeur secrète ***y*** et envoie ***y.P*** à **A.**

Ce qui se résume comme suit [52]:



A et B calculent: $K_A = K_B = x.(y.P) = xy.P$

Les deux participants A et B partagent, de ce fait, la valeur secrète $K_A = K_B$. Le problème de récupération des valeurs $x.P$ et $y.P$ données par $xy.P$ s'appelle ***ECDHP***

(Elliptic Curve Diffie–Hellman Problem). Nous savons que ***ECDHP*** fait partie de ***ECDLP***, dans le sens que si nous pouvons résoudre ECDLP, nous pouvons aussi résoudre ECDHP.

D'autres arrangements cryptographiques fondamentaux incluent l'algorithme de la signature ECDSA, qui prolonge le DSA à l'aide des courbes elliptiques. L'arrangement de chiffrement intégré par les courbes elliptiques (EC-IES) fonctionne fondamentalement comme Diffie–Hellman, mais, avec un chiffrement symétrique. La courbe elliptique de Menezes et Vanstone (l'EC-MQV) [53] fournit le meilleur mécanisme pour un échange authentifié.

De nos jours, de tels arrangements cryptographiques à base des courbes elliptiques ont subi beaucoup d'efforts d'adaptation et d'amélioration. En fait, la cryptographie à courbes elliptiques est particulièrement adaptée, d'un côté, aux réalisations qui manquent de l'interopérabilité, dues au grand nombre de paramètres qui doivent être choisis à différents niveaux et d'un autre côté aux arrangements cryptographiques à niveau élevé. Parmi d'autres éléments à prendre en considération dans ECC, nous avons besoin d'étalonnage dans la définition des formats pour la représentation des éléments de corps, de la courbe, de clés cryptographiques et ainsi de suite. En outre, il est important d'avoir une collection de courbes spéciales recommandées et de corps finis fondamentaux afin de favoriser l'interopérabilité et favoriser les choix techniques particuliers de sécurité et d'efficacité qui sont appropriés.

Parmi les principales normes comprenant la cryptographie à courbes elliptiques, nous pourrons citer les caractéristiques standards IEEE P1363 pour *la cryptographie à clé publique* [56], la norme *des signatures digitales* DSS (***D**igital **S**ignature **S**tandard*) de

NIST [62], la norme ANSI pour l'industrie de services financiers [57] [58] et les normes SECG *SEC1*et *SEC2* [59] [60].

Par exemple, la norme DSS de NIST inclut une collection de courbes elliptiques recommandées, à clé privée. Elle indique comment représenter les éléments du corps et prévoit l'utilisation des courbes aléatoire et en particulier, de Koblitz [61].

5.2.5 La signature digitale DSA

Le **DSA** pour ***Digital Signature Algorithm***, est un algorithme de signature numérique, faisant partie de la norme DSS, qui a été standardisée par le NIST en 1993. L'algorithme DSA se déroule en trois étapes :

1- Génération des clés ;

2- Signature du message ;

3- Vérification du message signé ;

5.2.5.1 Génération des clés

Pour générer la clé, l'exécution de l'algorithme suivant s'effectue :

1- Choisir un entier p;

2- Choisir un entier z;

3- Choisir un nombre premier q, avec $p-1 = qz$;

4- Choisir h avec $1 < h < p-1$ (avec $g = h^z \bmod p > 1$)

5- Générer x aléatoirement avec $0 < x < q$

6- Calculer $y = g^x \bmod p$

7- La clé publique (p, q, g, h) et la clé privée est x

5.2.5.2 La signature du message

La signature du message s'effectue comme suit :

1- Choisir un nombre aléatoire d $1 < d < q$;

2- Calculer $s_1 = (g^s \bmod p) \bmod q$;

3- Calculer $s_2 = (H(m) + s_1 * x)/s$; (avec m le message et H=SHA-1 le résultat du hachage cryptographique sur m)

4- La signature est (s_1, s_2).

5.2.5.3 La vérification du message signé

1- Si $0 < s_1 < q$ ou $0 < s_2 < q$ la signature sera rejetée,

2- Calculer $w = (mod\ q) / (s_2)$,

3- Calculer $u_1 = H(m) * (mod\ q)$,

4- Calculer $u_2 = s_1 * w\ (mod\ q)$,

5- Calculer $v = [g^{u_1} * y^{u_2}\ mod\ p]\ mod\ q$,

6- Si $v = s_1$, la signature est valide.

5.2.6 La signature digitale ECDSA

ECDSA est une variante de DSA. C'est aussi l'algorithme de signature digitale le plus populaire due à sa complexité, voir même, l'impossibilité de résoudre son logarithme qui est discret dans le sous-groupe d'une courbe elliptique. Le chiffrement du message avec ECDSA se fait en suivant les trois étapes suivantes :

- Préparation et génération de la clé ;
- Signature du message ;

- La vérification du message signé.

Soit **G** un élément d'une courbe elliptique d'ordre **n**. **a** et **b**, étant, deux éléments de la courbe et qui font partie d'un corps fini de cardinalité **q**.

5.2.6.1 La préparation et la génération des clés

1- Choisir un entier $s \in [1, n-1]$;

2- Calculer $Q = sG$ à l'aide de l'élément de la courbe elliptique utilisée;

3- **Q** est la clé publique tandis que la clé privée est **s**.

5.2.6.2 La signature du message

1- Choisir aléatoirement $k \in [1, n-1]$ (nombre entier);

2- Calculer $k.G = (x_1, y_1)$;

3- Calculer $r = integer\ (x_1)\ mod\ n$;

4- Si $x_1 \in GF\ (2^k)$ *alors* x_1 une dégradation binaire;

5- Calculer $s = \frac{H(m)+rz}{k}\ mod\ n$ (avec m le message et H=SHA-1 le résultat du hachage cryptographique sur m);

6- Si r=0 ou s=0, recommencer;

7- La paire (r, s) est la signature du message.

5.2.6.3 La vérification du message signé

1- Vérifier si **s** et **r** appartiennent bien à l'intervalle $[1, n-1]$;

2- Vérifier que $r = integer\ (x_1)\ mod\ n$,

$$\text{avec} \quad (x_1, y_1) = \left(\frac{H(m)}{s}\ mod\ n\right) G + \left(\frac{r}{s}\ mod\ n\right) Q;$$

3- Vérifier l'appartenance de $Q \neq (0,0)$ à la courbe elliptique;

4- Vérifier que nQ donne $(0,0)$.

5.3 Choix de la courbe elliptique

Pour arriver au but de notre protocole, qui est celui de la réduction au maximum de la taille des clés de cryptage, et du temps écoulé lors de leur génération et/où leur échange, nous avons choisi l'utilisation de l'algorithme de chiffrement par l'échange Diffie-Hellman à base des courbes elliptiques ECTGDH.

Vu l'efficacité du temps et de l'espace des opérations arithmétiques effectuées au sein du groupe, les courbes elliptiques que nous utilisons sont celles définies sur le corps fini GF (2^m).

5.4 Définitions des éléments

- Initiateur du groupe : un responsable primordial au sein du groupe dont la principale tâche consiste à débuter la session multicast. En effet, il permet de concevoir une liste intégrant les participants du groupe plus précisément une liste de contrôle d'accès ACL ainsi que l'arbre des clés. Ces derniers seront transmis, lors de session multicast, à tous les participants.
- Contrôleur initial: choisi parmi les participants du groupe, le contrôleur initial a pour mission principale d'assurer et de coordonner avec un haut niveau de précision l'établissement de la clé du groupe initial. Il faut noter que ce type de poste peut être assuré aussi bien par l'initiateur du groupe que par un autre participant désigné.
- Contrôleur temporaire : il s'occupera de l'établissement de la structure de l'arbre des clés en cas de tout changement imprévu causé par un ajout, un départ ou bien

une exclusion d'un membre du groupe. Il permet ainsi d'adapter la structure de l'arbre à la nouvelle situation.

- Sponsor : il effectue la distribution des clés pour assurer l'opérationnalisation de l'arbre tout en respectant toute opération liée à un renouvellement de la structure de l'arbre.
- Mise à jour des clés : cette opération à un caractère fondamental dans la mesure où certaines clés peuvent devenir accessibles au public. Dans une telle situation, il serait vital de mettre à jour les clés pour conserver la confidentialité de l'information.
- Échange DH : comporte deux messages entrepris entre deux membres du groupe [41].
- Tour synchrone : au début du tour et en relation avec la procédure d'échange entre participants, le tour synchrone permet l'envoie et la réception de manière aléatoire un nombre de paquets (pas forcement les mêmes) dans un temps précis [62] [1].
- Tour simple : chaque membre aura pour objectif d'assurer soit l'envoi ou la réception de plusieurs paquets durant un tour [62] [1].
- Participant : fait référence à tout membre assisté à la session multicast.
- Activité de groupe : les opérations comportant l'activité du groupe tiennent compte de la vivacité des membres à l'intérieur du groupe multicast et le niveau de leur mobilité. L'adaptation de l'arbre de clés du groupe multicast repose sur la valeur de cette activité.

5.4.1 Notations

Symboles	Description
$\{m\}_k$	Donnée **m** est chiffrée avec la clé k.
$\{M_i\}$	Membre M d'indice i.
BK_i	Clé cachée du membre i.
$\{BK^*_{Si}\}$	Groupe de clés cachées transmises par le sponsor s_i.
$\{M_1..M_n\}$	Groupe de membres.
GK	Clé de groupe

Tableau 4 : Notations

5.4.2 Types de paquets

Nous décrivons dans les pages qui suivent, les divers types de paquets utilisés durant les échanges des clés de communications entre la racine de l'arbre multicast et les participants. Ces paquets sont encapsulés dans un message encodé au sein d'un paquet plus général.

5.4.2.1 Description de chaque paquet

Paquet : GAKAP

Type_proto	*Taille_totale*
SOURCE	
DESTINATION	
DONNÉES	

Fonction : dans le paquet GAKAP, qui est un paquet général où tous les paquets qui circulent durant la session sont encapsulés. Les champs du paquet GAKAP sont :

Type_proto **:** c'est une chaîne de bits qui indique le type du contenu du champ *DONNÉES* et indiquant aussi le type du protocole.

Taille_totale : la taille totale du paquet.

SOURCE : adresse de la source

DESTINATION : adresse de la destination

DONNÉES **:** les autres types de paquets encapsulés dans le paquet général.

Utilisé : à chaque envoi de paquet

Type : unicast ou multicast

Paquet: BKEY

Key_ID	*Taille*	*Key_value*

Fonction : c'est le paquet qui contient la clé cachée. Les champs du paquet BKEY sont :

Key_ID: entier qui représente l'identifiant) de la clé.

Taille : la taille du paquet BKEY.

Key_value: entier désignant la valeur de la clé.

Utilisé : lors de la production d'un des événements de groupe décrit ultérieurement à l'intérieur du champ de données.

Type de message : Unicast ou Multicast

Paquet: BKEYS

Key_ID	*Taille*	*Key_value*
Key_ID	*Taille*	*Key_value*
Key_ID	*Taille*	*Key_value*
...	...	...

Fonction : donne l'ensemble des clés cachées qui doivent être ou déjà modifiées. Les champs du paquet BKEY sont :

Key_ID : entier désignant l'identifiant de la clé.

Taille : la taille du paquet BKEYS.

Key_valeur: entier désignant la valeur de la clé.

Utilisé : lorsque des ajouts, des départs multiples, de fusion ou de séparation se produisent durant la session.

Type de message : Unicast ou Multicast

Paquet : TREE

Nombre de niveaux	*Nombre de noeuds*
BKEYS	

Fonction: contient l'arbre des clés. Les champs du paquet TREE sont :

Nombre de niveaux: entier désignant la profondeur de l'arbre des clés.

Nombre de nœuds: entier permettant la construction de l'arbre des clés tout en désignant la taille totale de ce dernier.

Utilisé: lors de la production d'un ajout, d'une fusion ou d'une séparation.

Type de message : Multicast

Paquet : ADD

Type_mes	*Taille*
DONNÉES	

Fonction : ce paquet sera envoyé par un participant qui veut s'ajouter au groupe, en indiquant une demande d'ajout simple. Les champs du paquet ADD sont :

Type_mes : l'entier qui indique le type du message qui peut être : soit une requête (*add_req*) envoyée lors d'un d'ajout, soit une requête (*add_update*) envoyée lorsqu'une mise à jour est nécessaire à la suite d'un ajout simple ou bien du message qui contient l'arbre des clés envoyé au nouveau membre.

Taille: la taille du paquet d'ajouts.

***DONNEÉS* :** il s'agit d'un des paquets BKEY, BKEYS ou TREE

Utilisé : lors de l'ajout simple d'un participant.

Type de message : unicast ou Multicast

Paquet : ADD_G

Type_mesg	*Taille*
DONNÉES	

Fonction : indique aux participants que le paquet contient un message d'ajout multiple. Les champs du paquet ADD_G sont :

Type_mesg : l'entier qui indique le type du message qui peut être : soit une requête (*add_g_req*) envoyée lors d'un d'ajout, soit une requête (*add_g_update*) envoyée lorsqu'une mise à jour est nécessaire à la suite d'un ajout multiple, soit une requête (*add_g_update_tree*) envoyée par le contrôleur temporaire lors d'une mise à jour ou remplacement de l'arbre de clés.

Taille: la taille du paquet d'ajouts.

DONNEÉS **:** il s'agit d'un des paquets BKEY, BKEYS ou TREE

Utilisé : lors des requêtes d'ajout multiple et lors des différentes mises à jour à la suite d'un ajout multiple.

Type de message : multicast

Paquet : LEAVE

Type_mes	***Taille***
DONNÉES	

Fonction : ce paquet sera envoyé lors d'un départ d'un des participants. Les champs du paquet LEAVE sont :

Type_mes : l'entier qui indique le type du message qui peut être : soit une requête (leave_*req*) envoyée lors du départ d'un participant, soit une requête (*leave_update*) envoyée lorsqu'une mise à jour est nécessaire à la suite d'un départ simple.

Taille : donne la taille du paquet de départ.

DONNÉES : Il peut s'agir d'un des paquets BKEY ou BKEYS

Utilisé : lors des requêtes de départ et lors des mises à jour à la suite du départ d'un membre.

Type de message : Multicast.

Paquet : LEAVE_G

Type_mesg	*taille*
DONNEES	

Fonction : ce paquet sera envoyé aux participants lors d'un départ multiple (plusieurs participants simultanément). Les champs du paquet LEAVE sont :

Type_mesg : l'entier qui indique le type du message qui peut être : soit une requête (*leave_g_req*) envoyée lors d'un départ multiple, soit une requête (*leave_g_update*) envoyée lorsqu'une mise à jour est nécessaire à la suite d'un départ multiple.

Taille : donne la taille du paquet de départ.

DONNÉES : Il peut s'agir d'un des paquets BKEYS ou de la liste des participants sortants.

Utilisé : lors des requêtes de départ multiple et lors des mises à jour à la suite d'un départ multiple.

Type de message : Multicast

Paquet : PARTITION

sponsor_ID	*mesg_type*
TREE	

Fonction : signale aux participants la division du groupe en deux groupes ou plus. Les champs du paquet PARTITION sont :

Sponsor ID : identification envoyée par le sponsor au participant demandeur de la séparation.

Mesg_type : il s'agit d'une requête **part_req** du type message de division.

TREE : paquet TREE contenant le sous-arbre constitué des participants demandeurs de la séparation.

Utilisé : lors de la séparation du groupe multicast.

Type de message : Multicast.

Paquet : MERGE

Sponsor_ID	*mesg_type*
TREE	

Fonction : signale aux participants la fusion de deux ou plusieurs groupes. Les champs du paquet FUSION sont :

Sponsor ID : identification envoyée par le sponsor au participant demandeur de la fusion de son groupe.

Mesg_type : il s'agit d'une requête **merge_req** du type message de fusion.

TREE : paquet TREE contenant le sous arbre constitué des participants demandeurs de la fusion.

Utilisé : lors du partitionnement du groupe multicast.

Type de message : Multicast

Paquet : REFRESH

Sponsor_ID	*No_seq*
BKEYS	

Fonction : indique les nouvelles clés suite au rafraîchissement de la clé de groupe. Les champs du paquet FUSION sont :

Sponsor_ID : indique le sponsor qui a effectué le rafraîchissement de la clé de groupe.

No_seq : indique le numéro de séquence du rafraîchissement.

Bkeys : indique le champ qui contient les clés cachées rafraîchies.

Utilisé : le paquet est utilisé pour indiquer le rafraîchissement de la clé de groupe.

Utilisé : pour signaler le rafraîchissement de la clé de groupe à la suite d'un événement au sein du groupe.

5.4.2.2 Clé glissée

C'est une clé qui devient la clé du nœud père après le départ d'un des deux nœuds frères d'un sous-groupe participant.

5.4.2.3 Clé cachée

C'est une clé publique obtenue à l'aide d'une fonction à sens unique appliquée sur la clé privée du membre.

5.4.2.4 Fonction à sens unique

La fonction à sens unique est une fonction facile à calculer, mais, elle est très difficile à calculer dans le sens inverse. De telles fonctions ont des applications importantes dans la cryptographie, spécifiquement dans la cryptographie à clé publique.

5.4.2.5 Le stockage

Les clés secrètes et l'arbre des clés cachées de tous les nœuds intermédiaires sur le chemin d'un participant jusqu'à la racine sont stockées par ce participant.

5.4.2.6 La clé mise à jour

C'est la nouvelle valeur attribuée à la clé après un événement. Elle n'a aucune relation avec la valeur remplacée.

5.4.2.7 La constante d'activité: α

$$\alpha = \frac{\sum(Ajouts)}{\sum(Depart)}$$

La constante d'activité est le rapport de la somme du nombre des ajouts (simples et multiples) sur la somme du nombre de départ (simples et multiples) produit au sein du groupe.

5.5 Les opérations sur le groupe

Dans les réseaux ad hoc qui ont une dynamique très élevée, les événements d'ajout, de départ, de fusion et de division peuvent arriver très fréquemment. Par conséquent, la gestion des clés peut devenir très laborieuse dans la limite de la consommation de bande passante et du calcul des clés.

Dans ce qui suit, nous allons décrire les différentes opérations du groupe multicast qui se produisent lors de la gestion des clés de cryptage.

5.5.1 Élection du sponsor

L'élection du sponsor, lors d'un événement, est basée sur sa proximité du point de vue condition parentale au membre entrant ou sortant. Si le nœud entrant ou sortant a un frère du même père, le dernier va devenir le sponsor. Autrement dit, s'il existe un nœud voisin du nœud entrant ou sortant, il va devenir le sponsor.

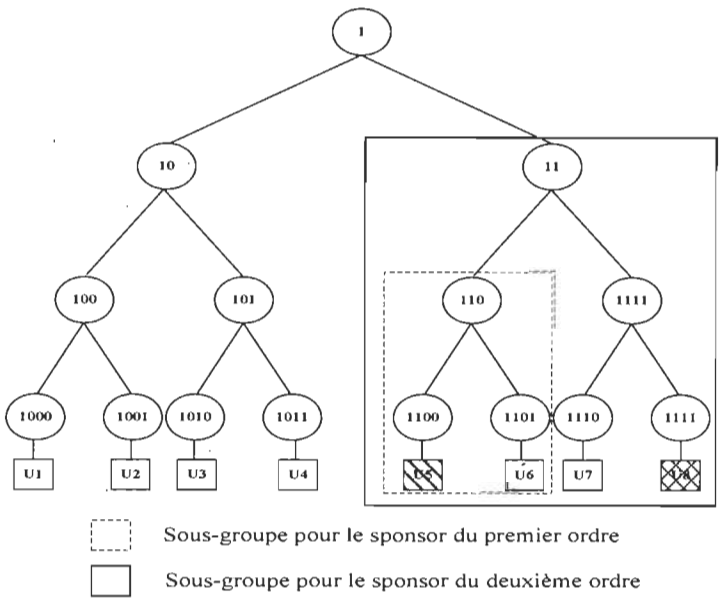


Figure 27: Ordre du choix du sponsor

5.5.2 Initialisation

Dans cette phase, l'arbre principal initial est construit et la clé initiale du groupe est établie. Le protocole fonctionne comme suit:

- **Étape 1** : l'initiateur du groupe (ou le contrôleur initial) édite l'ouverture d'une session multicast. Il accumule la liste des participants, construit l'arbre contenant les clés cachées et leurs noms (ou identification) et les diffuse aux membres.
- **Étape 2** :

1. Pour chaque tour i, i∈ [1, h] (avec h= $\log_2 n$)
    2. Presque $n/2^i$ des membres du groupe vont devenir des sponsors;
    3. Chacun des sponsors calcule les clés et les clés cachées sur son chemin jusqu'à la racine;
    4. Diffusion des clés cachées à tout le groupe;
    5. Fin pour.

- **Étape 3** : Au tour h+1: Chaque membre calcule la constante initiale d'activité du groupe ainsi que la clé du groupe.

À la fin de la phase d'initialisation, tous les membres vont accomplir le même arbre et vont calculer la même clé du groupe. La nomination des clés se fait durant cette phase. Les membres peuvent déterminer l'ordre de la condition parentale qui existe parmi eux. Cette condition parentale servira plus tard à élire le sponsor.

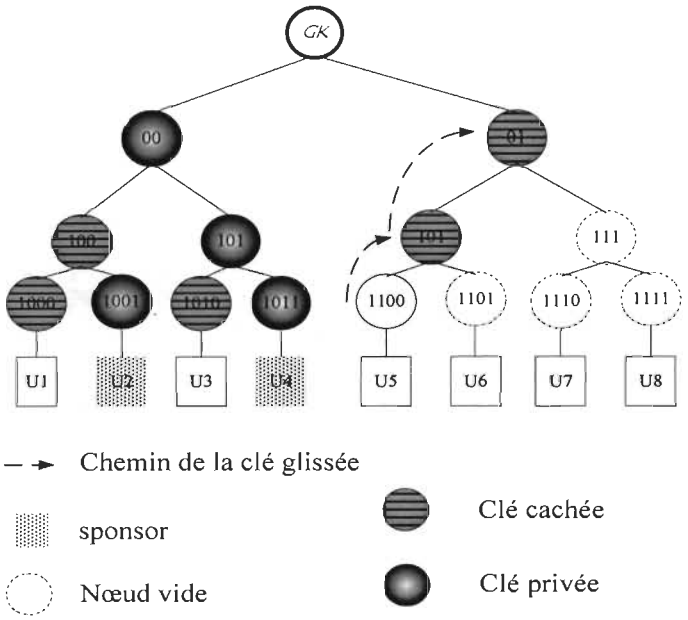


Figure 28: Phase d'initialisation de l'arbre des clés pour GAKAP

5.5.3 Les événements au sein du groupe

5.5.3.1 L'ajout simple

Le nouveau membre déclare son désire de se joindre au groupe en envoyant une demande contenant sa clé cachée (bkey) BK. Chacun des anciens membres du groupe détermine le point d'insertion qui est le premier nœud à feuille vide en traversant l'arbre principal de gauche à droite. Le nœud fils du même père du nouveau membre (s'il existe), devient le sponsor, et en même temps un des nœuds cousin devient aussi sponsor.

Le sponsor met à jour son arbre principal en calculant toutes les clés et les clés cachées sur son chemin jusqu'à la racine et envoie les clés cachées modifiées aux autres membres ainsi que l'arbre des clés en mode unicast au nouveau membre.

Chaque membre détermine le point d'insertion, calcule la nouvelle clé et la constante d'activité. Si, après l'addition d'un nouveau membre, tous les nœuds de l'arbre sont remplis, un nouvel arbre doit être construit selon les étapes suivantes :

- Un nouveau nœud de clé de groupe est construit sur le dessus de l'arbre;
- L'arbre principal précédent devient le sous-arbre gauche du nouvel arbre;
- Le sous-arbre droit est construit selon la constante d'activité.

Lors de sa réception du nouvel arbre principal du sponsor, le nouveau membre calcule les clés manquantes de tous les parents sur son chemin jusqu'à la racine.

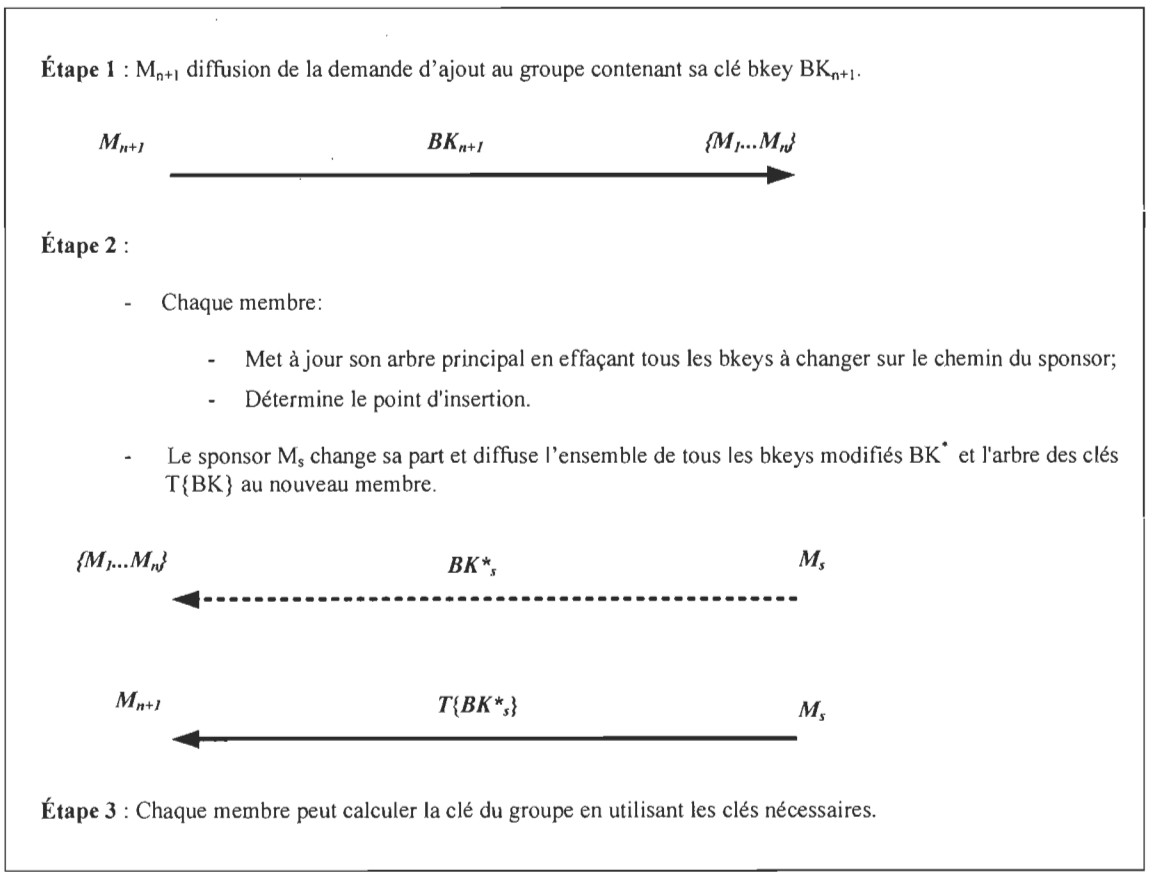


Figure 29: Événement d'ajout simple dans l'approche GAKAP [63]

5.5.3.2 Départ simple

Lors du départ d'un seul membre, ce dernier envoie une requête du départ qui contient son identité ID.

Le sponsor produit une nouvelle contribution, met à jour son arbre principal en effaçant toutes les clés et les clés cachées sur son chemin jusqu'à la racine et annonce les clés cachées modifiées à tous les autres membres.

Chaque membre met à jour l'arbre principal en déterminant et en supprimant les clés à changer, et en les remplaçant par celles contenues dans la liste reçue du sponsor.

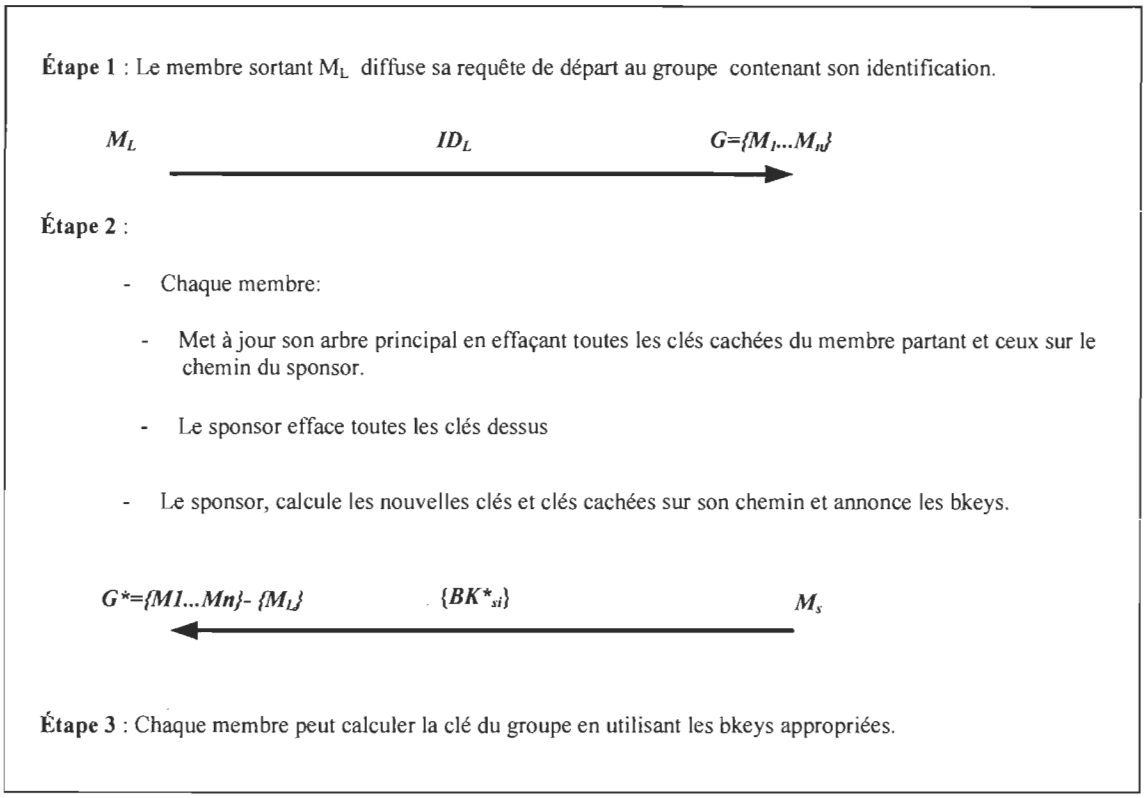


Figure 30: Événement du départ simple dans l'approche GAKAP [63]

5.5.3.3 Ajouts multiples

Comme dans l'événement de l'ajout simple, les membres qui désirent se joindre au groupe envoient leur demande en même temps (ou séparément). Puisque toutes les demandes d'ajout ne sont pas reçues dans le même ordre par tous les membres, il est nécessaire de les classifier afin de garantir la même structure principale. Alors, c'est dans ce but qu'un contrôleur temporaire est élu. Le contrôleur temporaire du groupe est le membre d'extrême droite concerné par l'événement d'addition. Le contrôleur provisoire détermine les divers points d'insertion et insère chaque nouveau participant dans un nœud vide.

Le contrôleur provisoire choisit une nouvelle contribution, calcule les nouvelles clés et les bkeys sur son chemin jusqu'à la racine et annonce l'arbre contenant le bkey et la constante d'activité à tous les participants. Chacun des anciens membres met à jour son arbre principal en effaçant toutes les clés et les bkeys à changer. Les nouveaux membres reçoivent la nouvelle structure arborescente principale. Chaque sponsor diffuse les clés cachées modifiées à tous les autres membres. Chaque membre remplace toutes les bkeys modifiées par celles reçues des sponsors pour pouvoir ensuite calculer la clé du groupe. Le nouvel arbre principal peut être construit comme indiqué dans l'événement d'ajout simple si à un moment dans le processus l'arbre est plein. Si deux nouveaux membres sont des feuilles d'un nœud vide intermédiaire, l'un d'entre eux (l'extrême gauche) devient le sponsor.

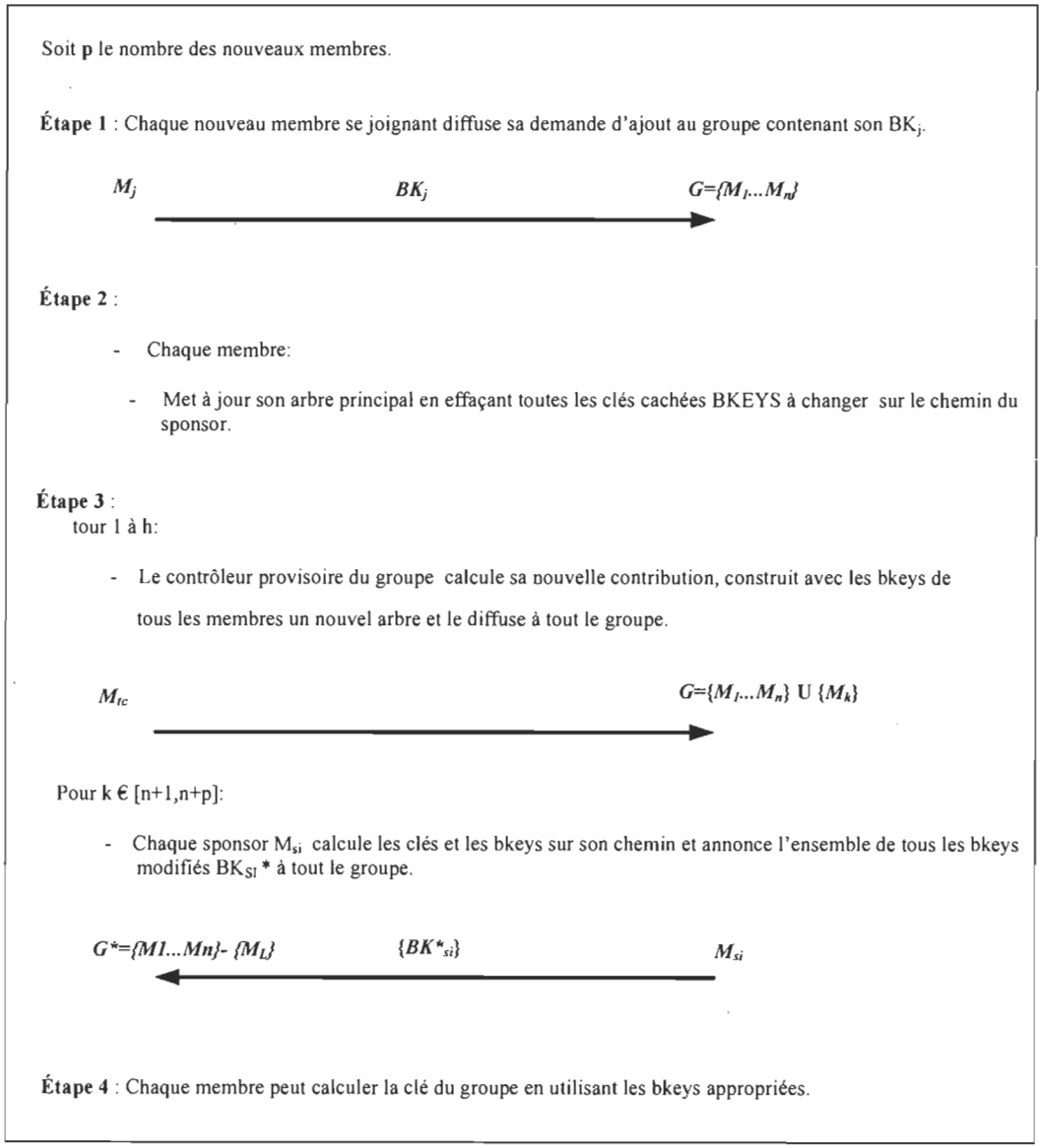


Figure 31: Événement des ajouts multiples dans l'approche GAKAP [63]

5.5.3.4 Départs multiples

Comme dans l'événement du départ simple, beaucoup de membres quittent le groupe en même temps en envoyant d'une manière individuelle la requête du départ.

Le sponsor identifie tous les participants partants. Il change sa contribution, génère une nouvelle clé et calcule les clés secrètes et cachées des nœuds parents sur son chemin

jusqu'à la racine. Il annonce les clefs cachées de ses parents. Il remplace toutes les autres clés cachées modifiées. Il calcule la nouvelle clé du groupe.

Chacun des membres identifie les participants partants. Il efface les clés des membres partants et des nœuds parents si c'est nécessaire. Il met à jour l'arbre en remplaçant toutes les clés cachées qui ont été modifiées par les sponsors. Il calcule la nouvelle clé de groupe.

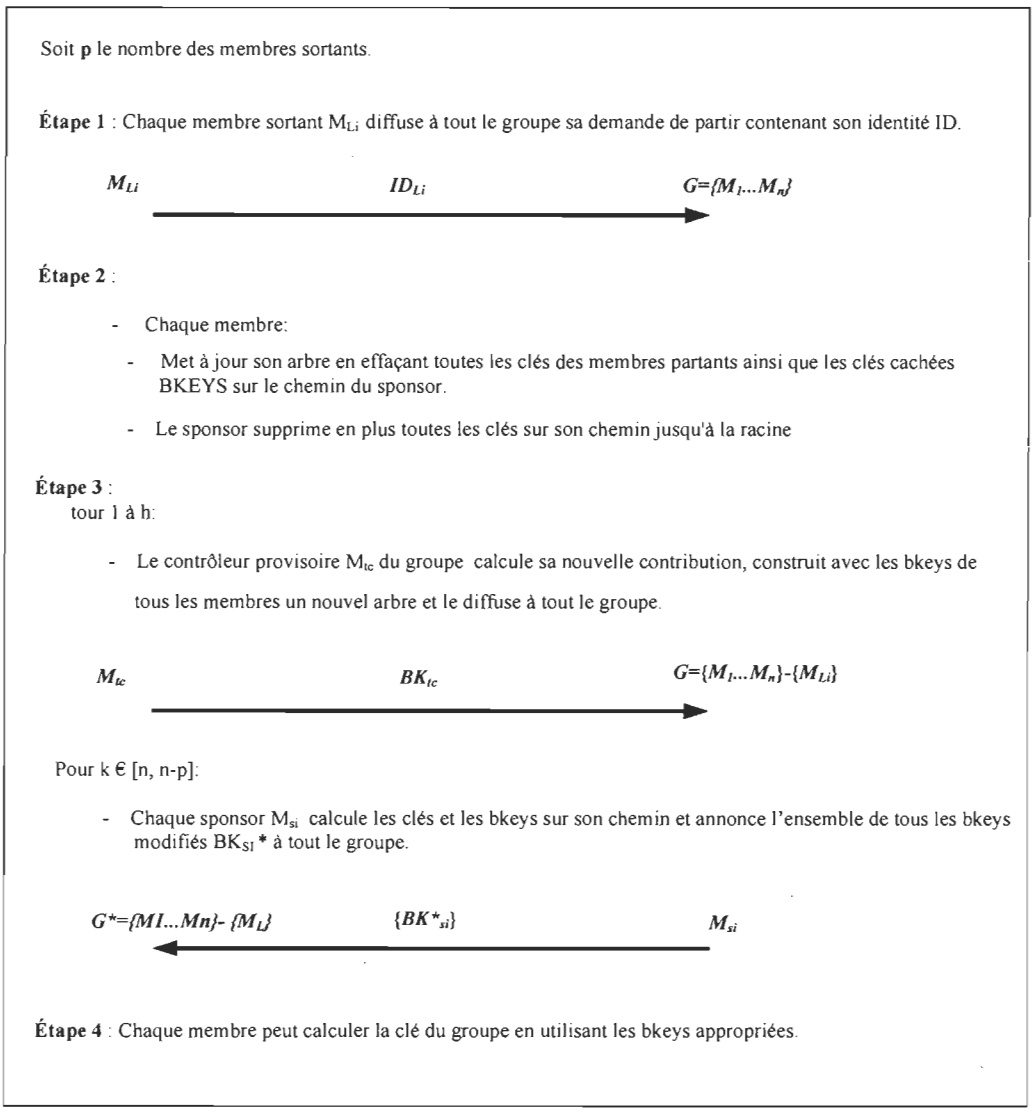


Figure 32: Événement des départs multiples dans l'approche GAKAP [63]

5.5.3.5 Fusion des nœuds

Chaque sponsor choisit une nouvelle contribution et calcule toutes les clés secrètes et cachées sur son chemin jusqu'à la racine. Il annonce son sous-groupe contenant les clés cachées. Après la réception du nouveau sous-groupe, il construit un nouvel arbre des deux anciens arbres dans lequel le plus profond devient le sous-groupe de gauche. Selon le seuil d'activité, il construit un arbre entièrement équilibré en accomplissant l'arbre secondaire droit et calcule la clé du groupe.

Chaque membre détermine le point d'insertion en fonction du seuil d'activité. Il construit le nouvel arbre principal et calcule, à la fin, la clé du groupe.

5.6 Conclusion

Au terme de ce chapitre, la méthode GAKAP était au centre d'intérêt de notre présentation. En effet, cette méthode s'inscrit dans une nouvelle optique de gestion et de distribution des clés de cryptage au sein d'un groupe multicast basée sur la méthode TGDH. Le but de notre méthode GAKAP contribuera à la diminution de l'exploitation des ressources (bande passante, alimentation, espace de stockage,...) lors de la combinaison des clés du groupe.

L'adoption de la constante d'activité dans notre méthode permet de limiter les méfaits d'un déséquilibre de l'arbre des clés causé par certains événements.

Si nous parvenons à intégrer les courbes elliptiques pour effectuer les combinaisons ainsi que la gestion des clés, nous pouvons contribuer, par la suite, à la limitation de la consommation de la bande passante tout en minimisant la taille des messages échangés. Ce

qui peut influencer sur le coût de calcul des clés, voire l'exploitation des éléments du calcul tel que la CPU.

Chapitre 6 - Les étapes de simulation

6.1 Introduction

En raison de la complexité du réseau ad hoc, les simulations jouent un rôle essentiel dans le développement de ce genre de réseaux, en essayant de caractériser le comportement des nœuds et les effets possibles causés par les événements d'ajout et de départ des membres vu leur liberté au sein du réseau.

Mon application consiste à simuler la mise en œuvre de mon approche GAKAP qui est une méthode de gestion et de distribution des clés de groupe au sein du réseau sans fil ad hoc.

J'ai monté mes simulations sous le simulateur réseau NS (*Network Simulator*) qui offre la possibilité de créer des environnements personnalisés (selon les besoins) et complexes à concevoir dans la réalité. Pour arriver à intégrer la technique des courbes elliptiques malgré la complexité de leur programmation, j'ai utilisé les bibliothèques des courbes elliptiques déjà programmées d'Openssl en les adaptant pour qu'elles soient applicables dans ma technique.

6.2 Les outils utilisés

6.2.1 Les librairies Openssl

Comme déjà mentionné, le grand inconvénient des courbes elliptiques réside dans leur programmation. Pour cette raison, j'ai dû faire appel aux librairies déjà préprogrammées de Openssl.

Openssl est une implémentation open-source des protocoles SSL et TLS utilisés dans les outils cryptographiques. Les librairies Openssl qui sont développées entièrement en langage C/C++, peuvent être employées sur une grande variété de plateformes matérielles et langages informatiques. OpenSSL est basé sur la bibliothèque *SSLeay* développée par *Eric A. Young* et *Tim J. Hudson*. Le kit d'OpenSSL est autorisé sous un permis d'*Apache-Style* qui offre la gratuité et la liberté d'obtenir et d'utiliser ces librairies à des fins commerciales et noncommerciales.

6.2.2 Les simulateurs réseau

J'ai trouvé deux grands simulateurs largement utilisés par les chercheurs en réseautique : NS (Network Simulator) et OPNET. Mon choix est porté sur le NS-2 en raison de sa compatibilité avec le système d'exploitation Linux (fédora 7). Celui-ci est nécessaire pour que les librairies Openssl puissent être installées.

6.2.2.1 NS-2 (*Network Simulator*)

NS est un simulateur discret dédié à la recherche dans le domaine de la gestion des réseaux. Il fournit l'appui substantiel pour les simulations du TCP et des protocoles de routage (multicast par exemple) au sein des réseaux filaires et sans fil.

NS est un simulateur orienté objet, écris en C++, Il utilise l'OTCL comme compilateur. Ce simulateur comprend une hiérarchie de classe en C++ (appelée la hiérarchie compilée), et une hiérarchie semblable de classe en OTCL (également appelée la hiérarchie interprétée). Les deux hiérarchies sont étroitement liées entre elles pour assurer une correspondance linéaire entre leurs classes. La racine de cette structure est la classe TclObject.

6.2.2.2 NS-2 dans l'application

L'utilisation du NS-2 dans notre projet pour simuler mon approche GAKAP se fait en quatre étapes :

- Décortiquer la méthode en TCL seulement;
- Ajouter du code C++ (librairie Openssl) au code source préétabli de NS-2 pour l'adapter à l'environnement;
- Exécuter les simulations en faisant appel au NAM qui est l'outil graphique intégré, utilisé pour visualiser les applications;
- Analyser les résultats générés.

Malgré que NS soit un fort simulateur réseau, en plus de sa robustesse, il possède plusieurs inconvénients, par exemple, l'absence des ressources à jour qui aident à mieux l'utiliser surtout dans le domaine du sans fil (ils sont récemment intégrés), le manque de stabilité au niveau du code ce qui amène à générer des erreurs incompréhensibles, la non-disponibilité des outils complémentaire aidant dans l'interprétation des résultats des simulations.

6.3 Les simulations

Tout d'abord et avant tout, j'ai commencé par l'adaptation des librairies Openssl (C++) afin qu'elles soient compatibles avec mon application.

Pour le bon fonctionnement de mes simulations et pour arriver à les monter sur le simulateur NS-2, j'ai procédé comme suit :

1. La première étape était la compréhension et l'assimilation du simulateur NS-2.
2. Développer le programme en TCL sur le simulateur NS-2.

3. Faire appel au code C++ des librairies Openssl dans le programme TCL.

Remarque

La troisième étape nécessite des modifications au niveau du code source du noyau du simulateur (la classe *TclObject*) en créant des objets de commande. La création de ces derniers nécessite un travail de programmation qui reste une tâche très difficile et longue à compléter, en raison du manque quasitotal de la documentation traitant ce problème et qui aide à gérer les paramètres d'environnement.

6.3.1 Scénario et métriques

6.3.1.1 Scénario

Pour assurer le bon déroulement de mes simulations, nous avons fixé (les membres de l'équipe de recherche sur la sécurité des communications multicast au sein du laboratoire LAMIA et moi) un scénario de paramètres comme suit:

Paramètres	Valeurs
Superficie de la zone	1000 m x 1000m
Nombre de nœuds	50 à 250
Déploiement de nœuds	Uniforme
Nombre de nœuds émetteurs	1
Nombre de nœuds récepteurs	49
Protocole MAC	802.11
Taux de transfert	2Mbps
Propagation du signal radio	FreeSpace
Type d'antenne	Omnidirectionnelle
Modèle de mobilité	RWP (*Random Waypoint*)[1]
Vitesse minimale	2 m/s (minimale non nulle) pause =10s

Programme TCL

C'est la partie de définition des variables

```
# Define options

set val(chan)         Channel/WirelessChannel
set val(prop)         Propagation/TwoRayGround
set val(netif)        Phy/WirelessPhy
set val(mac)          Mac/802_11
set val(ifq)          Queue/DropTail/PriQueue
set val(ll)           LL
set val(ant)          Antenna/OmniAntenna
set val(x)              1000    ;# X dimension of the topography
set val(y)              1000    ;# Y dimension of the topography
set val(ifqlen)         50      ;# max packet in ifq
set val(seed)           0.0
set val(adhocRouting)   DSR
set val(nn)             51          ;# how many nodes are
simulated
set val(stop)           800.0       ;# simulation time
```

La partie de l'initialisation de l'environnement

[1] RWP est un modèle synthétique généralement utilisé pour la mobilité et surtout dans les réseaux sans fils. C'est un modèle élémentaire qui décrit le mouvement aléatoire des nœuds avec de simples limites

```
# create trace object for ns and nam

set tracefd     [open wireless1-out.tr w]
set namtrace    [open wireless1-out.nam w]
#set f0 [open out0.tr w]
#set f1 [open out1.tr w]
#set f2 [open out2.tr w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
```

6.3.1.2 Métriques

Pour arriver au but visé par les simulations, soit l'étude des performances de l'approche GAKAP, j'ai centralisé les paramètres à observer en trois métriques : le nombre de tours effectués afin d'arriver à calculer la clé de groupe, la valeur maximale de la constante d'activité et le nombre de paquets échangés pendant la génération de la clé de cryptage.

6.3.2 Les étapes d'une simulation réseau

Le déroulement des simulations passe par :

- La fixation et la définition de la problématique à étudier;
- La modélisation du problème;
- La mise en scène de la simulation;
- La collecte et l'analyse des résultats obtenus;
- L'interprétation des résultats atteints et la prise de décision.

6.3.3 Implémentation du programme de simulation

6.3.3.1 L'implémentation et les problèmes rencontrés

L'implémentation de ma méthode GAKAP s'est déroulée en 4 étapes :

- L'adaptation des codes des librairies de Openssl;

- Le développement de l'environnement de simulation et la programmation du protocole en TCL;
- L'intégration du code C/C++ et les librairies Openssl adaptées dans le simulateur NS-2.

Par contre, les difficultés rencontrées durant l'implémentation des simulations ont été au niveau de l'intégration des bibliothèques adaptées de Openssl dans NS-2. Malgré les modifications effectuées au niveau de quelques parties du code source du simulateur, et aussi faute de la non-accessibilité au noyau (la classe *TclObject*), je ne suis pas arrivé à faire appel à toutes les librairies Openssl.

6.3.3.2 Visualisation des simulations

Pour visualiser les simulations sous NS-2, ce dernier fait appel à un outil intégré qui s'installe séparément appelé **NAM** (*Network Animation Monitor*).

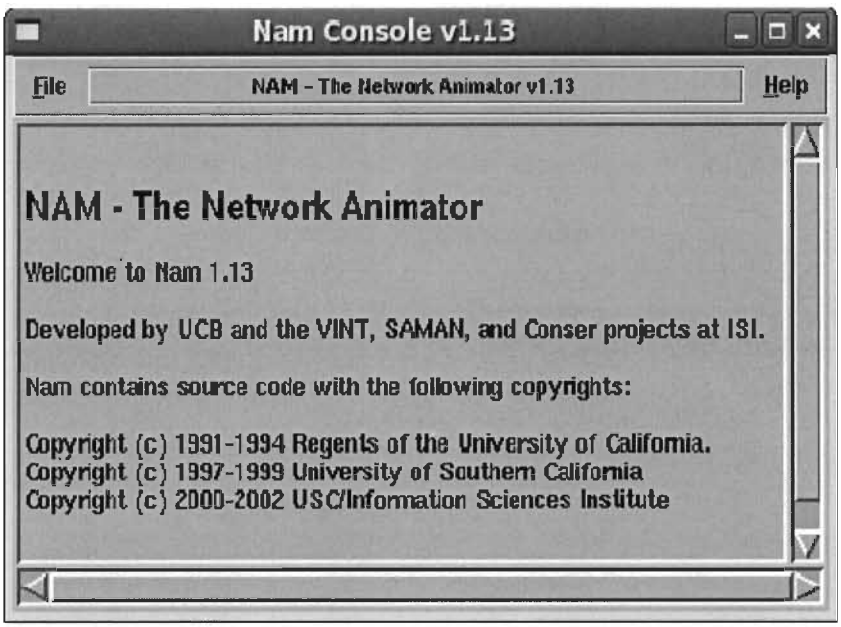


Figure 33: NAM Outil de visualisation de NS-2

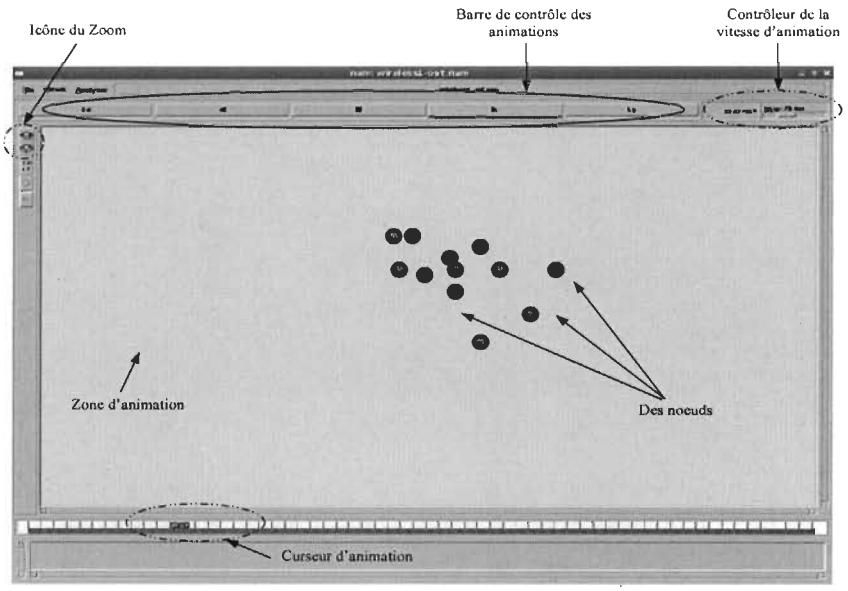


Figure 34: Explication du NAM

6.3.3.3 Événement au niveau de l'arbre

Le développement de cette partie de l'application s'est fait au niveau des librairies Openssl tout en effectuant des adaptations et des modifications au niveau des paramètres.

6.3.3.3.1 L'élection du sponsor

Le sponsor est élu selon sa proximité au point d'insertion ou du départ, du point de vue condition parentale.

Programme Openssl C++

```
GPacket elect_spons(int noPart,int protocole)
{

      GPacket P ,t ,pktM ,pktU,accept,packet ;
      participant sponsor,spons;
      char part[MAX_SIZE];

      part = new char;
      packet.data_ = 0;
      packet.p_type = 0;
      packet.receiver_name = "";
      packet.sender_name = "";
```

```
        itoa(noPart,part,10);
        P=init_packet(packet);
        t=init_packet(packet);
        pktM=init_packet(packet);
        pktU=init_packet(packet);
        accept=init_packet(packet);

        spons.bkey = 0;
        spons.key = 0;
        spons.level = -2;
        spons.name = "";
        spons.no = -1;
        spons.start = -1;
        spons.stop = -1;

        sponsor = init_participant(spons);
if(protocole == 1)
        {

              t =  packet_type(P,part,G_NAME,P_BKEY,JOIN);
              SendM(t);

              //Génération et calcul de clés
              BIGNUM key = compute_key(protocole);
              BIGNUM bkey = compute_bkey(protocole);

              accept = packet_type(P,part,G_NAME,JOIN, P_TREE);
              sponsor.name = "aaaaaa";
       pktU=SendUnicastPacket(accept,sponsor.name,part,P_TREE);

              pktM = SendMcastPacket
              (accept,sponsor.name,G_NAME,P_ADD_UPDATE);

              if(!nb_free_nodes(nbrParticipants))
              {
              tot_num_of_nodes = nb_tot_noeuds
              (2*nbrParticipants,protocole);
              free_nodes = 2*nbrParticipants-1;
              }
              else

              free_nodes--;
              nbrParticipants++;
              nbrJoins++;
              nbPkJoin++;
              nbPkUnicast++;
              nbPkMulticast++;
              nbRndAj++;
              nbTotRnd++;
              nb_key_computed += (profondeur - 1);
              size_of_packet += (profondeur-1) * sizeof(BIGNUM);

              if(nbrJoins != 0)
```

```
                    constante =
((double)nbrJoins/(double)nbrLeave);
                else
                constante = -1;

        }
        else if(protocole == 2)
        {

        t =  packet_type(P,part,G_NAME,P_BKEY,JOIN);
        SendM(t);

                //Génération et calcul de clés
                BIGNUM key = compute_key(protocole);
                BIGNUM bkey = compute_bkey(protocole);


                accept = packet_type(P,part,G_NAME,P_BKEY, P_TREE);
                accept = packet_type(P,part,G_NAME,JOIN, P_TREE);
                sponsor.name = "bbbbbb";


pktM = SendMcastPacket(accept,sponsor.name,G_NAME,P_ADD_UPDATE);

                if(!nb_free_nodes(nbrParticipants))
                {
                int tmp = tot_num_of_nodes;
                tot_num_of_nodes = nb_tot_noeuds
                (nbrParticipants,protocole)+2;
                free_nodes = tot_num_of_nodes - tmp;
                }
                else
                free_nodes--;
                nbrParticipants++;
                nbrJoins++;
                //nbPkUnicast++;
                nbPkJoin++;
                nbPkMulticast++;
                nbRndAj++;
                nbTotRnd++;
                nb_key_computed += (profondeur - 1);
                size_of_packet += (tot_num_of_nodes-1) *
sizeof(BIGNUM);

        }
        return t;
}
```

6.3.3.3.2 L'initialisation

Au début de cette opération, le contrôleur du groupe édite le début de la session multicast en diffusant la liste initiale des participants et selon les retours de la diffusion de la part des participants connectés il construit l'arbre du groupe.

L'initialisation se fait surtout au niveau des listes `leaving_list` et `disconnect_list`.

Programme Openssl C++

```
rp = init_packet(packet);

      for(int k=0; k < nbrParticipants; k++)
      {
            //sprintf(liste[k].name,"%d",k);
            liste[k].no = k;
            liste[k].level = profondeur;
            liste[k].key = 0;
            liste[k].bkey = 0;
            liste[k].start = -1;
            liste[k].stop = -1;
      }

// Ici on initialise les différentes listes définies plus haut:
liste, leaving_list,disconnect_list

      for( k= nbrParticipants;k<MAX_USERS;k++)
            {
                  //liste[k].name = "";
                  liste[k].no =  -1;
                  liste[k].level = -2;
                  liste[k].key = 0;
                  liste[k].bkey = 0;
                  liste[k].start = -1;
                  liste[k].stop = -1;
            }
            for(int j=0;j<MAX_USERS;j++)
            {
                  leaving_list[j].no =  -1;
                  leaving_list[j].level = -2;
                  leaving_list[j].key = 0;
                  leaving_list[j].bkey = 0;
                  leaving_list[j].start = -1;
                  leaving_list[j].stop = -1;
            }
```

```
        for(int c=0;c<MAX_USERS;c++)
        {
              //disconnect_list[c].name = "";
              disconnect_list[c].no =  -1;
              disconnect_list[c].level = -2;
              disconnect_list[c].key = 0;
              disconnect_list[c].bkey = 0;
              disconnect_list[c].start = -1;
              disconnect_list[c].stop = -1;
        }
        srand( (unsigned int)time( NULL ) );
        for(int temps=0;temps<TEMPS_SIMUL;temps++)
        {
              if(temps%INTERVAL == 0)
              {
                    for( j=0; j < nbofevents; j++)
                    {
event = rand_event_generator(NB_EVENT+1);
        if(event == 0)
                    {

                    int new_memb;
                    new_memb = nbrParticipants +1;
                    sprintf(index,"%d",new_memb);

                    liste[new_memb].name = index;
                    liste[new_memb].no = new_memb;
        rp = AddPart(liste[new_memb].no,protocole);
                    nb_participants++;
                          }
if(event == 1)                                {
        int nb_arrivant, nb_nouveau;
        char tabindex[MAX_SIZE];

        if(( nb_nouveau = rand()%5)>1
        nb_arrivant = nb_nouveau;


          int new_member;
          for(i = 0;i<nb_arrivant;i++)
                                {

          new_member =  nbrParticipants +1;

sprintf(tabindex,"%d",new_member);
liste[new_member].name = tabindex;
liste[new_member].no = new_member;
rp = AddPart(liste[new_member].no,protocole);
        nb_participants++;
              }
                    }
if(event == 2)
              {
```

```
            sprintf(index,"%d",rand()%nbrParticipants);
                  int trouve;
                  int indice = 0;
                        while(indice < nbrParticipants)
                                    {
if(event == 3)
            {
int leaving_num = nbrParticipants;

//ici on initialise les variables de la structure liste

liste[leaving_num].name = "gone";
liste[leaving_num].no = -1;
liste[leaving_num].bkey = 0;
liste[leaving_num].key = 0;
liste[leaving_num].level = -2;
liste[leaving_num].start = -1;
liste[leaving_num].stop = -1;
rp = DelPart(leaving_num,protocole);
nb_participants--;
for(int val=0;val<nbrParticipants;val++)
      {
      if(leaving_list[val].name == "")
      leaving_list[val].name = index;
      leaving_list[val].no = leaving_num;
                                    }
                              }
                        }
if(event == 4)
            {
int nb_sortant, nb_leaving;
char tabindex[MAX_SIZE];
int *tab_index;
if(( nb_leaving = rand()%5)>1)
nb_sortant = nb_leaving;

tab_index = new int[nb_sortant];
for(i = 0;i<nb_sortant;i++)
      {
int leaving_num;
leaving_num = nbrParticipants;

      sprintf(tabindex,"%d",leaving_num);
      tab_index[i] = leaving_num;
      liste[leaving_num].name = "gone";
      liste[leaving_num].no = -1;
      liste[leaving_num].bkey = 0;
      liste[leaving_num].key = 0;
      liste[leaving_num].level = -2;
      liste[leaving_num].start = -1;
      liste[leaving_num].stop = -1;
 rp = DelParts(leaving_num,nb_sortant,protocole);
rp = DelPart(leaving_num,protocole);
nb_participants--;
```

```
nbRndDepM++;

for(int val=0;val<nbrParticipants;val++)
                                        {
      if(leaving_list[val].name == "")
                                        {
      leaving_list[val].name = tabindex;
      leaving_list[val].no = leaving_num;
                                        }
                                    }
                                }
                            }
if(event == 5
                            {
int discon_num;
discon_num = nbrParticipants;
sprintf(index,"%d",discon_num);

      liste[discon_num].name = "disconnected";
      liste[discon_num].no = -1;
      liste[discon_num].bkey = 0;
      liste[discon_num].key = 0;
      liste[discon_num].level = -2;
      liste[discon_num].start = -1;
      liste[discon_num].stop = -1;

      rp = DisconnectPart(discon_num,protocole);
      nb_participants--;
      for(int val=0;val<nbrParticipants;val++)
                                    {
      if(disconnect_list[val].name == "")
                                    {
      disconnect_list[val].name = index;
      disconnect_list[val].no = (int)index;
                                    }
                                }
                            }
                        }
      AfficherStats(protocole);
      strcpy(NomFichier, "c:\\results2.txt");
      WriteStats(protocole);
                }
          }
    }
```

6.3.3.4 Événements au niveau des nœuds

Les deux figures qui suivent présentent les événements produits au sein du réseau lors de la simulation. Vu la similitude (du côté visualisation) entre l'ajout simple et multiple,

ainsi qu'entre le départ simple et multiple, nous avons dû insérer qu'une seule capture d'écran pour chaque type d'événements.

6.3.3.4.1 L'ajout

Dans cette partie, je traite l'opération de l'ajout du nœud 50, ce dernier envoie la demande de s'ajouter au groupe avec son identité *$numN*, un point d'insersion sera determiné par une contribution de tous les membres du groupe `"valut du noeud $"`, une fois que le point sera fixé, un agent d'attache sera généré pour confirmer l'ajout `$ns_ attach-agent $node.`

Programme TCL

```
#puts "Ajout du noeud 50"

set numN 49

#demande du noeud 49 pour ajouter un agent pour le noeud 50

      set i 0
      while {$numN > 0}
       {
        puts "valut du noeud $numN";
        set udp_($i) [new Agent/UDP];# agent UDP
        $ns_ attach-agent $node_([expr ($numN)]) $udp_($i);
                 # l'attache agent d'envoie a un noeud
        set null_($i) [new Agent/Null];
                # l'attache agent qui recois a un noeud
        $ns_ attach-agent $node_([expr (($numN+1)/2-1)]) $null_($i);
                #Attache de l'agent au noeud jusqu'en haut
        set numN [expr ($numN+1)/2-1];
                #monter dans l'arbre d'une etape ou d'ue branche
        puts "valut du noeud $numN";
        set cbr_($i) [new Application/Traffic/CBR];

       if {$i == 0}
        {
            #le demarage de l'envoie de la demande de l'ajout
             $ns_ at 22.557023746220865 "$cbr_($i) start"
             $ns_ at 22.757023746220864 "$cbr_($i) stop"

        puts "valut du noeud $";
        set udp_($i) [new Agent/UDP];
        $ns_ attach-agent $node_($i) $udp_($i);
```

```
set null_($i) [new Agent/Null];
set null_([expr ($i+1)]) [new Agent/Null];
$ns_ attach-agent $node_([expr (($i+1)*2-1)]) $null_($i);
$ns_ attach-agent $node_([expr (($i+1)*2)]) $null_([expr
$i+1)]);
}
}
```

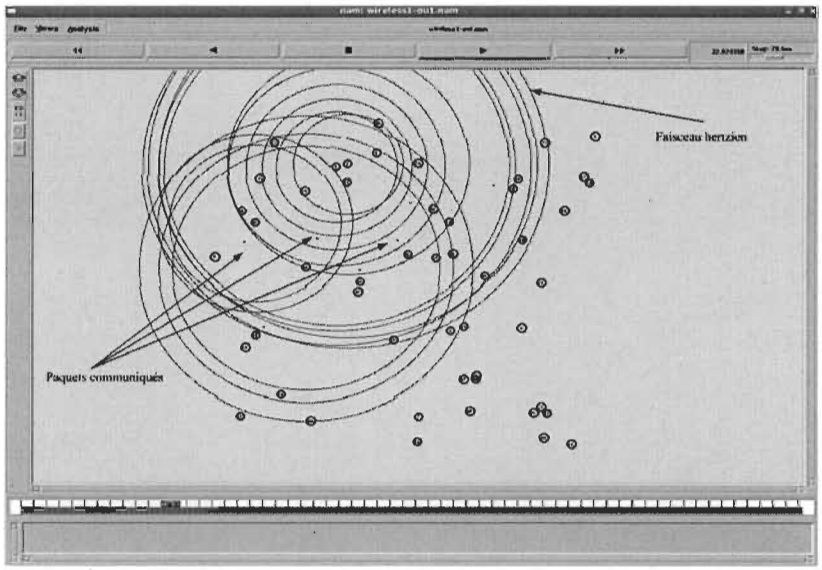


Figure 35: Ajout d'un nœud

6.3.3.4.2 Le départ

Dans cette partie, je traite l'opération du départ du même nœud 50, ce dernier envoie la réquette du départ contenant son identité *$numN*, l'agent d'attache du nœud en question sera detectée `$ns_ attach-agent $nod,` pour confirmer la suppression du nœud et la libération de cet agent `set numN [expr ($numN-1)/2+1].`

Programme TCL

```
while {$numN > 0} {
  puts "valut du noeud $numN";
  set udp_($i) [new Agent/UDP];    # agent UDP
  $ns_ attach-agent $node_([expr ($numN)]) $udp_($i);
   set null_($i) [last Agent/Null];

  $ns_ attach-agent $node_([expr (($numN-1)/2+1)]) $null_($i);
```

```
        set numN [expr ($numN-1)/2+1];
        puts "valut du noeud $numN";
        set cbr_($i) [new Application/Traffic/CBR];

       $ns_ connect $udp_($i) $null_($i)

        $cbr_($i) start

        $ns_ at 24.557023746220864 "$cbr_($i) start"
        $ns_ at 42.557023746220864 "$cbr_($i) stop"

        incr i

      if {$i == 0} {
            #le demarage de l'envoie de la demande de départ
             $ns_ at 22.557023746220865 "$cbr_($i) start"
             $ns_ at 22.757023746220864 "$cbr_($i) stop"
        }
     if {$i == 1} {
            $ns_ at 22.657023746220865 "$cbr_($i) start"
            $ns_ at 22.757023746220864 "$cbr_($i) stop"
        }
        if {$i == 2} {
              $ns_ at 22.757023746220864 "$cbr_($i) start"
             $ns_ at 22.857023746220864 "$cbr_($i) stop"
        }
       if {$i == 3} {
             $ns_ at 22.857023746220864 "$cbr_($i) start"
             $ns_ at 22.957023746220864 "$cbr_($i) stop"
        }
        if {$i == 4} {
             $ns_ at 22.957023746220864 "$cbr_($i) start"
             $ns_ at 23.057023746220864 "$cbr_($i) stop"
        }

         incr i
}
```

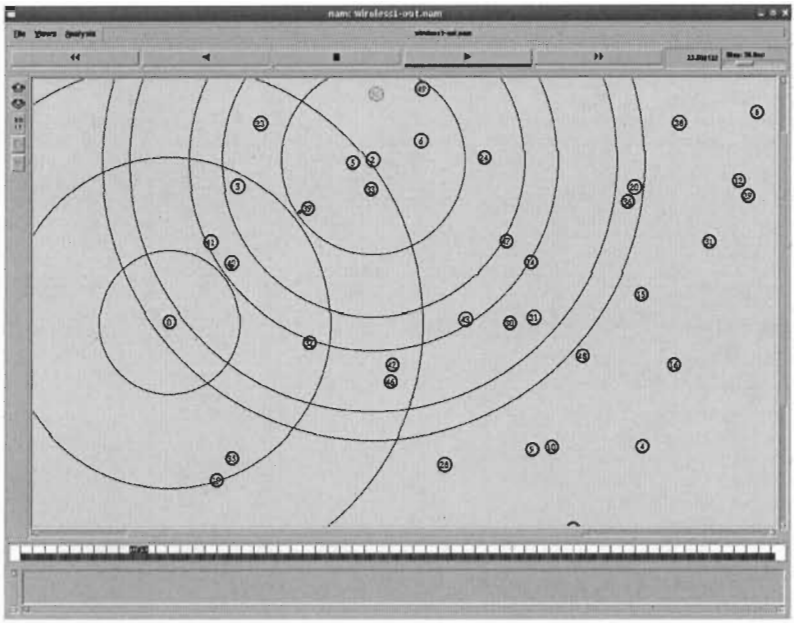


Figure 36: Départ d'un nœud

6.4 Conclusion

Dans ce chapitre et après l'explication des avantages des simulateurs réseau, j'ai expliqué ce qui est le simulateur NS-2 pour lequel j'ai opté. J'ai choisi ce simulateur pour la possibilité de simuler presque tous les modèles de simulation réseau, dans le cas de la dernière version NS-2.31 (lancée le 10 mars 2007) et aussi pour la présence des bibliothèques dédiées aux communications sans fil.

Malgré sa puissance, NS-2 présente plusieurs difficultés concernant sa compatibilité avec les systèmes d'exploitation. Jusqu'à présent, NS-2 n'est compatible qu'avec Unix (Linux) vu l'utilisation de sa console de commande pour écrire le code TCL et lancer les outils complémentaires de NS-2 par exemple le NAM.

Les opérations au niveau des nœuds, les départs et les ajouts, ont été correctement programmées et visualisées. Par contre, les prélèvements des paramètres à étudier n'étaient pas possibles à cause des difficultés rencontrées lors de l'intégration des librairies C++

d'Openssl dans le TCL. L'obstacle était l'impossibilité d'accéder au code source de la classe du noyau du simulateur pour la modifier et l'adapter, ainsi que le manque de la documentation nécessaire afin d'y procéder.

Suite à toutes mes questions au niveau des ressources existantes, aucun n'à été capable de répondre à une question. Vu que ce travail est pionnier dans le domaine, aucune étude n'est amenée à ce point à ce jour.

Chapitre 7 - Conclusion

Le monde est en train de vivre une véritable révolution de l'univers de la communication avec un besoin d'échanger une grande quantité d'informations entre plusieurs utilisateurs avec un minimum de consommation de la bande passante. C'est dans ce contexte que le multicast a vu le jour. Cette technique permet à un utilisateur d'envoyer à plusieurs autres utilisateurs les mêmes informations d'une manière simultanée sans causer l'encombrement de la bande passante ainsi que la surcharge du réseau. C'est dans ce sens que nous avons essayé d'étudier de près les méthodes utilisant le multicast pour échanger les clés de cryptage afin de mieux assurer la confidentialité des données circulantes au sein d'un réseau.

Parmi les réseaux sans fil, nous trouvons les réseaux ad hoc qui ne demandent aucune infrastructure, ce qui a valu un développement rapide de ce type de technologies. Néanmoins, cette technologie souffre de problèmes relatifs aux transmissions radio ou à la mobilité qui est un facteur clé au niveau de la qualité des services (transmission des paquets) et de la bonne exploitation de la bande passante du réseau. La mobilité des nœuds peut causer des déconnexions fréquentes et ce phénomène est très important lors des communications de groupe au sein des réseaux sans fil ad hoc, ce qui peut perturber l'opération de distribution des clés de groupe multicast. Pour cela la sécurisation reste un grand défi pour ce genre de réseaux.

Dans ce travail, j'ai exposé les différentes méthodes de distribution et/ou d'établissement de clés de groupe lors de la communication multicast dont j'ai dégagé les avantages et les inconvénients de chacune d'elles.

À travers ce travail de recherche, il a été question de la méthode GAKAP qui est le centre d'intérêt de notre travail. En effet, cette méthode s'inscrit dans une nouvelle optique de gestion et de distribution des clés de cryptage au sein d'un groupe multicast dans un réseau sans fil ad hoc et elle est basée sur la méthode TGDH. De plus, la méthode GAKAP doit contribuer à la diminution de l'exploitation des ressources (bande passante, alimentation espace de stockage,...) lors de la combinaison des clés du groupe et aussi lors des événements d'ajout et de départ des nœuds participants dans la session multicast.

Durant la réalisation de ce travail, nous avons rencontré plusieurs difficultés à savoir l'étude des théories des courbes elliptiques vu que c'est une technique très récente en plus de l'apprentissage du simulateur NS-2. En ce qui concerne les courbes elliptiques, nous avons modifié les bibliothèques préétablies d'Openssl, une tâche qui n'était pas facile. L'apprentissage du simulateur NS-2, son installation ainsi que ses outils complémentaires et son langage de programmation TCL ont été un défi à relever en plus de faire appel aux bibliothèques d'openssl. Au départ, ce n'était pas tâche facile, mais après de longues recherches, nous avons constaté que pour arriver à faire appel au C/C++ dans TCL, il fallait modifier et aussi ajouter plusieurs paramètres dans la structure source de NS-2 pour simuler les réseaux sans fil ad hoc vu la différence de la structure de ces derniers par rapport aux réseaux sans fil ordinaires, ainsi que l'appel et l'insertion des librairies Openssl dans le simulateur, chose qui demande des connaissances techniques très avancées au niveau du code source et précisément celui de la classe *TCLObejct* qui est le noyau du simulateur.

Tous les algorithmes et les codes nécessaires à la communication multicast sur les réseaux sans fil mobiles ont été développés sur simulateur NS-2.

Bibliographie

[1] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam, "**Secure group communications using key graphs**". in Proceedings of ACM SIGCOMM '98, September 1998.

[2] Sally Fyold, Vern Paxon, **"Difficulties in simulating the internet"**, IEEE/ACM, Transaction on networking, vol 9, no 4, pages. 392-403, August 2001.

[3] RFC 1112 Steve Deering. "**Host Extensions for IP Multicasting**", August 1989.

[4] RFC 2236 W. Fenner "**Internet Group Management Protocol, version 2**". November 1997.

[5] M. Aydos, T. Yanik, and C. K. Koc. "**High-speed implementation of an ECC-based wireless authentication protocol on an ARM microprocessor**". *IEEE Proceedings - Communications,* 148(5):273-279, October 2001.

[6] Cain Steve Deering, Isidor Kouvelas, Ajit Thyagarajan. **"Internet Group Management Protocol, Version 3"**,AT&T Labs – Research A. Thyagarajan Ericsson October 2002.

[7] Deering, S., "**MBone: The Multicast Backbone**". CERFnet Seminar, Mar. 3, 1993,

[8] Yajnik, Kurose et al. "**Packet Loss Correlation in the MBone Multicast Network: Experimental Measurements and Markov Chain Models**". (1996) .

[9] Vipul Gupta, Douglas Stebila, Stephen Fung, Sheueling Chang, Nils Gura, and Hans Eberle. "**Speeding up Secure Web Transactions using Elliptic Curve Cryptography**". 11th Ann. Symp. on Network and Distributed System Security — NDSS 2004, Internet Society, February, 2004.

[10] Adamson, B., **"Tactical Radio Frequency Communication Requirements for IPng"**, RFC 1677, August 1994.

[11] Y. Amir, Y. Kim, C. Nita-Rotaru, G. Tsudik. "**On the performance of Group Key Agreement Protocols**". ACM Transactions on Information and System Security (TISSEC), Volume 7, pp. 457 – 488, August 2004.

[12] Suna Melek Önen. "**La sécurité des communications multipoints pour les réseaux satellitaires: Une approche centrée sur la satisfaction des utilisateurs".** PhD thesis de l'Ecole Nationale Supérieure des Télécommunications de Paris Juillet 2005.

[13] Cordeiro de Morais Cordeiro, Hrishikesh Gossain, and Dharma P. Agrawal. "**Multicast over wireless mobile ad hoc networks**". present and future directions. *IEEE Network*, 17:52–59, Jan.-Feb. 2003.

[14] Imrich Chlamtac, M.Conti, Jennifer J-N. Liu. "**Mobile ad hoc networking: imperatives and challenges**". Elsevier , 2003.

[15] F. Bai, N. Sadogopan, A. Helmy. "**IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of Routing protocols for Adhoc Networks".** In IEEE INFOCOM, 2003.

[16] C. Bettstetter. Smooth is Better than Sharp. "**A Random Mobility Modelfor Simulation of Wireless Networks**". In ACM MSWiM, July 2001.

[17] F. Bergadano, D. Cavagnino, B. Crispo. "**Issues in Multicast Security**" Cambridge Intl. Workshop on Security Protocols, UK, LNCS 1796, Springer, pp 119-131, April, 1999.

[18] Manoj Pandey and Daniel Zappala. "**The Effects of Mobility on Multicast Routing in Ad Hoc Networks**". March, 2004 (Technical Report, UO-CIS-TR-2004-2).

[19] W. Ford. "**Computer Communication Security: Principles, Standard Protocols and Techniques**". Prentice Hall, 1994.

[20] J. Snoeyink, S. Suri, G. Varghese. "**A Lower Bound for Multicast Key Distribution**". In Proc. Of INFOCOM 2001, 2001.

[21] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. " **Handbook of Applied Cryptography**". CRC press series on Discrete mathematics and its Applications. CRC Press 1997 ISBN 0-8493-8523-7.

[22] V. S. Miller, **"Use of elliptic curves in cryptography"** in *Advances in Cryptology—CRYPTO'85*, H. C. Williams, Ed. Heidelberg, Germany: Springer-Verlag, 1986, vol. 218, Lecture Notes in Computer Science, pp. 417–426.

[23] Y. Amir, Y. Kim, C. Nita-Rotaru, G. Tsudik "**On the performance of Group Key Agreement Protocols**". In Proc. Of the 22nd International Conference on Distributed Computing Systems (ICDCS 2002), Vienna, Austria, July 2-5, 2002.

[24] T. Dunigan, C. Cao. "**Group Key Management**". Tchnical Report, ORNL/TM-13470, Computer Sceince and Mathematics division, U.S. Departement of Energy, 1998.

[25] G. H. Chiou, W. T. Chen. "**Secure Broadcast using Secure Lock**". IEEE transaction on Software Engineering, August 1989.

[26] H. Chu, L. Qiao, K. Nahrsteds. "**A Secure Multicast Protocol with Copyright Protection**". In Proc. Of IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology, January, 1999.

[27] D. Wallner, E. Harder, R. Agee. "**Key management for Multicast: Issues and architectures**". RFC 2627, June, 1999.

[28] M. Aydos, T. Yanik, and C. K. Koc. "**High-speed implementation of an ECC-based wireless authentication Protocl on an ARM Microprocessor**". IEEE Proceedings – Communications, 148(5):273-279, October 2001.

[29] A. Perrig, D. Song, J.D. Tygar. "**ELK, a new protocol for Efficient large-Group Key Distribution**". In Proc. Of IEEE Security and privacy Symposium, S&P2001, May, 2001.

[30] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, B. Plattner. "**The Verskey Framework: Verstile Group Key Management**". IEEE Journal on selected Areas in Communications (Special Issue on Middleware), 17(9), pp. 1614-1631, August 1999.

[31] T. Ballardie. "**Scalable Multicast Key Distribution**", RFC 1949, Mai, 1996.

[32] T. Ballardie, B. Chain, Z. Zhang. "**Core based Trees (CBT version 3) Multicast Routing**", Internet draft, August 1998.

[33] S. Mittra. "**Iolus: A Framework for Scalable Secure Multicasting**". In Proc. ACM SIGOMM, pp. 277-288, Cannes, France, September 1997.

[34] C. Shields, J.J. Garcia-Luna-Aceves. "**KHIP-A Scalable Protocol for Multicast Routing**". In Proc. Of the ACM SIGCOMM99, USA, September 1999.

[35] L.R. Dondeti, S. Mukherjee, A. Samal. "**DISEC: a Distributed Framework for Scalable Secure Many-to-may Communication**". In Proc. Of the 5th IEEE Sympoium on Computers and Communications (ISCC00) France, July, 2000.

[36] R. Molva, A. Pannetrat. "**Scalable Multicast Security in Dynamic Groups**". In 6th ACM Conference on Computer and Communications Security, Singapore, pp. 101-112, November, 1999.

[37] W. Diffie, M. Hellman. "**New Directions in Cryptography**". IEEE Transactions on Information Theory, IT-22(6), pp. 644-654, November, 1976.

[38] A.Fiat, M. Naor. "**Braodcast encryption**". In Advances in Cryptology-CRYPTO'93, Douglas R. Stinson, Ed. 1993, vol. 773 of Lecture Notes in Computer Science, pp. 480-491, Springer-Verlag, Berlin Germany, 1993.

[39] I. Ingemarsson, D. Tang, C. Wong. "**A Conference Key Distribution System**". IEEE Transactions on Information Theory, 28(5), pp. 714-720, September 1982.

[40] M. Steiner, G. Tsudik, M. Waidner. "**Diffie-Hillman Key Distribution Extended to group Communication**". ACM symposium on Computer and Communication Security, March, 1996.

[41] R. Canetti, B. Pinkas. "**A taxonomy of multicast security issues**". Internet Engineering Task Force, August 2000.

[42] D. Steer, L. L. Strawczynski, W. Diffie, M. Weiner. "**A Secure Audio Teleconference System**". In CRYPTO88, 1988.

[43] Y. Amir, Y. Kim, C. Nita-Rotaru, G. Tsudik. "**On the Peroformance of Group Key Agreement Protocols**". ACM Transactions on Information and System Security (TISSEC), volume 7, August 2004, pp. 457-488.

[44] Y. Kim, A. Perrig, G. Tsudik. "**Communication-Efficient group Key Agreement**". In Proc. Of IFIP SEC 2001, June, 2001.

[45] A. Perrig. "**Efficient collaborative Key Management protocols for Secure Autonomous Group Communication**". In Proc. Of Intl. Workshop on cryptographic techniques and E-commerce, 1999.

[46] K. Lauter, **"The advantages of elliptic curve cryptography for wireless security"** *IEEE Wireless Commun.,* vol. 11, no. 1, pp. 62–67, Feb. 2004.

[47] I. Blake, G. Seroussi, and N. Smart,**"Elliptic Curves in Cryptography"**. Cambridge, U.K.: Cambridge University Press, 1999, vol. 265, London Mathematical Society Lecture Note Series.

[48] N. Koblitz, **"Elliptic curve cryptosystems"** *Math. Comput.,* vol. 48, no. 177, pp. 203–209, 1987.

[49] Marie Virat, **"Chiffrement El Gamal sur une courbe de Weierstrass",** Laboratoire J.A. Dieudonné, U.M.R. C.N.R.S. N_6621, Université de Nice Sophia-Antipolis, 26 novembre 2004.

[50] R. Schoof,**"Elliptic curves over finite fields and the computation of square roots mod p"** *Math. Comp.,* vol. 44, pp. 483–494, 1985.

[51] T. Satoh, **"The canonical lift of an ordinary elliptic curve over a finite field and its point counting"** *J. Ramanujan Math. Soc.,* vol. 15, no. 4, pp. 247–270, 2000.

[52] A. Menezes, T. Okamoto, and S. Vanstone, **"Reducing elliptic curve logarithms to logarithms in a finite field"** *IEEE Trans. Inf. Theory,* vol. 39, no. 5, pp. 1639–1646, September. 1993.

[53] N. Smart, **"The discrete logarithm problem on elliptic curves of trace one"** *J. Cryptol.*, vol. 12, pp. 193–196, 1999.

[54] P. Gaudry, F. Hess, and N. P. Smart, **"Constructive and destructive facets of weil descent on elliptic curves"** *J. Cryptol.*, vol. 15, no. 1, pp. 19–46, 2002.

[55] R. Gallant, R. Lambert, and S. Vanstone, **"Improving the parallelized Pollard lambda search on anomalous binary curves"** *Math. Comput,* vol. 69, pp. 1699–1705, 2000.

[56] *Digital signature standard (DSS)*, FIPS PUB 186-2, National Institute of Standards and Technology (NIST), 2000.

[57] *Public key cryptography for the financial services industry: Keyagreement and key transport using elliptic curve cryptography*, ANSI X9.63-2001, American National Standards Institute.

[58] Standards for Efficient Cryptography Group (SECG), **"SEC 1: Elliptic curve cryptography,"** 2000.

[59] Standards for Efficient Cryptography Group (SECG), **"SEC2: Recommended elliptic curve domain parameters, "** 2000.

[60] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. *IETF* **"Criteria for Evaluating Reliable Multicast Transport and Application Protocols"**, June 1998. Request for Comments: RFC2357.

[61] R. Gallant, R. Lambert, and S. Vanstone, **"Improving the parallelized Pollard lambda search on anomalous binary curves"** *Math. Comput.*, vol. 69, pp. 1699–1705, 2000.

[62] Maria Striki, John S. Baras. "**Key Distribution Protocols for Multicast Group Communication in MANETs**". ISR Technical Reports, no TR 2003-17, 2003.

[63] **A. Boumso**, **S. Khali**, B. Amar Bensaber, I. Biskri, ***"Multicast key agreement protocol for mobile Ad-Hoc Networks"***, Conférence Winsys 2006 (*the International Conference on Wireless Information Networks and Systems*), Lisbonne, Portugal, August 2006.

[64] A. Boumo, **" Méthode exploratoire de distribution des clés de cryptage lors des communications de groupe dans un réseau mobile ad hoc",** Mémoire de maîtrise en mathématique & informatique appliquée, *Bibliothèque UQTR,* Mars 2006.