République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaïa
Faculté des Sciences Exactes
Département d'Informatique
Option : Administration et Sécurité des Réseaux

MÉMOIRE DE MASTER PROFESSIONNEL

Thème

Amélioration d'un protocole de gestion de clés hiérarchique dans les réseaux AD-Hoc.

Présenté par :

M. ***IDIR*** **Amine** et *M.* ***KHENTACHE*** **Amirouche**

Devant le jury :

Président	**M. OMAR MAWLOUD**	**M.C.A**	**U. A/Mira Béjaia.**
Rapporteur	**M. SAADI MUSTPHA**	**M.A.A**	**U. A/Mira Béjaia.**
Examinateur	**M. FARAH ZOUBEYR**	**M.C.B**	**U. A/Mira Béjaia.**
Examinateur	**M. CHEKRID Mohamed**	**Doctorant**	**U. A/Mira Béjaia.**

Promotion 2015 / 2016.

Remerciements

Nous remercions notre famille ainsi que nos proches qui nous ont épaulés tout au long de ces années.

Nous remercions chaleureusement Mr SAADI de nous avoir proposé ce sujet de mémoire, et de l'attention qu'il a porté à notre travail : Ses conseils et ses commentaires précieux nous ont permis de surmonter nos difficultés et de progresser dans notre projet.

Nos vifs remerciements d'adressent plus particulièrement à nos collègues MAOUCHE Lydia, KHANOUCHE Feriel et HEMECHE Selma pour leurs aide, leurs disponibilité ainsi que leurs générosité, nous leurs souhaitons bien du succès pour leurs soutenance prochaine ainsi que de la réussite dans leurs vie.

A tous ceux qui ont fait de nous ce que nous sommes aujourd'hui.

Dédicaces

A la mémoire de la mère de Mr SAADI que dieu l'accueil dans son vaste paradis.

Au petit "Amine Bilal" félicitations aux parents.

For the EPL Player of the year 2015/2016 "RYAD MAHREZ".

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

BD	**B**urmester **D**esmedt
CGKA	**C**ontributory **G**roup **K**ey **A**greement
CKD	**C**entralized Group **K**ey **D**istribution
CRTDH	**C**hinese **R**emainder **T**heorem and **D**effie-**H**ellman
DCGKA	**D**ynamic **C**ontributory **G**roup **K**ey **A**greement
DLP	**D**iscrete **L**ogarithm **P**roblem
DT	**D**is**T**rust
ECC	**E**lliptic **C**urve **C**ryptography
ECDLP	**E**lliptic **C**urve **D**iscrete **L**ogarithm **P**roblem
ECGK	An **E**fficient **C**lustering scheme for **G**roup **K**ey managment in MANETs
GC	**G**roup **C**ontroller
GDH	**G**roup **D**iffie-**H**ellman
GECDH	**G**roup **E**lliptic **C**urve **D**iffie-**H**ellman
GKA	**G**roup **K**ey **A**greement
GKMPAN	An Efficient **G**roup Re**K**eying Scheme for Secure **M**ulticast **P**rotocol in **A**d-Hoc **N**etwork
HDH	**H**ypercubic **D**iffie-**H**ellman
KEK	**K**ey **E**ncryption **K**ey
LID	**L**owest-**ID**
MANET	**M**obile **A**d hoc **NET**work
MR	**M**ulticast **R**outer
PT	**P**artial **T**rust
RWP	**R**andom **W**ay**P**oint
SGC	**S**ecure **G**roup **C**ommunication
SGCP	**S**ecure **G**roup **C**ommunication **P**rotocol
STR	**S**teer et al
TEK	**T**raffic **E**ncryption **K**ey
TFAN	**T**ree-based Group Key Agreement **F**ramewok for Mobile **A**d-Hoc **N**etworks
TGDH	**T**ree **G**roup **D**iffie-**H**ellman
TGECDH	**T**ree **G**roup **E**lliptic **C**urve **D**iffie-**H**ellman
TRP	**T**hree **R**ound **P**rotocol

TT	**T**otal **T**rust
VDBP	**V**irtual **D**ynamic **B**ackbone **P**rotocol
WAN	**W**ide **A**rea **N**etwork

Introduction Générale

Les réseaux mobiles Ad-Hoc annoncent les réseaux de communication du futur où la mobilité en est l'idée maîtresse. Ces réseaux devront être capable d'interconnecter des nœuds en groupe, pour leurs fournir des services de manière omniprésente. Ils sont de ce fait plus vulnérables à de nombreux types d'attaques. Leurs succès dépendra sans aucun doute du niveau de sécurité qu'apporteront ces derniers à leurs usagers. Les solutions de sécurité traditionnels ne répondent pas aux nouvelles exigences de tels réseaux.
Afin de sécuriser des communications de groupe dans les réseaux mobiles Ad-Hoc, une clé partagée entre tous les membres est nécessaire. Cette clé doit être mise à jour après chaque évènement de groupe ajout, départ ou révocation d'un membre du groupe. Dans ce mémoire nous nous focalisons sur des mécanismes pouvant assurer une gestion efficace de clés de groupe avec le moindre coût de communication et de calcul. Il s'agit en effet, d'incorporer la notion de cryptographie à courbes elliptiques (ECC) dans un protocole d'accord de clés. La sécurité d'un protocole se reposant sur ECC peut être assurée par le problème du logarithme discret à courbes elliptiques. Ce mécanisme permet de réduire considérablement la taille des clés échangés avec le même niveau de sécurité que des mécanismes classiques ce qui pourra améliorer considérablement les performances d'accord de clés pour ces réseaux.

Dans notre mémoire nous nous sommes intéressés aux deux protocoles Three Round Protocol (TRP) ainsi qu'an efficient clustering scheme for group key management in MANETs (ECGK) le premier est un protocole d'accord de clé, le deuxième est un protocole de clustering basé sur la clusterisation à travers la confiance et qui se repose sur TRP pour la gestion des clés.

Ce mémoire est organisé comme suit :

Dans le chapitre 1, nous présentons un préambule sur les réseaux ad hoc et leurs caractéristiques, nous faisons aussi un état de l'art sur quelques protocoles existants portants sur l'échange de clés. Ensuite, sont abordés la notion de cryptographie à courbes elliptiques, et quelques protocoles basés sur cette dernière, dans le chapitre 2.

Dans le chapitre 3, nous présentons l'amélioration apportée aux deux protocoles TRP et ECGK, nous développons cette dernière par des explications et des exemples, et nous conclurons ce chapitre par une évaluation du coût de communication, de calcul et de mémoire. Le chapitre 4 est consacré quant à lui à la partie simulation de notre travail, nous y présenterons l'interface du simulateur développé à cet effet, le profil de simulation choisi et les résultats obtenus. Enfin nous clorons ce mémoire par une conclusion générale.

Chapitre 1

Généralités sur les MANETs et la gestion de clés de groupe

1.1 Introduction

La gestion de clé de groupe peut être classifiée en modèles : centralisé, décentralisé et distribué. Le modèle centralisé qui emploie un contrôleur de clé pour les tâches de gestion de clé comprenant la génération des clés, le transfert, la distribution, la révocation, etc. n'est pas approprié aux MANETs.

Le modèle décentralisé (ou semi-distribué) subdivise un groupe en sous-groupes en général hiérarchiquement pour étendre la charge de travail d'une unité centrale de traitement. Cette approche convient aux réseaux filaires puisque les contrôleurs des clés sont fixes et les communications pour verrouiller des contrôleurs sont fiables. Cependant, il n'est pas approprié aux MANETs où les nœuds sont mobiles et la communication sans fil est souvent incertaine.

Le modèle distribué n'a pas un contrôleur fixe pour la gestion de clé de groupe. Au lieu de cela, une clé de groupe est produite d'une façon contributive par tous les membres dans le système. Les protocoles d'accord de clé de groupe GKA (Group Key Agreement) qui permettent aux participants de convenir sur une valeur secrète commune, sont basées sur la contribution publique de chaque participant. Ils n'ont pas besoin de la présence d'une autorité centrale. Nous allons présenter dans le suivant chapitre les réseaux Ad Hoc ainsi qu'un état de l'art sur les protocoles de gestion de clé de groupe dans les réseaux mobiles Ad Hoc (MANETs).

1.2 Réseaux mobiles Ad-Hoc

Un réseau mobile Ad Hoc est un environnement mobile sans infrastructure, appelé généralement MANET (Mobile Ad Hoc NETwork). Il consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou une administration centralisée [2].

Par ailleurs, les systèmes de communication cellulaire sont basés essentiellement sur l'utilisation des réseaux filaires (tel que Internet ou ATM) et la présence des stations de base qui couvrent les différentes unités mobiles du système. Les réseaux mobiles « Ad Hoc » sont à l'inverse, des réseaux qui s'organisent automatiquement de façon à être déployé rapidement, sans infrastructure fixe, et qui doivent pouvoir s'adapter aux conditions de propagation, aux trafics et aux différents mouvements pouvant intervenir au sein des nœuds mobiles.

1.2.1 Caractéristiques des réseaux Ad Hoc

Les réseaux mobiles Ad Hoc sont caractérisés par ce qui suit :

Topologie dynamique : Les unités mobiles du réseau, se déplacent d'une façon libre et arbitraire. Par conséquent la topologie du réseau peut changer, à des instants imprévisibles, d'une manière rapide et aléatoire. Les liens de la topologie peuvent être unis ou bidirectionnels.

Bande passante limitée : Une des caractéristiques primordiales des réseaux basés sur la communication sans fil est l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte soit modeste.

Contraintes d'énergie : Les hôtes mobiles sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables. Le paramètre d'énergie doit être pris en considération dans tout contrôle fait par le système.

Sécurité physique limitée : Les réseaux mobiles Ad Hoc sont plus touchés par le paramètre de sécurité, que les réseaux filaires classiques. Cela se justifie par les contraintes et limitations physiques qui font que le contrôle des données transférées doit être minimisé.

Absence d'infrastructure : Les réseaux Ad Hoc se distinguent des autres réseaux mobiles par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau d'une

manière continue.

Notion de « multisaut » : un réseau Ad Hoc est qualifié par « multisaut » car plusieurs nœuds mobiles peuvent participer au routage et servent comme routeurs intermédiaires.

1.3 Protocoles de gestion de clés dans les réseaux mobiles Ad Hoc

1.3.1 Traitement des événements de changement dans le groupe

Il existe une forte relation entre les événements de gestion de groupe et les protocoles des gestion de clés de groupe. En effet, ces dernier nécessitent les techniques d'ajustement des clés lors de l'occurrence de changement dans la composition du groupe. Le groupe peut changer fréquemment tant que quelques membres veulent quitter ou rentrer dans le groupe. Le système d'accord de clés de groupe doit être capable de gérer efficacement les événements de changement dans le groupe, en garantissant par ailleurs que les propriétés importantes suivantes soient satisfaites :

1. *Le secret calculable de la clé :* elle garantie qu'il est infaisable par calcul pour tout adversaire passive de découvrir toute clé de groupe précédemment utilisée.
2. *Le secret décisionnel de la clé :* elle s'assure qu'il n'existe aucune fuite d'informations autres que celles des clés publiques blindées.
3. *l'indépendance de la clé :* elle garantie qu'un adversaire passive qui connaît son propre sous-ensemble de clés de groupe ne peut découvrir aucune des clés restantes. L'indépendance de la clé peut être décomposé en *secret futur* et *secret passé*. Le secret futur garantit qu'un adversaire passif qui connaît une suite de sous-ensemble des anciennes clés ne peut découvrir la clé actuelle du groupe. Le secret passé garantit qu'un adversaire passif qui connaît une suite d'un sous-ensemble de clés ne peut en aucun cas découvrir les clés précédente du groupe.

Certainement, la méthode la plus simple qui permet de manipuler les événements de gestion de groupe est de ré-exécuter les protocoles de génération des clés. Cependant, dans les réseaux Ad Hoc sans fil, le coût de calcul et de communication imposés à chaque fois après la ré-exécution de ces protocoles peut sévèrement épuiser les ressources des unités mobiles (comme la batterie). Par conséquent, il est toujours intéressant de penser aux solutions qui sont construites en dessus des clés existantes, réduisant les modifications à apporter (dans le but de traiter chaque événement de changement de groupe) et ainsi minimiser donc l'énergie totale consommée.

D'après K.Drira et al. [3] les protocoles hiérarchiques de gestion de clé de groupe dans MANETs peuvent être classifiés dans deux approches : l'approche horizontale et l'approche orientée topologie, comme montré dans la figure 1.1.

FIGURE 1.1 – Taxonomie de quelques protocoles de gestion de clés de groupe pour MANETs.

1.3.2 Approche horizontale

Dans cette approche, il n'y a aucune organisation antérieure des membres de groupe et tous ces derniers partagent une TEK (clé de trafic) commune. La gestion de cette clé est centralisée à un serveur unique ou distribuée entre tous les membres de groupe.

Dans [4], les auteurs proposent CRTDH (Chinese Remainder Theorem and Diffie-Hellman based scheme for secure group communication). CRTDH est un protocole de gestion de clé contributif, cela signifie que la TEK commune est calculée en fonction de l'information contribuée par tous les participants du groupe. CRTDH a besoin de deux séries de diffusion pour établir la TEK. Il n'assure aucune autorité centrale de confiance et le calcul est également distribué sur tous les membres. Cependant, il est basé sur la diffusion et par conséquent non scalable. En outre, il suffit qu'un membre soit défaillant (en panne) ou un message soit perdu pour bloquer le protocole en entier ou des membres (faire calculer des différents TEKs).

Dans [5], les auteurs proposent le protocole Two Round Key Agreement Protocol nommé TRP. Dans TRP, l'initiateur du protocole devient le leader de groupe. Le leader commence par diffuser un message INIT pour lancer le processus de calcul de clé. Puis, le protocole a besoin de deux Tours :

Tour 1 : Chaque participant i, répond à la demande INIT en choisissant un secret r_i et en envoyant sa clé publique g^{r_i} à l'initiateur.

Tour 2 : Le leader de groupe élève à la puissance de la clé publique de chaque membre se joignant à son secret r_l et les diffuse avec les contributions originales au groupe, c'est-à-dire il envoie $\{g^{r_i}, g^{r_l r_i}\}$ pour tout i = 1, n et i $\neq$ l, n est le nombre de participants. Puis, chaque membre i vérifie si sa contribution est incluse correctement, enlève son secret r_i de $g^{r_l r_i}$ pour obtenir g^{r_l} et calcule la clé de groupe :

$$TEK = g^{r_i} * \prod_{i=1,n}^{i \neq l} g^{r_l r_i} = g^{r_l(1+\sum_{i=1,n}^{i \neq l} r_i)}.$$

TRP exige moins de coût en messages échanges que CRTDH. Néanmoins, TRP souffre du problème du point central de défaillance et est vulnérable aux messages retardés ainsi qu'aux défaillances de nœuds.

Dans [6], les auteurs proposent une version authentifiée de TRP en ajoutant un troisième Tour au protocole initial. Ce Tour est employé pour l'authentification des membres.

Dans [7], les auteurs proposent GKMPAN (An Efficient Group Rekeying Scheme for Secure Multicast in Ad Hoc Networks), une approche probabiliste de gestion de clés de groupe. GKMPAN assume l'existence d'un serveur de clés pour la distribution de la clé initiale et envoie les mises à jour authentifiées de clés de groupe aux nœuds. Durant l'initialisation de GKMPAN, tous les nœuds dans le réseau ad-hoc sont dotés (données) un certain nombre m de clés en dehors d'un grand ensemble de l clés. L'ensemble d'index des clés qu'un nœud possède est déterministe et peut être publiquement calculé de son identifiant unique du nœud. Ceci permet à n'importe quel nœud de pouvoir calculer quelles clés des nœuds en particulier et communiquer directement avec eux. La vitesse de la distribution de la clé comme la sécurité dépend du nombre m des clés localement stockées et de la taille de la clé de l'ensemble l. Si m est très petit comparé au l puis la probabilité d'avoir une clé partagée avec n'importe quel voisin est petite. Ainsi, deux nœuds quelconques qui n'ont pas la clé partagé n'ont pas besoin de trouver un ou plusieurs nœud(s) intermédiaire à pouvoir communiquer directement.

Dans [8], les auteurs proposent un protocole de gestion de clé distribué basé sur la cryptographie à seuil. La particularité de cette approche est la gestion des nœuds compromis. Cependant, elle est basée sur des diffusions et par conséquent n'est pas scalable.

L'approche horizontale souffre du problème 1-affects-n, ou le changement de composition de groupe (jointure ou départ) affecte tous les membres de groupe. D'ailleurs, la plupart des protocoles de cette approche ont besoin d'un serveur central. Ainsi, ils ne sont ni scalables ni tolérants aux défaillances.

1.3.3 Approche orientée topologie

Cette approche propose d'organiser les membres de groupe selon une topologie virtuelle. Les structures les plus utilisés sont les arbres hiérarchiques, mais d'autres structures de groupe sont également proposées. Le but principal de l'utilisation d'une topologie virtuelle est de diminuer le coût du processus d'ajout ou suppression afin de faire face au problème 1-affects-n. L'inconvénient d'une telle approche est son coût de mise à jour des clés lors de changement de la topologie particulièrement dans MANETs. En raison de la mobilité, la mise à jour de la topologie de la structure de groupe peut causer d'énormes coûts de communication. Généralement, ces structures ne sont pas à organisation automatique. En cas de défaillance, il est souvent le cas que la gestion des clés doit redémarrer et synchroniser la structure du groupe avant de lancer le processus d'établissement de clé à nouveau.

Dans [9], un certain nombre des protocoles de gestion de clé de groupe orienté topologies tels que STR [10], CLIQUES [11] et TGDH [12] qui ont été à l'origine conçus pour des réseaux filaires sont testés et adaptés dans MANETs avec des limitations sur le coût de calcule des clés. Cela induit à modifier des versions de ces protocoles (STR, CLIQUES et TGDH) en employant la cryptographie à courbe elliptique qui est considérée plus applicable pour les dispositifs mobiles que la cryptographie à clé publique en raison de la plus petite taille de la clé et du calcul plus efficace [9].

Dans [13], les auteurs prouvent que bien que les coûts dans des protocoles μSTR et $\mu TGDH$ soient sensiblement réduits comparés aux protocoles originaux de STR et TGDH, les coûts de calcul du STR sont encore plus importantes (élevés) que ceux de TGDH et les coûts de communication de TGDH sont en général plus importants (élevés) que ceux de STR. Par conséquent, les auteurs proposent de combiner les deux protocoles dans une structure afin de réaliser un coût plus optimale entre le calcul et l'efficacité de communication. Ce protocole est appelé TFAN(Tree-based Group Key Agreement Framework for Mobile Ad-Hoc Networks). Dans μSTR et $\mu TGDH$ (comme dans les STR et TGDH originaux), les membres de groupe sont logiquement organisés en arbre binaire. Les protocoles basés sur un arbre réduisent le problème 1-affects-n en diminuant la complexité de l'échange de messages et en effectuant un changement d'adhésion à $O(\log_n^2)$ si n est la taille du groupe. Toutefois, ils apportent de nouveaux défis tels que

la synchronisation de membres dans l'arbre particulièrement quand les défaillances se produisent.

Dans [14, 15], les auteurs ont étudié des considérations de tolérance aux défaillances (scalabilité) dans des protocoles de gestion de clé de groupe dans MANETs. Ils ont proposé une version robuste du protocole de TGDH pour MANETs.

Dans [16], les auteurs proposent de subdiviser le groupe en sous-groupes. Chaque sous-groupe est supervisé par un leader tel que les leaders des sous-groupes forment un ensemble reliés de dominants. Les leaders de sous-groupes produisent une TEK par accord en utilisant un protocole de CLIQUES [11], appelé aussi GDH2. Cette clé est alors distribuée dans chaque sous-groupe. GDH2 est une généralisation du protocole d'accord de clés de Diffie-Hellman des deux parties [17], pour gérer des groupes. Avec GDH2, des membres de groupe doivent être organisés en un anneau (séquence). Le dernier membre dans l'anneau (la séquence) est le leader de groupe qui contrôle les opérations de jointure et de départ. Mais, cette organisation de groupe exige d'importants coûts de communication pour MANETs et est vulnérable aux défaillances de nœuds.

Dans [18], les auteurs proposent SGCP (Secure Group Communication Protocol). SGCP est un protocole de gestion de clé de groupe basé sur les clusters qui se fondent sur un protocole VDBP (Virtual Dynamic Backbone Protocol) détaillé dans [19, 20] pour construire des clusters. Pour réaliser le clustering, tous les nœuds dans VDBP créent une table d'information de couverture de voisinage . Cette table contient les enregistrements de fréquence d'échec de lien, du degré, d'identité, de puissance restante et des capacités de calcul. La table s'appelle le poids du nœud. Chaque nœud doit mettre à jour son information de poids périodiquement et la diffuser avec le message de vie à ses voisins d'un hop (saut). Quand un temps spécifié t_{wait} (temps d'attente) expire, chaque nœud vérifie si c'est le nœud local optimal, c'est-à-dire qu'il a le meilleur poids parmi les poids reçus. S'il est optimal, il change son état a un multicast router (MR). MRs doivent diffuser leurs poids et leurs identités dans le réseau. Chaque nœud non-MR choisit le meilleur poids et envoie un message de jointure à MR correspondant. Chaque MR constitue un sous-groupe avec tout nœud non-MR qui lui a envoyé un message de jointure, il produit l'identité et une clé de sous-groupe. Ensuite il chiffre avec les clés publiques des membres (identité et la clé de sous groupe) et la distribue aux membres du sous-groupe. Après la création de sous-groupes, la prochaine étape de SGCP est d'établir des chemins de connexions entre les sous-groupes. Dans cette étape, tous les nœuds doivent diffuser l'identité du sous-groupe à leur voisinage. Chaque nœud qui a au moins un voisin dans un sous-groupe différent s'appelle un connecteur. Chaque connecteur rassemble l'information de tous ses connecteurs adjacents dans une table et l'envoie à son MR. Ce processus de construction des connecteurs est lancé périodiquement pour assurer que le chemin de connexions est correct. Deux MRs adjacent peuvent alors établir une clé de communication en utilisant les protocoles d'accord de clé publiques existants. Ce protocole

favorise le nœud puissant en énergie pour qu'il soit un clusterhead mais aucun autre membre de deux sous-groupe ne peut avoir les mêmes critères de même groupe.

Dans [21], les auteurs proposent un protocole de gestion de clé de groupe basé sur les clusters où les clusters sont situés géographiquement sur des régions délimitées. Tous les nœuds sont équipés du GPS (grouper par satellite) et sont ainsi capables de connaître leurs endroits géographiques. Tous les membres du groupe partagent une clé de groupe secrète KG et tous les membres de sous-groupe d'une région Ri partagent une clé secrète KRi. Pour calculer ces clés, les membres de groupe emploient le protocole de gestion d'accord de clé de groupe GDH2 [19]. Ce protocole est semblable à SGCP [18] et les différences principales sont les limites géographiques des clusters, et le critère d'élection du clusterhead lequel utilise seulement l'identité.

Dans [22], J.H Li et al, décrivent un modèle de gestion de clés de groupe basé sur des clusters pour les groupes hiérarchiques, c'est-à-dire que les membres (noeuds) se groupent comme des applications militaires, où des membres sont organisés dans des couches différentes. Ainsi, le groupe est subdivisé en sous-groupes, chaque sous-groupe possèdera un leader et chaque sous-groupe sera encore organisé en hiérarchies. Chaque niveau de la hiérarchie s'appelle une rangée ou une couche et la génération de la clé, de la distribution, et de la transmission de données suivent la hiérarchie et toutes les clés sont distribuées par un serveur de clé centralisé. Les sous-groupes sont obtenus en utilisant le protocole de clustering DECA décrit dans [23]. Dans DECA, chaque nœud calcule une fonction qui capture son énergie résiduels, sa connectivité et son identité. Les nœuds diffusent leurs points dans leur *couverture voisinage*, le nœud qui a des meilleurs points lui permettra de s'annoncer clusterhead. Cependant, le modèle de gestion de clé qui est développé sur ce clustering compte sur un serveur central.

Dans [24, 25], les auteurs proposent un protocole de clustering appelé LID (Lowest-ID). LID est l'un des protocoles de clustering les plus connus. Il suit une approche heuristique ou de découverte, qui assigne une identification unique à chaque nœud et choisit le nœud avec l'identification minimum comme clusterhead. LID est habituellement utilisé comme protocole de référence pour l'évaluation des performances des modèles de clustering.

Dans [3] les auteurs proposent ECGK (An efficient clustering scheme for group key management in MANETs). ECGK est un protocole de gestion de clé de groupe basé sur les clusters, qui compte sur l'emploi de la quantification de confiance proposée dans [24], et les relations de confiance entre les nœuds sont classifiées en trois catégories : confiance total, confiance partielle et méfiance ; et emploie la confiance comme critère de clustering, tel que chaque cluster se compose de clusterhead, de noyau et de périphérie. L'algorithme de clustering possède trois phases :

- **Phase 1 :** Élection du clusterhead,
- **Phase 2 :** Formation du noyau,
- **Phase 3 :** Formation de la périphérie du cluster.

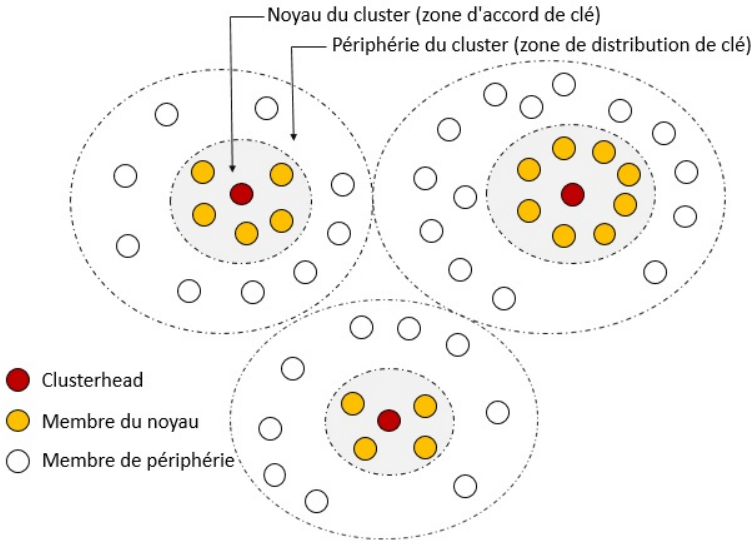

FIGURE 1.2 – Architecture générale d'ECGK.

Après avoir terminé la phase 1 et la phase 2, les nœuds entourant le noyau joignent le cluster selon les deux étapes suivantes :

- **Étape 1** (membres de périphérie avec des relations de confiance total).
- **Étape 2** (membres de périphérie avec des relations de confiance partielle).

Selon l'architecture ECGK, les membres de noyau ainsi que le clusterhead forment une zone d'accord de clé. Cela signifie que tous les membres de noyau contribuent au calcul du TEK du cluster en utilisant le protocole d'accord de clé TRP [6], et la périphérie est une zone de distribution de clé, cela signifie que les membres de périphérie obtiennent la TEK du cluster à partir des membres de noyau correspondant chiffrés avec KEK (la clé de chiffrement de la clé), ce dernier utilise aussi TRP. La communication sécurisé entre les clusters est assurée en calculant une TEK entre chaque paire de clusters adjacent, et chaque clusterhead calcule une clé de chiffrement du trafic avec le clusterhead de chaque cluster adjacent, en utilisant le protocole d'exchange de clé de Diffie-Hellman [17].

La clusterisation présente une solution intéressante pour simplifier et optimiser les fonctions et les services du réseau. Ainsi, ECGK propose la confiance comme critère de clustering pour la gestion de clé de groupe dans MANETs, la confiance peut être diffusée par la mobilité des nœuds et peut aider la détection des membres malveillants de groupe. Les résultats de simulation d'ECGK prouvent que c'est une bonne approche comparée à deux autres approches de la littérature (LID et SGCP).

1.4 Défis de gestion de clés dans MANETs

La Sécurité des Communications de Groupe (SGC) se réfère à un scénario dans lequel un groupe de participants peut émettre/recevoir des messages aux/des membres du groupe de façon que les nœuds externes aux groupes soient incapables d'intercepter aucune des informations échangées. La majorité des protocoles de SGC dans les MANETs utilise le Problème du Logarithme Discret (DLP) basée sur l'échange de Deffie-Hellman comme protocole d'accord de clés de base [26].

1.5 Conclusion

Dans ce chapitre, nous avons présenté les des réseaux Ad Hoc et quelques unes de ses caractéristiques. Nous avons présenté ensuite des protocoles de gestion de clés de groupe dans les réseaux mobiles Ad Hoc, où nous avons présenté des protocoles de gestion de clés de groupe existants dans la littérature et pour cela, nous avons présenté deux approches :

Approche orientée Topologie : Cette approche repose sur l'organisation des membres de groupe selon une topologie virtuelle.

Approche Horizontale : Dans cette approche, il n'y a aucune organisation antérieure des membres de groupe et tous les membres de groupe partagent un TEK commun.

Dans le chapitre suivant, nous allons aborder la question des courbes elliptiques, et présenté quelques protocoles de gestion de clés dans MANETs avec courbes elliptiques.

Chapitre 2

Gestion de clés de groupe avec courbe elliptique

2.1 Introduction

Dans ce chapitre nous présentons la définition d'une courbe elliptique et les différents lois de groupe. Nous présentons également le Problème du logarithme discret, Protocole d'échange de clés de Deffie-Hellman, Échange de clé et Sécurité.

Nous présentons également quelques Protocoles de gestion de clés de groupe dans MANETs avec courbes elliptiques (GECDH, TGECDH et DCGKA).

2.2 Cryptographie à courbes elliptiques

2.2.1 Définition d'une courbe elliptique

Soit K un corps fini, on appelle courbe elliptique sur K une courbe dans le plan projectif, cubique et sans points singuliers, et munie d'un point distingué qui jouera un rôle particulier : élément neutre. Elle est donc définie par un polynôme irréductible homogène en trois variables à coefficient dans K [27]. Par un changement de variables homographique, on peut toujours se ramener à une équation dite de Weierstrass :

$$y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3,\ Avec\ a_1, a_2, a_3, a_4, a_5, a_6 \in K$$

La courbe elliptique E est l'ensemble des points (x,y)$\in K^2$ satisfaisant cette équation et d'un point imaginaire ϑ appelé point à l'infini [28].

2.2.2 Loi de Groupe

Les applications des courbes elliptiques en cryptographie sont principalement dues à l'existence d'une loi de groupe que nous pouvons définir sur ces dernières. En effet, l'ensemble $E \cup \vartheta$ peut être doté d'une opération d'addition qui produit un groupe abélien dont l'élément neutre est le point à l'infini ϑ [29]. En cryptologie, les courbes elliptiques sont utilisées dans le corps F_p avec p un nombre premier strictement supérieur à 3.

Soit E une courbe elliptique définie sur F_p.

L'équation affine de Weierstrass de E peut être simplifiée à cette équation :

$$y^2 = x^3 + ax + b \tag{2.1}$$

Un exemple typique de courbe elliptique est donné sur la figure 2.1. Son équation est $y^2 = x^3 - 5x^2 + 3$.

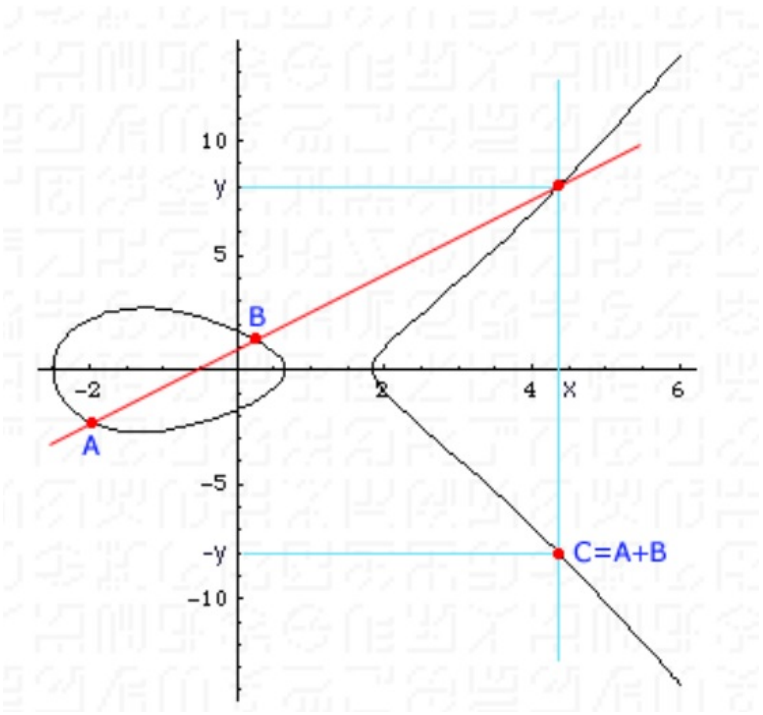

FIGURE 2.1 – Courbe elliptique d'équation $y^2 = x^3 - 5x^2 + 3$.

Soit F_p un corps fini premier de sorte que p soit un nombre premier impair, et soit a, b $\in F_p$, tel que :

$$4a^3 + 27b^2 \neq 0 \ (mod p) \tag{2.2}$$

puis une courbe elliptique E(F_p) de F_p définie par les paramètres a, b $\in F_p$ comprend l'ensemble de solutions ou de points P = (x, y) pour x, y $\in F_p$ à l'équation : $y^2 = x^3 - 5x^2 + 3$.

2.2.3 Addition de deux points

Prenons deux points P et Q sur cette courbe. En général, la courbe passant par P et Q recoupe la courbe en un troisième point de coordonnées (x, y). Son symétrique (x, -y) est lui aussi sur la courbe et on le désigne par P+Q pour signifier qu'il est construit à l'aide de P et Q. La chose surprenante est que cette opération "+" possède toutes les propriétés de l'addition des nombres. C'est-à-dire que l'on peut faire tous les calculs de type addition, soustraction et division avec un reste entier que nous faisons sur la droite des nombres réels sur cet courbe elliptique.

Règles de l'addition

Soient P = (x_1, y_1) et Q = (x_2, y_2) deux points sur E :

- Le point $(x_1, -y_1)$ est l'opposé du point P et il est noté –P.
- Si Q ≠ P et Q ≠ –P, alors le point R = P+Q = (x_3, y_3) est défini par :

$$\begin{cases} x_3 &= (\frac{y_1-y_2}{x_1-x_2})^2 - x_1 - x_2 \\ y_3 &= (\frac{y_1-y_2}{x_1-x_2})(x_3 - x_1) \end{cases} \tag{2.3}$$

- Si P = Q, alors le point 2P = (x_3, y_3) est défini par :

$$\begin{cases} x_3 &= (\frac{3x_1^2+a}{2y_1})^2 - 2x_1 \\ y_3 &= x_1 + (\frac{3x_1^2+a}{2y_1})(x_3 - x_1) \end{cases} \tag{2.4}$$

- Si $x_1 = x_2$ mais $y_1 \neq y_2$, alors R = ϑ.
- Si P = Q et $y_1 = 0$, alors R = ϑ.

2.2.4 Multiplication d'un point par un nombre entier

Les modèles cryptographiques basés sur ECC se fondent sur la multiplication scalaire des points des courbes elliptiques. étant donner un nombre entier k et un point p ∈ E(F_p), la multiplication scalaire est le processus d'ajouter p à elle-même k fois. Le résultat de cette multiplication scalaire est dénoté k*p ou kp. La multiplication scalaire des points des courbes elliptiques peut être calculée efficacement en utilisant la règle d'addition de l'ensemble .

Exemple 1 :

On remplace la multiplication par une série d'additions. Prenons un exemple :

Soit la courbe $y^2 \bmod 11 = (x^3 + x + 2) \bmod 11$.

Calculons p.d, avec d=3 et P=(4, 2).

On peut vérifier que le point P appartient bien à la courbe elliptique.

On peut remplacer 3*P par P + P + P. Calculons d'abord P + P. D'après la règle (2.4), P + P = (4, 2) + (4, 2) = (8, 4) = 2*P.

D'après la règle (2.3), 2*P + P = (8, 4) + (4, 2) = (2, 10) = 3·P.

Exemple 2 (exemple d'addition de courbe elliptique) :

Dans les règles (2.3) et (2.4) on peut le définir comme suit :

On pose : P = $(x_1, y_1) \in E(Z_p)$ et Q = $(x_2, y_2) \in E(Z_p)$, si p ≠ -Q. donc P + Q = (x_3, y_3), avec :

$$x_3 = \lambda^2 \text{ - } x_1 \text{ - } x_2$$
$$y_3 = \lambda(x_1 \text{ - } x_3) \text{ - } y_1$$
$$\text{où } \lambda = \frac{y_2 - y_1}{x_2 - x_1} \text{ si } P_1 \neq P_2, \text{ et } \lambda = \frac{3x_1^2 + a}{2y_1} \text{ sinon.}$$

Prenons p = 23 et considérons la courbe elliptique E : $y^2 = x^3$ + x + 1 (*) défini sur Z_{23}. (Dans la notation de l'équation (*), nous avons a = 1 et b = 1). Noter que $4a^3 + 27b^2 = 4 + 4 = 8 \neq 0$, ainsi E est une courbe elliptique.

1. soient P = (3,10) et Q = (9,7). P + Q = (x_3, y_3) est calculé comme suit :

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} \in Z_{23}.$$
$$x_3 = 11^2 \text{ - } 3 \text{ - } 9 = 6 \text{ - } 3 \text{ - } 9 = 6 \equiv 17 \pmod{23}, \text{ et}$$
$$y_3 = 11(3 + 6) \text{ - } 10 = 11(9) \text{ - } 10 = 89 \equiv 20 \pmod{23}.$$
$$\text{d'où P + Q} = (17, 20).$$

2. Prenons P = (3,10). 2P = P + P = (x_3, y_3) est calculé comme suit :

$$\lambda = \frac{3(3^2+1)}{20} = \frac{5}{20} = \frac{1}{5} \in Z_{23}.$$
$$x_3 = 6^2 \text{ - } 6 = 30 \equiv 7 \pmod{23}, \text{ et}$$
$$y_3 = 6(3 \text{ - } 7) \text{ - } 10 = \text{-}24 \text{ - } 10 = \text{-}11 \equiv 12 \pmod{23}.$$

D'où 2P = (7, 12).

Remarque : Notons ϑ le point distingué de E. Il existe une loi de groupe sur les points de E telle que ϑ soit l'élément neutre. L'opposé de chaque point de E est son symétrique par rapport à l'axe des abscisses [30]. La somme de deux points P, Q finis et non opposés est donc le point de coordonnées (x_3, -y_3) où x_3, y_3 sont donnés par la relation (2.2). Lorsque l'un des points est à l'infini, ou lorsqu'ils sont symétriques l'un de l'autre, on a les identités P + ϑ = P et P + (-P) = ϑ.

2.2.5 Calcul du nombre de points d'une courbe elliptique sur un corps fini

Soit E une courbe elliptique d'équation (2.1) sur F_p (avec p premier).
Le nombre de points de E est donné par :

$$\#E = 1 + \sum_{x \in F_p} (1 + (\frac{x_3 + ax + b}{p})) \tag{2.5}$$

Si le corps de base est fini, la courbe elliptique est un groupe fini et le théorème suivant donne des renseignements très utiles sur son ordre [31].
Nous nous intéressons aux courbes elliptiques E, qui sont définies sur un corps fini F_p.
E(F_p) a au maximum p - 1 points.

Théorème 01 : (Théorème de Hasse) : Soit une courbe elliptique sur le corps fini F_p.
Le nombre de points de E vérifie :

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p} \tag{2.6}$$

D'autre part, deux éléments suffisent pour engendrer le groupe d'une courbe elliptique et bien souvent, il est même cyclique.

Exemple :
Soit p = 23 et considérons la courbe elliptique : $y^2 = x^3$ + x + 1, définie sur Z_{23} fini. (Dans la notation de l'équation (2.1), nous avons a = 1 et b = 1.) On note cela $4a^3 + 27b^2 = 4 + 4 = 8 \neq 0$, ainsi E est en effet une courbe elliptique. Les points dans E(Z_{23}) sont ϑ et les suivants : (0, 1), (0, 22), (1, 7), (1, 16), (3, 10), (3, 13), (4, 0), (5, 4), (5, 19), (6, 4), (6,19), (7, 11), (7, 12), (9, 7), (9, 16), (11, 3), (11, 20), (12, 4), (12, 19), (13, 7), (13, 16), (17, 3), (17, 20), (18, 3), (18, 20), (19, 5), (19,18).

2.2.6 Problème du logarithme discret

Soit G un groupe (noté additivement) cyclique fini d'ordre N engendré par un élément P. Soit Q un élément de G. Comme G est un groupe cyclique engendré par P, il existe un unique entier k compris entre 1 et N tel que Q = kp. Cet entier k est appelé le logarithme discret de Q en base P noté $\log_p(Q)$ [27]. Cependant, le groupe que nous utiliserons par la suite est usuellement noté additivement et nous garderons donc les notations initiales. Notons bien que kP signifie P + P + ... + P, k fois où le signe « + » est la loi du groupe G. La multiplication scalaire d'un point par un entier est l'opération de base et l'opération la plus coûteuse des protocoles basés sur les courbes elliptiques. Pour réduire le temps de calcul, on peut utiliser des algorithmes très connus d'exponentiation pour le calcul de [k]P pour un k grand. On peut également améliorer le calcul pour un k petit.

Problème du logarithme discret sur les courbes elliptiques

Les crypto-systèmes utilisant les courbes elliptiques sont basés sur l'analogie du problème du logarithme discret sur les courbes elliptiques. Soit E une courbe elliptique définie sur un corps fini F_q et soit P un point sur la courbe E. Le problème du logarithme discret sur les courbes elliptiques (noté par ECDLP, Elliptic Curve Discrete Logarithme Problem) consiste à trouver un nombre k, étant donné le point P et le point Q = kP où kP = P + P + ... + P. Actuellement, le meilleur algorithme pour résoudre l'ECDLP est selon le temps exponentiel et la taille de la clé k, contrairement aux algorithmes exponentiels connus pour factoriser les entiers les plus élevés. Ceci permet aux cryptosystèmes basés sur les courbes elliptiques d'utiliser «à sécurité équivalente» des clés beaucoup plus courtes que les crypto-systèmes asymétriques classiques comme RSA ou ElGamal [32].

2.2.7 Protocole d'échange de clés de Diffie-Hellman

Un des moyens pour sécuriser les données transitant entre l'émetteur et le récepteur est l'établissement d'une clé privée entre eux. La méthode de Diffie-Hellman [17] permet de :

1) Alice et Bob choisissent une courbe elliptique E définie sur un corps fini F_q , tel que le logarithme discret soit difficile à résoudre. Ils choisissent aussi un point P $\in F_q$ tel que le sous-groupe généré par P possédant un ordre de grande taille. (E et P sont choisis de telle manière que l'ordre soit un grand nombre premier).

2) Alice choisit un nombre entier secret a, calcule P_a = aP et l'envoie à Bob.

3) Bob choisit un nombre entier secret b, calcule P_b = bP et l'envoie à Alice.

4) Alice calcule aP_b = abP et Bob calcule bP_a = baP.

5) Alice et Bob utilisent une méthode quelconque connue pour extraire une clé secrète de abP. Par exemple, ils peuvent utiliser les derniers 256 bits de la première coordonnée de abP comme clé, ou ils peuvent hacher une des coordonnées de abP avec une fonction de hachage pour laquelle ils se sont mis d'accord.

2.2.7.1 Échange de clés

Alice et Bob se mettent d'accord (publiquement) sur une courbe elliptique E(a, b, p), c'est-à-dire qu'ils choisissent une courbe elliptique y^2 mod p = (x^3 + ax + b) mod p. Ils se mettent aussi d'accord, publiquement, sur un point P situé sur la courbe. Secrètement, Alice choisit un entier d_A, et Bob un entier d_B. Alice envoie à Bob le point d_AP, et Bob envoie à Alice d_BP . Chacun de leur côté, ils sont capables de calculer $d_A(d_B\mathrm{P}) = d_B(d_A\mathrm{P}) = (d_A\ d_B)\mathrm{P}$, qui est un point de la courbe, et constitue leur clé secrète commune.

2.2.7.2 Sécurité

Si Eve a espionné leurs échanges, elle connaît E(a, b, p), P, d_AP et d_BP. Pour pouvoir calculer $d_A\ d_B$P, il faut pouvoir calculer d_A connaissant P et d_AP. C'est ce qu'on appelle résolution du logarithme discret sur une courbe elliptique. Où, actuellement, si les nombres sont suffisamment grands, on ne connaît pas de méthode efficace pour résoudre ce problème en un temps raisonnable.

2.3 Travaux connexes

Les protocoles GDH (Groupe Diffie-Hellman) ont été premièrement présenté dans [33]. Ils constituent l'extension de l'échange de Diffie-Hellman à plusieurs participants. Les auteurs dans [34] ont proposé TGDH (Tree Groupe Diffie-Hellman) utilisant un arbre binaire afin de minimiser le nombre total des messages échangés lors du calcul de la clé du groupe. L'évaluation des performances des protocoles de groupe Diffie-Hellman ont été introduite dans [35, 36]. Dans [35], les auteurs ont évalué 5 protocoles d'accord des clés : Centralized Group Key Distribution (CKD), Burmester Desmedt (BD), Steer et al. (STR), Group Diffie-Hellman (GDH) et Tree based Group Diffie Hellman (TGDH). Ils ont conclu que TGDH donne la meilleure moyenne des performances pour les réseaux WAN. Dans [36], les auteurs ont évalué 3 protocoles d'accord des clés basés sur Diffie-Hellman : GDH, TGDH et Hypercubic Diffie-Hellman. Ils démontrent que GDH est efficace pour les petits réseaux et TGDH pour les réseaux de grande taille. Tous ces protocoles des Systèmes de Communication de Groupe sont connus par des protocoles basés sur DLP. Récemment, pour plus d'efficacité, L. Harn et al. dans [37] proposent une solution d'accord

de clés de groupe Diffie-Hellman basé sur le partage d'un secret.

Dans [38], ont démontré qu'ECC peut atteindre le même niveau de sécurité en utilisant une clé plus courte que celle de RSA (une clé de 160 bits dans ECC est équivalente à une clé de 1024 bits dans RSA). Cette méthode elle donc de plus en plus utilisée pour transmettre les clés.

Malan et al. [39] donnent le résultat d'une implémentation du protocole Diffie-Hellman basé sur les courbes elliptiques du problème du logarithme discret. Cependant, les méthodes pour le calcul du logarithme discret des courbes elliptiques sont moins efficaces que ceux du calcul du logarithme discret conventionnel et il est indiqué que plus de temps de traitements est nécessaire pour les ECC. D'où, l'évaluation des performances des protocoles basés sur ECDLP est nécessaire. Nous détaillerons dans ce qui suit les principaux protocoles proposés dans ce sens.

2.3.1 Prtoctocle GECDH

Y. Wang et al. dans [40] proposent GECDH (The Group Elliptic Curve Diffie-Hellman Protocol). C'est une extension de la famille du GDH à GDH basé sur ECDLP. Lui aussi, il est divisé en deux étapes : ascendant et descendant. L'étape du flux ascendant consiste à collecter les contributions de tous les membres. Les clés intermédiaires et la valeur cardinale sont des points de la courbe elliptique, elles ont donc une longueur double de la clé privé. L'étape du flux descendant du dernier tour, consiste à diffuser les valeurs intermédiaires à tous les membres du groupe afin de calculer la clé du groupe. La dernière valeur cardinale du membre du plus grand index M_n diffuse $n-1$ valeurs intermédiaires à tous les autres membres. Chaque membre M_i récepteur de ce message identifie sa valeur intermédiaire et la multiplie avec N_i et calcul donc la clé secrète. La clé de groupe peut être dérivée de la clé secrète.

Exemple [1] : Considérons un exemple de 4 membres du groupe. Au début le groupe est constitué des utilisateurs A et B. Ensuite deux autres utilisateurs C et D intègrent le groupe.
Les utilisateurs A et B vont échanger leurs clés : Prenons $p = 211$, $Ep = (0, -4)$ qui est équivalent à la courbe $y^2 = x^3 - 4$ et $G = (2, 2)$.
La clé privé de A est $nA = 47568$, donc sa clé publique est $P_A = 47568(2, 2) = (206, 121)$. La clé privé de B est $nB = 13525$, donc sa clé publique est $P_B = 13525(2, 2) = (29, 139)$.
La clé de groupe est calculée comme suit : L'utilisateur A envoie sa clé publique $(115, 48)$ à l'utilisateur B, ensuite B calcule la clé de groupe nB(la clé publique de A) $= 13525(206, 121) = (155, 115)$. L'utilisateur B envoie sa clé publique $(29, 139)$ à l'utilisateur A, ensuite A calcule la clé de groupe nA(la clé publique de B) $= 47568(29, 139) = (155, 115)$. La figure 2.2 illustre cela.

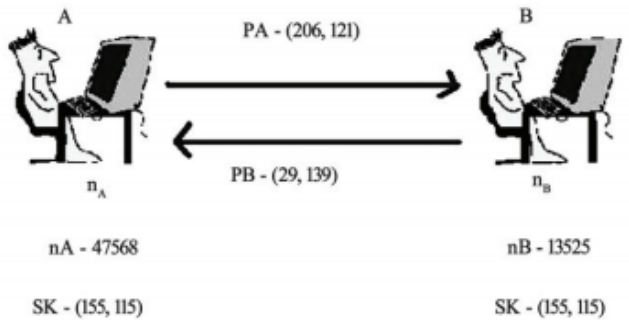

FIGURE 2.2 – Echange de cles entre A et B.

Jointure d'un membre : Lorsque l'utilisateur C veut joindre le groupe, telle que sa clé privé est $nC = 82910$. C devient contrôleur de groupe. Ensuite, le processus de mise à jour de la clé va procéder comme suit : le contrôleur précédent du groupe B envoie une clé intermédiaire sous forme : (clé publique de A \$ clé publique de B \$ la clé du groupe de A & B) = (29,139), (206,121) et (155,115). Puis, C génère une nouvelle clé de groupe nC(Clé de groupe de A & B) = $82910(155, 115) = (120, 31)$.
Maintenant, C diffuse les clés intermédiaires à A et B. Ces valeurs sont (la clé de groupe de B et C) \$ (la clé de groupe de A et C)= ((131,84) \$ (147,97)). L'utilisateur B génère une nouvelle clé de groupe : Il extrait les clés de groupe de A et C envoyées par C et calcule ensuite la nouvelle clé de groupe comme suit : nB(la clé de groupe de A & C) = $13525(147, 97) = (120, 31)$. Et c'est la même nouvelle clé de groupe calculée par C. L'utilisateur A génère une nouvelle clé de groupe : Il extrait les clés de groupe de B et C envoyées par C et calcule ensuite la nouvelle clé de groupe comme suit : nA(la clé de groupe de B & C) = $47568(131, 84) = (120, 31)$. La figure 2.3 illustre cela.

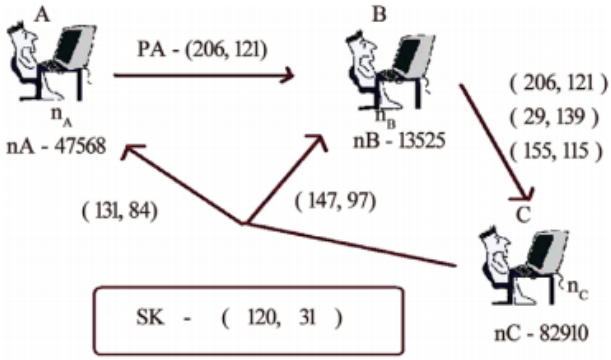

FIGURE 2.3 – Jointure d'un membre C.

La même procédure est suivie quand D veut joindre le groupe.

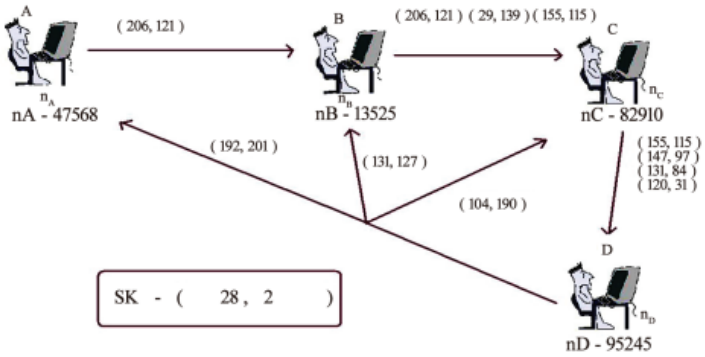

FIGURE 2.4 – Jointure d'un membre D.

Départ d'un membre : Lorsqu'un membre quitte le groupe, le contrôleur de groupe change sa clé privé. Il diffuse ensuite sa nouvelle clé publique à tous les membres restants, et une nouvelle clé du groupe sera à nouveau générée.
Soit B l'utilisateur sortant, alors le contrôleur du groupe D change sa clé privé $nD = 43297$, calcule sa clé publique et l'envoi à C et A. La nouvelle clé de groupe ainsi générée est $k = (207, 115)$. La figure 2.5 illustre cela.

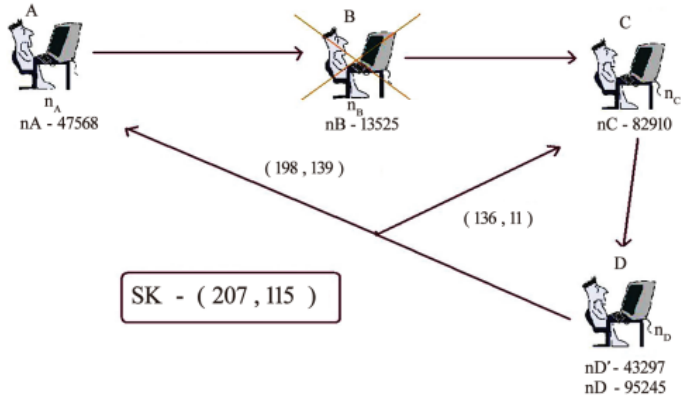

FIGURE 2.5 – Départ d'un membre B.

Départ du contrôleur du groupe : Dans ce cas, l'ancien contrôleur du groupe change sa clé privé. Ensuite, il diffuse sa nouvelle clé publique à tous les utilisateurs du groupe. La nouvelle clé du groupe sera ainsi générée. Considérons dans cet exemple le contrôleur du groupe D qui va quitter le groupe, l'ancien contrôleur du groupe C change sa clé privé $n_C = 52898$, calcule sa clé publique et l'envoie à A et B. La nouvelle clé de groupe ainsi généré est $k = (198, 139)$. De la même manière, l'utilisateur A et B calcule la même clé du groupe. La figure 2.6 illustre cela.

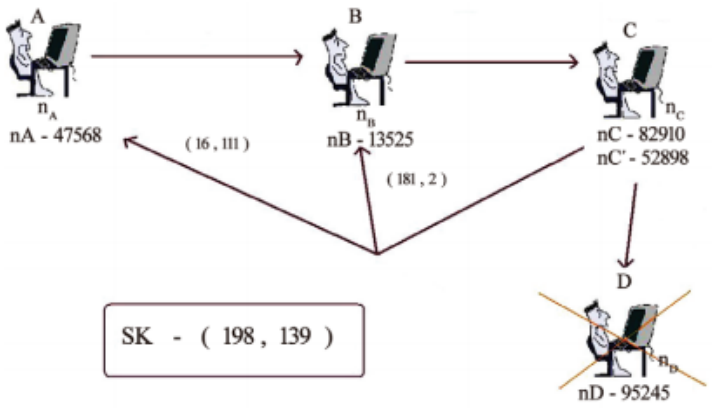

FIGURE 2.6 – Départ du contrôleur du groupe D.

2.3.2 Protocole TGECDH

Les mêmes auteurs dans [40] proposent TGECDH (Tree based Group Elliptic Curve Diffie Hellman Protocol). Elle constitue une variante du protocole TGDH basé sur ECDLP. Dans ce cas, les membres du groupe sont organisés en un arbre binaire. Les nœuds sont dénotés par $< l, v >$, où $0 \leq v \leq 2^l - 1$, chaque niveau l héberge 2^l nœuds. Chaque nœud $< l, v >$ lui est associé une clé $K_{<l,v>}$ et la clé blindée $BK_{<l,v>} = F(K_{<l,v>})$, où $F()$ est la fonction de la multiplication scalaire des points des courbes elliptiques dans le champ des nombres premiers. Supposons un nœud feuille $< l, v >$ héberge le membre M_i, le nœud $< l, v >$ aura la clé de session $K_{<l,v>}$. De plus, le membre M_i au niveau du nœud $< l, v >$ connaît toute clé dans le chemin des clés dans l'arbre allant du $< l, v >$ au $< 0, 0 >$. La réversibilité de la clé blindé $BK_{l,v}$ de chaque nœud n'est pas nécessaire. Donc, il est suffisant d'utiliser la x-coordonnée de $K_{l,v}$ comme clé blindée. A chaque fois q'un membre joigne/quite le groupe, le nœud sponsor calcule d'abord la clé de session, ensuite diffuse la nouvelle clé blindée à tout le groupe afin que les membres restants puissent générer la clé de session du groupe. La figure 2.7 représente le schéma général d'un arbre de clés.

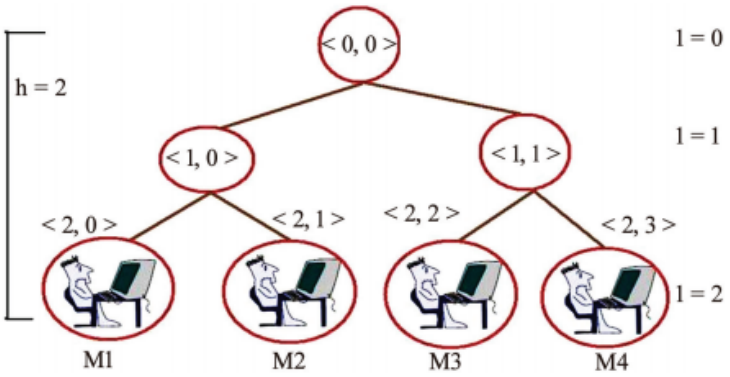

FIGURE 2.7 – Arbre de clés [1].

exemple [1] : Considérons l'exemple précédent d'un groupe de 4 utilisateurs M_1, M_2, M_3 et M_4 :

Les utilisateurs M_1 et M_2 vont échanger leurs clés : Prenons $p = 751$, $E_p = (1, 188)$ qui est équivalent à la courbe $y^2 = x^3 + \mathrm{x} + 188$ et G = (0, 376). La clé privé de M_1 est 1772, donc sa clé publique est $(290, 638)$. La clé privé de M_2 est 1949, donc sa clé publique est $(504, 163)$. La clé de groupe est calculée dans la figure 2.8, l'utilisateur M_1 envoie sa clé publique $(290, 638)$ à l'utilisateur M_2, l'utilisateur M_2 calcule leur clé de groupe en tant que PV (0,0) = X_co (PV (1,0) * PB (1,1)) et PB (0,0) = PV (0,0) * G = (540, 111). De même, l'utilisateur M_2 envois sa clé publique $(504, 163)$ à l'utilisateur M_2, puis l'utilisateur M_1 calcule la clé de groupe $(540, 111)$. Ici, le contrôleur du groupe est utilisateur M_2.

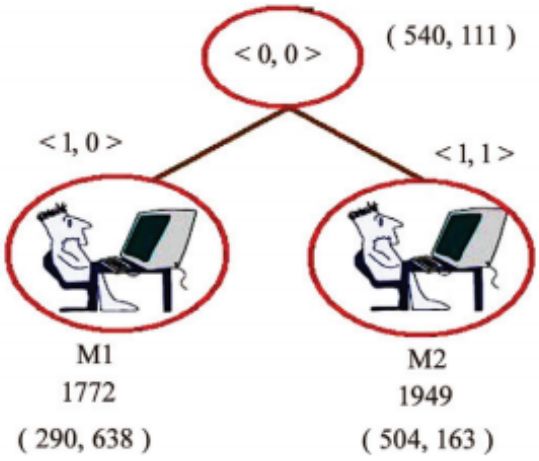

FIGURE 2.8 – L'utilisateurs M_1 et M_2 echange leurs clés.

jointure d'un membre : Lorsque l'utilisateur M_3 joint le groupe comme le montre la figure 2.9. L'ancien contrôleur de groupe M_3 modifie sa clé privée de 1949 à 2835 et passe sa clé publique

à l'utilisateur M_3. Maintenant, M_3 devient le nouveau contrôleur de groupe. Ensuite, M_3 génère la clé publique $(623, 52)$ à partir de sa clé privée 14755 et calcule la clé de groupe (664736) représenté sur la figure 2.9. M_3 envoie sa clé publique à tous les utilisateurs. Maintenant, l'utilisateur M_1 et M_2 calculent leur clé de groupe.

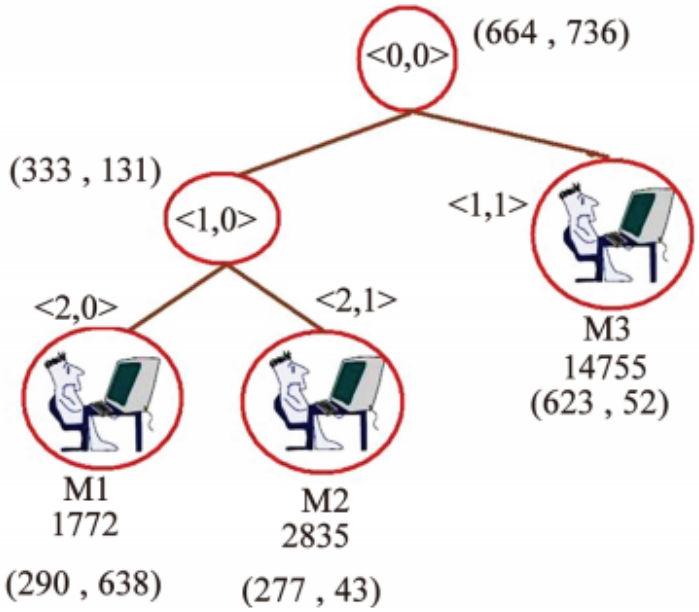

FIGURE 2.9 – Jointure d'un membre M_3.

La même procédure est suivie en rejoignant l'utilisateur M_4 comme le montre la figure 2.10.

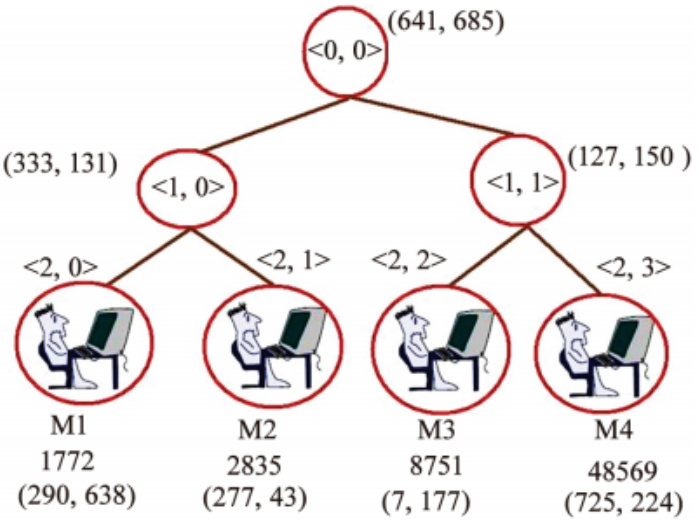

FIGURE 2.10 – Jointure d'un membre M_4.

départ d'un membre : Lorsque l'utilisateur M_3 quitte le groupe (figure2.11), le contrôleur du groupe change sa clé privée de 48569 à 98418 et la clé de groupe est recalculé (428.686). Après cela, il diffuse sa clé publique à tous les utilisateurs du groupe. Ensuite, la nouvelle clé de groupe sera générée par les utilisateurs restants.

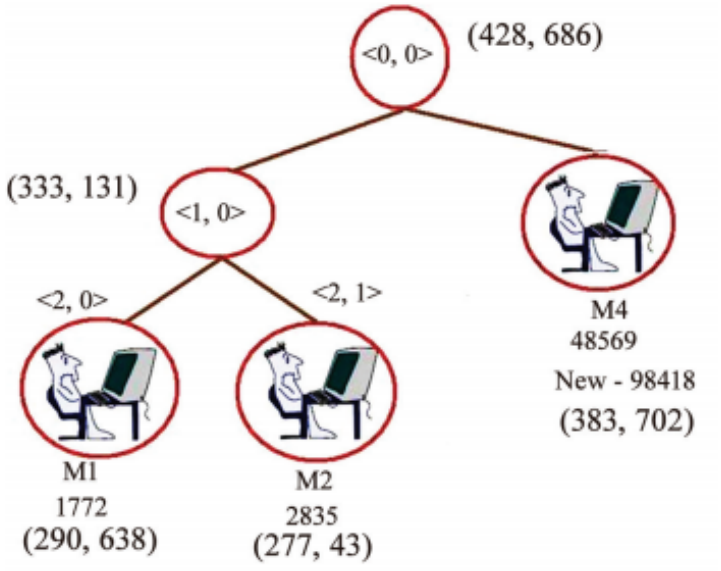

FIGURE 2.11 – Départ d'un membre.

départ du contrôleur du groupe : Quand le contrôleur quitte le groupe (figure 2.12), l'ancien Contrôleur modifie sa clé privée 8751 à 19478 et recalcule la clé de groupe (681.475). Après cela, il diffuse sa clé publique à tous les utilisateurs du groupe . Ensuite, la nouvelle clé de groupe sera générée par les utilisateurs restants.

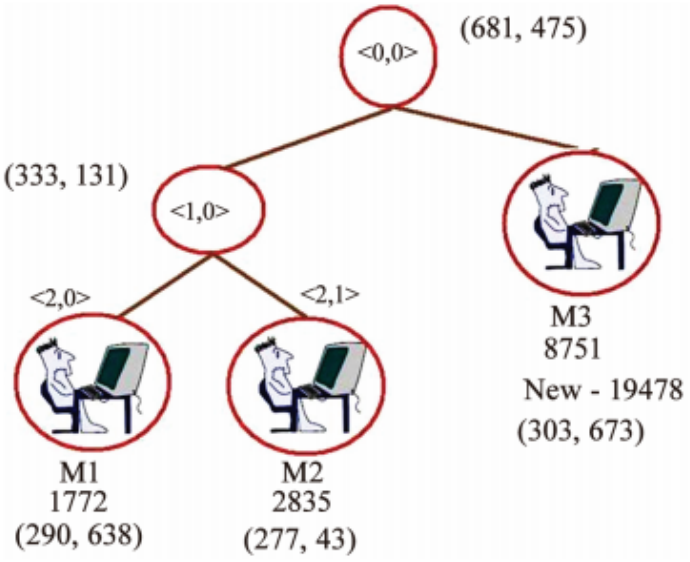

FIGURE 2.12 – Départ du contrôleur du groupe.

2.3.3 Protocole DCGKA

Récemment, Naresh et al. dans [41], proposent DCGKA (Dynamic Contributory Group Key Agreement protocol) qui est une extension de CGKA pour les groupes dynamiques, basé sur ECDH. Dans ce cas, un membre parmi ceux du groupe joue le rôle du contrôleur de groupe, qui publie publiquement les paramètres (p, a, b, P, n, h) et procède comme suit : Soient $M_1, M_2, ..., M_l, ..., M_m$ les membres du groupe, et soit M_l le contrôleur du groupe,où $1 \leq l \leq m$.

Étape 1 : Initialement, le contrôleur du groupe (GC), M_l forme $(m-1)$ groupes de deux éléments avec chaque membre M_i restant et produit $(m-1)$ clés partagés comme suit :

1. Le contrôleur du groupe M_l choisit une clé privé $x_l \in 1, 2, ..., n-1$ et génère une clé publique $X_l = [x_l]P$
2. Chaque membre du groupe M_i, tel que $i \neq l$, choisit une clé privé $x_i \in 1, 2, ..., n-1$ et génère une clé publique
$$X_i = [x_i]P, \text{ pour } 1 \leq i \leq m \ , \ i \neq l$$
3. Le G.C M_l diffuse, X_l aux membres restants du groupe et chaque membre M_i transmet X_i au G.C M_l
4. Après l'échange des clés publiques, chaque membre génère la clé partagée du style-ECDH comme suit :

$$K_{li} = [x]X_l = [x_i]([x_l]P) = [x_i x_l]P = (x_{K_{li}}, y_{K_{li}}), \text{ pour } 1 \leq i \leq m \ , \ i \neq l$$

où $x_{K_{li}}, y_{K_{li}} \in F_p$ sont respectivement des coordonnées-x et coordonnées-y de K_{li}.

Étape 2 : De cette manière, soit $x_{K_{li}}$ les $(m-1)$ clés partagées respectives entre le GC, M_l et M_i, où $1 \leq i \leq m, i \neq l$:

Le contrôleur de groupe calcule les $(m-1)$clés publiques L_i comme suit et les transmets respectivement aux M_i :

$$L_i = [\prod_{j=1, j\neq i}^{m} x_{K_{li}}]P, \text{pour } 1 \leq i \leq m, i \neq l \text{ et } j \neq l$$

Puis chaque membre M_i du groupe génère la clé de groupe K comme suit :

$$K = [x_{K_{li}}]L_i = [x_{K_{li}}][\prod_{j=1, j\neq i}^{m} x_{K_{li}}]P = [\prod_{i=1}^{m} x_{K_{li}}]P = (x_K, y_K)$$

Puisque le CG connaît toutes les clés partagées, il génère aussi la clé de groupe

$$K = [\prod_{i=1}^{m} x_{K_{li}}]P = (x_K, y_K)$$

et prend donc x_K comme clé de groupe.

2.4 Conclusion

Dans ce chapitre nous avons constaté que la cryptographie basée sur les courbes elliptiques, offre un niveau de sécurité élevée et surtout plus de performance par le fait de la taille minime des clés et l'optimisation du temps de calcul. Il est donc intéressant d'envisager l'implémentation de l'ECC (Elliptic Curve Cryptosysteme) sur les systèmes présentant des ressources en calcul et stockage de données limitées. Nous avons présenté quelques protocoles de littérature utilisant la cryptographie à courbe elliptique.

Dans le chapitre suivant nous introduisons la technique de courbe elliptique sur les protocoles d'accord de clés qui sont utilisé sur l'architecture ECGK.

Chapitre 3

Amélioration des performances d'ECGK

3.1 Introduction

Dans ce chapitre nous allons présenter l'amélioration de protocole d'accord de clé de groupe TRP en introduisant la cryptographie à courbe elliptique.

3.2 Protocole μTRP

Dans cette section, nous décrivons l'amélioration apportée au protocole d'accord de clé de groupe TRP. L'idée de base de notre amélioration est d'introduire la notion de calcul des clés et de la mémoire en utilisant les courbes elliptiques (le protocole ECDH) tel qu'il a été fait dans le protocole μSTR, μTGDH, μBD et μCLIQUE, car la multiplication scalaire d'un point est moins coûteuse que les exponentiations modulaires. Nous présentons les modifications des principaux protocoles des événements d'adhésion, particulièrement les départ des membres du protocole TRP.

3.2.1 Hypothèses

Soit E une courbe elliptique sur un corps fini F_q. Considérons G appartenant à E (F_q) en tant que premier, soit G un sous-groupe des points de E (F_q) c'est-à-dire, <G> = (O, G, 2G, ..., (t-1) G).

3.3 Comportement des membres

3.3.1 Initialisation

Dans cette phase, le protocole comporte trois tours :

Tour 1 : le leader M_l génère son secret aléatoire $r_l \in$ [1, q - 1] et diffuse un message INIT contenant : r_l * G(mod q).

Tour 2 : chaque membre M_i voulant participer génère un secret aléatoire r_i et envoie un message IREPLY au leader contenant : r_i * G(mod q).

Tour 3 : a la réception de toutes les contributions, M_l diffuse un message IGROUP contenant : { r_1 * G(mod q), ... , r_i * G(mod q), ..., r_M * G(mod q), r_l r_1 * G(mod q), ... , r_l r_i * G(mod q), ..., r_l r_M * G(mod q)}, avec i∈M\{l}.

Ainsi tout les participants peuvent calculer la clé du groupe :

$$Key = r_l(1 + \sum_{i \in M \setminus \{l\}} r_i) * G(mod\ q)$$

.

Exemple :

Supposons que M_l initie la découverte des membres du groupe et seulement 5 membres montrent un intérêt et envoient r_2 * G(mod P), r_3 * G(mod P), r_4 * G(mod P), r_5 * G(mod P) et r_6 * G(mod P) respectivement. Enfin seulement 3 s'intègrent en raison des contraintes de connectivité. Supposons que les membres qui se joignent finalement sont M_2, M_4 et M_5. Puis le leader de groupe, tel que M_l, diffuse le message suivant : {r_2 * G(mod P), r_4 * G(mod P), r_5 * G(mod P), r_1 r_2 * G(mod P),r_1 r_4 * G(mod P), r_1 r_5 * G(mod P) }. A la réception de ce message, chaque membre peut dériver r_1 * G(mod P) en utilisant son secret respectif.

Ainsi la clé $r_1(1 + r_2 + r_4 + r_5)$ * G(mod q) peut être calculée.

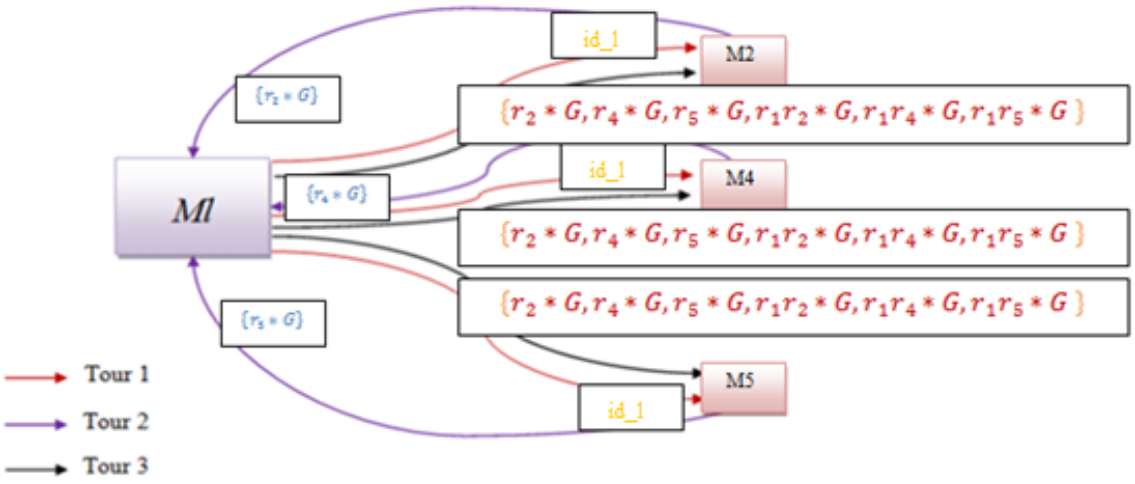

FIGURE 3.1 – Calcul de la clé de groupe.

3.3.2 Evénement d'adhésion d'un nouveau membre (Join)

Chaque nouveau membre, M_i (i $\in$ j) (j : les nouveaux membres du groupe), envoie une demande JOIN avec sa clé publique r_i * G(mod p) au groupe existant. Le leader de groupe M_l choisit une nouvelle clé secrète r_l, et envoie toutes les clés publiques au nouveau leader de groupe, $M_{l'}$. Le nouveau leader de groupe choisit un nouveau secret et calcule sa clé publique puis il diffuse un message semblable, c'est à dire, toutes les clés publiques et les clés publiques élevé à sa clé secrète.

Tour 1 : chaque nouveau membre, ($\forall$i $\in$ j), choisit une clé secrète $r_i \in$ [1, q-1] et envoie un message de jointure : {JOIN, r_i * G(mod q)}.

Tour 2 : le leader de groupe M_l choisit un nouveau secret r_l, et envoie toutes les clés publiques au nouveau leader de groupe $M_{l'}$.

Tour 3 : le nouveau leader de groupe choisit un nouveau secret et calcule sa clé publique et diffuse IGROUP = {r_l * G(mod q), r_l r_i * G(mod q)}, avec i$\in$M\{l}.

Ainsi tous les participants peuvent calculer la clé du groupe :

$$Key = r_l(1 + \sum_{i \in M \setminus \{l\}} r_i) * G(mod\ q)$$

.

Exemple : Supposons que les nouveaux membres, M_9 et M_{10} joignent le groupe des membres M_1,

M_2, M_4 et M_5 avec leurs contributions r_9 * G(mod p) et r_{10} * G(mod p) respectivement. Alors le leader précédent du groupe M_1 change son secret en $r_1^{'}$ et envoie $r_1^{'}$ * G(mod p), r_2 * G(mod P), r_4 * G(mod P), r_5 * G(mod P), r_9 * G(mod P) à M_{10}. Ce dernier produit un nouveau secret $r_{10}^{'}$ et diffuse le message suivant aux membres du groupe : {$r_1^{'}$ * G(mod P), r_2 * G(mod P), r_4 * G(mod P), r_5 * G(mod P), r_9 * G(mod P), $r_{10}^{'}$ $r_1^{'}$ * G(mod P), $r_{10}^{'}$ r_2 * G(mod P), r_{10} r_4 * G(mod P), $r_{10}^{'}$ r_5 * G(mod P), $r_{10}^{'}$ r_9 * G(mod P)}. La nouvelle clé est $r_{10}^{'}(1 + r_1^{'} + r_2 + r_4 + r_5 + r_9)$ * G(mod P).

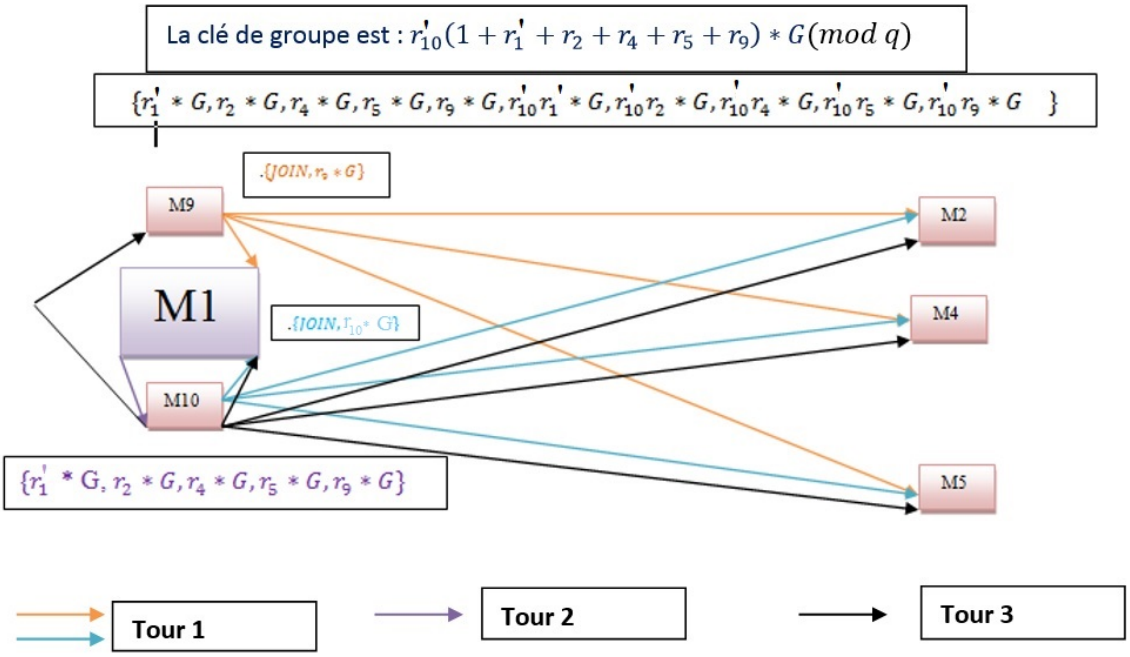

FIGURE 3.2 – Adhésion des membres.

3.3.3 L'événement de départ d'un membre (Leave)

Le départ est tout à fait semblable à l'adhésion. Quand les membres quittent le groupe, un nouveau leader de groupe est choisi parmi les membres restants, change sa contribution secrète et envoie IGROUP (Tour 2) comme un message aux membres de groupe, négligeant les contributions des membres partants.

Tour 1 : chaque membre partant, (∀i ∈ D : l'ensemble des nœuds partants), envoie un message de départ : départ, r_i * G(mod q) aux membres de groupe.

Tour 2 : le leader du groupe M_l choisit un nouveau secret r_l, et envoie toutes les clés publiques au nouveau leader du groupe $M_{l'}$.

Tour 3 : le nouveau leader de groupe choisit un nouveau secret et calcule sa clé publique et diffuse IGROUP = {r_l * G(mod q), r_l r_i * G(mod q)}, avec i∈M\{l}.

Ainsi tout les participants peuvent calculer la clé du groupe :

$$Key = r_l(1 + \sum_{i \in M \setminus \{l\}} r_i) * G(mod\ q)$$

.

Exemple : Supposons que le membre M_9 quitte le groupe des membres M_1, M_2, M_4, M_5, M_9 et M_{10}. Il envoie le message de départ avec sa contribution {Départ, r_9 * G(mod p)} . Alors l'ancien leader M_{10} change son secret en r''_{10} et envoie r''_{10} * G(mod p), r_2 * G(mod P), r_4 * G(mod P), r_5 * G(mod P) à M_1. Ce dernier produit un nouveau secret r''_1 et diffuse le message suivant aux membres de groupe : {r_2 * G(mod P), r_4 * G(mod P), r_5 * G(mod P), r''_{10} * G(mod P), r''_1 r_2 * G(mod P), r''_1 r_4 * G(mod P), r''_1 r_5 * G(mod P), r''_1 r''_{10} * G(mod P)}. La nouvelle clé est $r''_1(1 + r_2 + r_4 + r_5 + r''_{10})$ * G(mod P).

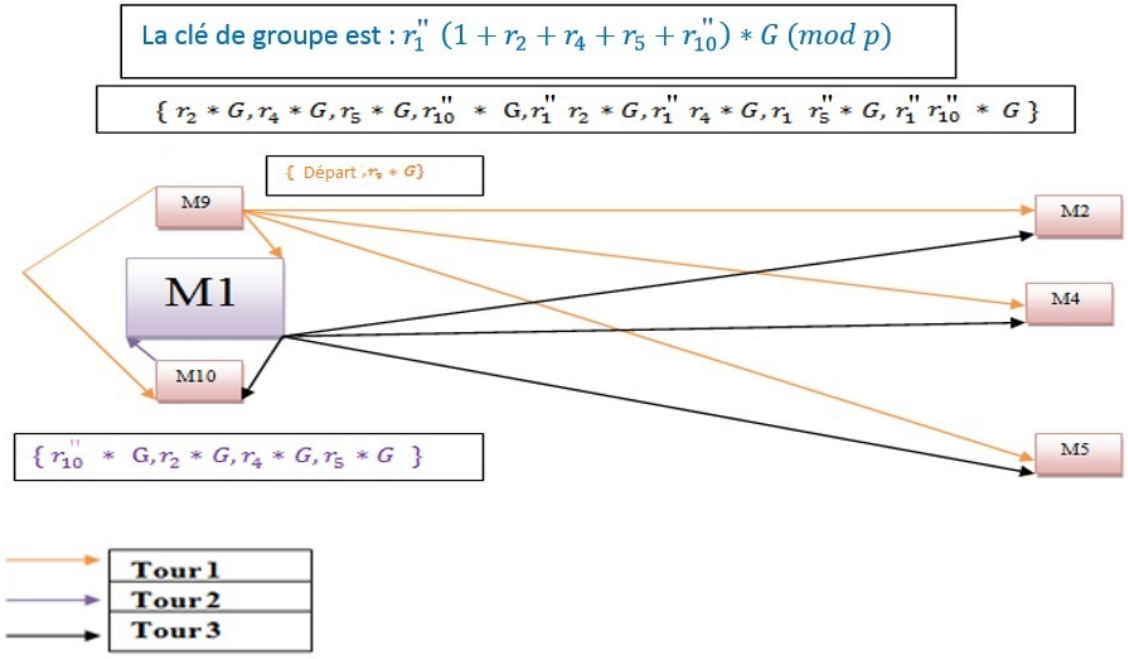

FIGURE 3.3 – Départ des membres.

Le protocole TRP qu'on a étudié est efficace seulement pour un groupe de nœuds de petite taille. Dans le cas contraire, en utilise le modèle ECGK. Le problème le plus important dans le protocole TRP est qu'il ne traite pas la tolérance aux pannes et la détection du départ involontaire d'un membre, et que c'est un problème commun de la totalité des protocoles existants dans la littérature.

Le modèle ECGK traite les problèmes du protocole TRP car il utilise deux parties : le clustering et l'établissement de clé. Dans la première partie, ECGK utilise la confiance comme critère de clustering pour traiter le problème de la tolérance aux pannes, et dans la deuxième partie « établissement de clé » ECGK utilise les protocoles TRP et Diffie-Hellman. Dans notre travail, on a rajouter la notion de cryptographie à courbe elliptique.

3.4 Détail d'établissement de clé d'ECGK

3.4.1 Exemple de clustering

Nous expliquons le modèle de clustering d'ECGK en s'appliquant sur l'ensemble des nœuds décrits dans la figure 3.4(a). Dans cet exemple, nous supposons que les nœuds ont déjà calculé une valeur de confiance pour chacun de leurs voisins directs. Les nœuds sont représentés par des cercles contenant leurs identifications et chaque nœud est marqué par les valeurs de confiance qu'il a calculés pour ses voisins.

Des seuils de confiance sont placés à $S_{min} = 0$ et $S_{max} = 0.2$. La figure 3.4(b) illustre les différentes relations TT, PT et DT qui apparaissent dans le réseau selon les valeurs de S_{min} et S_{max}. Comme montré dans la figure 3.4(b), les nœuds 4 et 13 sont les clusterheads élus selon le nombre de relations de TT qu'ils ont avec leurs voisins. Chacun des nœuds 13 et 4 met son identité dans le champ CH de ses balises et place le champ Hop_CH de ses balises à 0. Puis, les nœuds 1, 7 et 10 (resp. 2, 8 et 9) qui partagent une relation de TT avec le clusterhead 13 (resp. 4) forme le noyau (figure 3.4(b)).

Les nœuds 1, 7 et 10 (resp. 2, 8 et 9) définissent les champs de CH et Hop_CH de leur balises à 13 et 1 (resp. 4 et 1), respectivement. Les nœuds 14, et 15 tous deux ont des relations de TT avec le nœud 2. Ainsi, ils joignent le cluster 4, ils sont attachés au membre de noyau 2 (figure 3.4(c)). Ils mettent à jour leurs CH et champs de Hop_CH en conséquence. Pendant les derniers étapes, les nœuds 12 et 5 (resp. 3) lesquels partagent une relation de PT avec un nœud appartenant au cluster 4 (resp. 13), joignent ce cluster et mettent à jour leurs CH et champs de Hop_CH de leurs balises. Les arcs orientés dans les figures de l'exemple illustrent l'arbre construit pendant le processus de clustering. Chaque fois qu'un nœud joigne un cluster, il choisit un parent appartenant a ce cluster avec qui il aura le chemin le plus court au clusterhead. Chaque membre de noyau, y compris le clusterhead est responsable de l'ensemble de membres de périphérie qui font partie du sous-arbre conduit par ce membre de noyau. Notons que les nœuds 6 et 11 ne partagent aucune relation de TT ou de PT avec d'autres nœuds dans le réseau. Ainsi, l'algorithme de clustering ne les a pas inclus dans les clusters obtenus, même si ils font partie des nœuds

autorisés et authentifiés dans le groupe des membres.

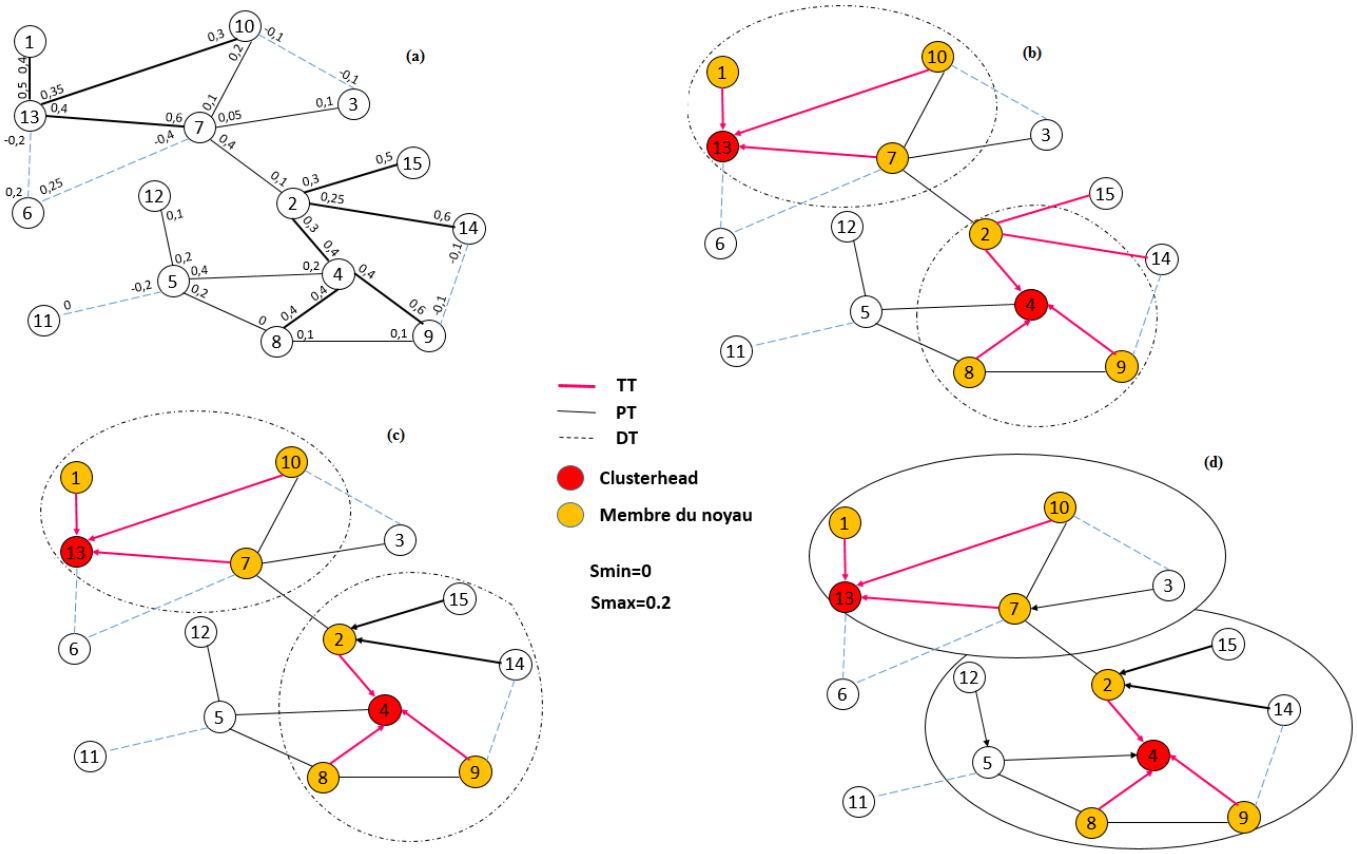

FIGURE 3.4 – Exemple de clustering.

3.4.2 Le processus de gestion de clé de groupe

Selon le modèle ECGK, les membres de noyau ainsi que le clusterhead forment une zone d'accord de clé. Ceci signifie que tous les membres de noyau contribuent au calcul du TEK du cluster. Chaque cluster c maintient une clé de chiffrement du trafic TEK_c qui sert à chiffrer/ déchiffrer des données de multicast de tous les membres du cluster. TEK_c est calculée par des membres de noyau en utilisant un protocole d'accord de clé. La périphérie est une zone de distribution de clé. Cela signifie que les membres de périphérie obtiennent la TEK du cluster par le membre du noyau auquel ils sont liés, sans pour autant pouvoir participer au calcul de cette TEK.

Pour distribuer TEK_c dans le cluster, c'est à dire, dans la périphérie du cluster, une clé principale de chiffrement (KEK) est exigée. Ainsi, chaque membre q de noyau partage une KEK : KEK_q, avec les membres de périphérie qui sont dans le sous-arbre du cluster enraciné au membre de noyau q. KEK_q sert à communiquer TEK_c à ces membres de périphérie. La communication sécurisée entre les clusters est assurée en calculant une TEK entre chaque paire de cluster adjacent.

3.4.3 La gestion de clé d'Intra-cluster

Établissement de la clé

Les membres de noyau du cluster c calculent TEK_c en utilisant le protocole TRP présenté dans [6]. TRP a besoin seulement de deux Tours de communication pour calculer une clé du groupe. Les membres de noyau opèrent comme suit :

Tour 1 : Chaque membre de noyau i choisit une clé secrète et envoie sa clé publique $b_{ri} = r_i$ * G(mod p) au clusterhead.

Tour 2 : Le clusterhead l multiplie la clé publique de chaque membre de noyau à son propre secret r_l et diffuse à ses voisins $r_i\ r_l$ * G(mod p) a tous i $\in$ M\{l}, où M est l'ensemble des membres du noyau. Chaque membre de noyau enlève son secret du $r_i\ r_l$ * G pour obtenir le r_l * G. Tous les membres de noyau calculent la même clé de cluster :

$$TEK_c = r_l(1 + \sum_{i \in M \setminus \{l\}} r_i) * G(mod\ p)$$

.

Une fois que TEK_c est calculée, chaque membre q de noyau distribue à ses membres de périphérie la TEK_c chiffrée avec KEK_q. En fait, chaque membre q de noyau d'un cluster c partage une clé KEK_q de chiffrement avec tous les membres de périphérie qui dépendent de lui. KEK_q est également calculée pendant le modèle de clustering comme suit :

- chaque fois qu'un nœud i de périphérie joigne un cluster c, il choisit une clé secrète r_i et envoie sa clé publique r_i * G(mod p) dans un message de JOINTURE, à son parent dans le cluster comme contribution pour calculer KEK_q. Le parent du nœud i transmettent la contribution vers le haut dans l'arbre de cluster jusqu'à ce qu'il atteigne l'élément de base (membre de noyau).
- le membre de noyau exécute le tour 2 du protocole TRP qui lui permet de calculer une clé avec tous les membres de périphérie qui sont attachés à lui.

3.4.4 La gestion de clé d'Inter-cluster

Il n'y a aucune clé globale de chiffrement du trafic pour le groupe. Les différents clusterheads assurent la communication entre les clusters. Chaque clusterhead calcule une clé de chiffrement du trafic avec le clusterhead de chaque cluster adjacent en utilisant le protocole authentifié d'échange de clé de Diffie-Hellman [5] ; comme suit :

— Chaque clusterhead h produit un secret r_h et envoie sa clé publique r_h * G(mod p) dans un message de Contact_Adj_CH en bas dans son arbre de cluster.

— Lorsque le message arrive à une passerelle, il est transmis à la passerelle correspondante du cluster adjacent si le lien entre les deux passerelles est un lien de TT ou PT.

— Le message va alors vers le haut au clusterhead du cluster adjacent.

— Chaque paire de clusterhead x et y peut alors calculer une $TEK_K = r_x\ r_y$ * G(mod p) selon le protocole d'accord de clé de Diffie-Hellman.

3.4.5 Exemple d'illustration

Dans cette section, nous illustrons le modèle de la gestion de clé, ainsi que le modèle de mise à jour de la topologie (départ/ jointure) sur l'exemple représenté dans la figure 3.5(d) (exemple utilisé pour illustrer l'algorithme de clustering). Dans l'exemple, nous avons deux clusters C_{13} et C_4. Ainsi, nous avons besoin de deux TEKs, une TEK par cluster : TEK_{13} et TEK_4. Comme illustré dans la figure 3.5(a), le cluster C_{13} contient seulement un membre de périphérie : nœud 3. Ce dernier est attaché au membre 7 de noyau et partage avec lui une KEK : KEK_7. Le cluster 4 contient quatre membres de périphérie : 5, 12, 14 et 15. Les nœuds 5 et 12 sont attachés au clusterhead et partages avec lui une KEK : KEK_4. Les nœuds 14 et 15 sont attachés au membre 2 de noyau et partages avec lui une KEK : KEK_2. Enfin les deux clusterheads partagent une clé commune : K_{4-13}.

Pour calculer ces clés, les nœuds procèdent comme suit :

— Le nœud 13, le clusterhead de C_{13} produit deux secrets : une contribution à TEK_{13} et une contribution à K_{4-13}.

— Le nœud 4, le clusterhead de C_4, produit trois secrets : une contribution à TEK_4, une contribution à KEK_4 et une contribution à K_{4-13}.

— Chacun des nœuds 1, 7 et 10 produit sa contribution à TEK_{13}. De même, les nœuds 2, 8 et 9 produisent leurs contributions à TEK_4. Nœud 2 (resp. 7) produit également une contribution à KEK_2 (resp. KEK_7).

— Les membres 14 et 15 de périphérie (resp. 5 et 12) produisent leurs contributions à KEK_2 (resp. KEK_4).

— le membre 3 de périphérie produit sa contribution à KEK_7.

Supposons maintenant que le nœud 9 quitte le groupe (figure 3.5(b)). Le nœud 9 envoie un message de départ au clusterhead. TEK_4 doit être mise à jour. Cependant, avant de lancer cette

opération le nœud 4 met à jour le champ de TT_edge de ses balises : il a perdu une relation de TT et par conséquent n'est plus le nœud avec le nombre le plus élevé de relations de TT parmi ses voisins. Le nœud 2 est le nouveau clusterhead. Il produit une contribution a la nouvelle TEK : TEK_2. Chaque membre de noyau, à savoir les nœuds 4, 14 et 15 produisent une contribution à TEK_2 et l'envoie au nouveau clusterhead. Le nœud 4 envoie la nouvelle TEK aux nœuds 5 et 12 chiffrée avec KEK_4. Le nœud 2 envoie un message de Contact_Adj_CH et calcule une clé commune K_{2-13} avec le nœud 13.

La figure 3.5(c) illustre l'addition d'un nouveau membre de groupe, à savoir le nœud 16, qui partage une relation de TT avec le nœud 1. En recevant le message de JOINTURE du nœud 16, le nœud 1 envoie un NEW_TEK_RQ au clusterhead. Ce dernier produit une nouvelle contribution et exécute le deuxième Tour de TRP. Tous les membres de noyau peuvent alors calculer la nouvelle TEK et la distribuer aux membres de périphérie. Le nœud 1 calcule également une KEK : KEK_1 avec le nœud 16. Il chiffre la nouvelle TEK avec KEK_1 et l'envoie au nœud 16.

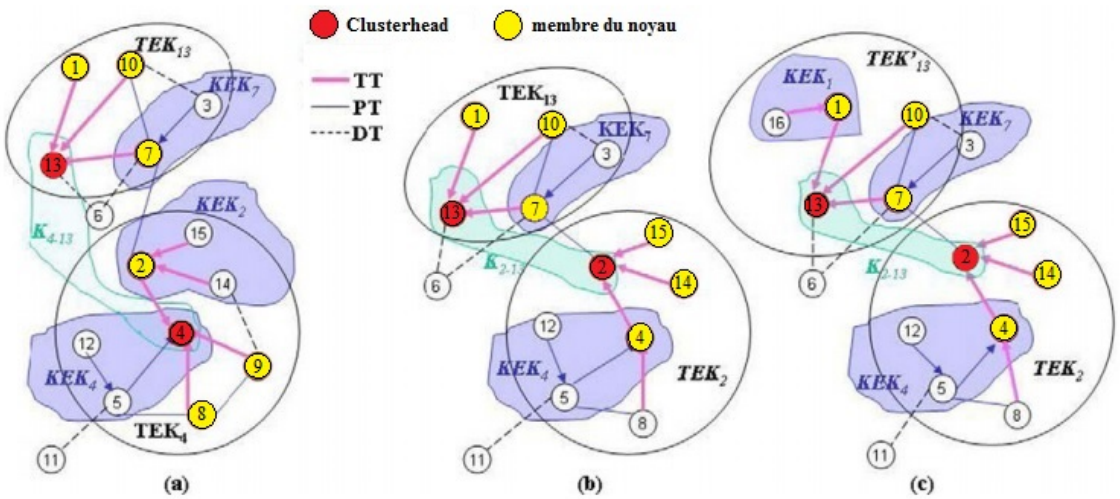

FIGURE 3.5 – Exemple de calcul des clés.

3.5 Évaluation

Nous évaluons dans cette section notre solution (ECGK) en termes d'optimalité : de coût de la communication, de coût de calcul et de coût de mémoire.

3.5.1 Discussion sur l'optimalité de coût de la communication

Le protocole μTRP est particulièrement adapté au réseaux mobiles Ad-Hoc, il est efficace en terme de nombre de tours (seulement trois tours). De plus, ses deux premiers tours peuvent

être exécutés parallèlement avec la de clusterisation du groupe (dans notre cas ECGK), comme démontré dans l'événement jointure/ départ d'un membre de groupe, ce qui permet de réduire le coût de communication et également le temps d'exécution.

3.5.2 Discussion sur l'optimalité de coût de calcul

Etant donné que notre protocole d'accord de clé μTRP s'appuie sur la théorie de ECC lors du calcule de la clé. Et comme vu précédemment, ECC utilise les multiplications scalaires de points, qui sont nettement moins coûteuses que les exponentiations modulaires utilisées dans la version originale du TRP. La cryptographie basée sur les courbes elliptiques, offre une performance optimale du temps de calcul.

3.5.3 Discussion sur l'optimalité de coût de mémoire

Notre protocole étant basé sur ECC, et comme vu précédemment le calcule de clé avec des protocole basée sur ECC offre une optimisation de la taille de la clé (compression), ainsi μTRP minimise le coût de stockage (coût de mémoire).

3.6 Conclusion

Dans ce chapitre nous avons amélioré les deux protocoles : TRP et ECGK. Cette amélioration consiste à remplacer la cryptographie à clé publique par ECC. Cette dernière offre un niveau de sécurité et surtout une performance élevée. Par le fait de la taille minime des clés, et l'optimisation du temps de calcul. L'utilisation d'ECC se révèle très intéressant pour les systèmes présentant des ressources limitées en calcul et stockage de données comme les appareils mobiles.

Dans le chapitre suivant nous allons présenter les résultats de simulation que nous avons obtenus.

Chapitre 4

Réalisation et Simulation

4.1 Introduction

Dans ce chapitre, nous allons passer à la phase finale de notre travail, en l'occurrence la simulation. Nous commencerons par présenter l'interface du simulateur que nous avons conçus à cet effet, ensuite nous donnerons un aperçu sur les paramètres de simulation que nous avons pris en considération. Enfin nous donnerons les résultats de simulation obtenus du protocole ECGK, ainsi que μTRP.

4.2 Interface de simulateur

Nous présentons l'interface du simulateur ECGK que nous avons développés en java. L'interface se compose d'une zone de simulation, un panneau supérieur qui contient les boutons qui permettent de choisir l'action à exécuter, et un panneau droit qui contient les paramètres de simulation. Comme montré dans la figure 4.1.

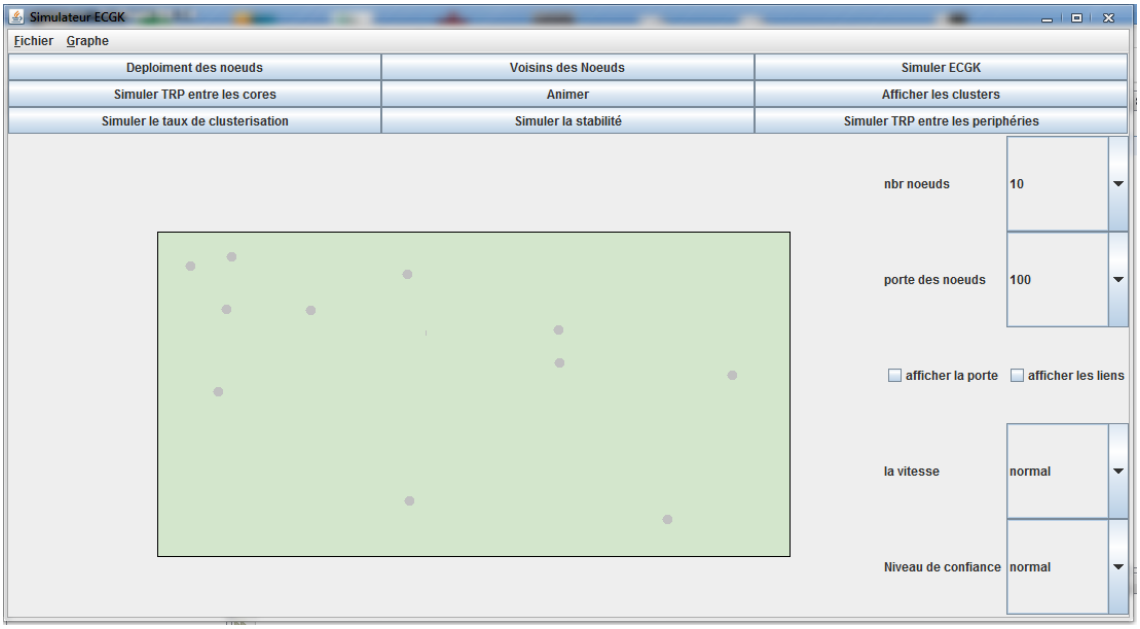

FIGURE 4.1 – Interface du simulateur.

4.2.1 Panneau supérieur

— Déploiement des nœuds : ce bouton permet de déployer un nombre de nœuds défini dans une zone de simulation de $600 * 300\ m^2$.

— Voisins des nœuds : ce bouton permet d'afficher les voisins de chaque nœud.

— Animer : permet de rendre les nœuds mobiles, suivant le modèle de mouvement RWP, dans la zone de simulation.

— Simuler ECGK : exécute les trois phases du protocole de clustering ECGK et affiche le résultat dans la zone de simulation.

— Afficher les clusters : affiche la liste des clusters (clusterheads, membres de noyau, et membres de périphérie) obtenus lors de la simulation du protocole ECGK.

— Simuler TRP entre les membres de noyau : permet de simuler le protocole d'accord de clés de groupe TRP pour les membres du noyau de chaque cluster (calcul du nombre de messages échangés lors du calcul de la TEK).

— Simuler TRP entre les membres de périphérie : permet de simuler le protocole d'accord de clés de groupe TRP pour les membres de périphéries de chaque cluster (calcul du nombre de messages échangés lors du calcul de la KEK).

— Simuler le taux de clusterisation : permet de calculer le taux de clusterisation du protocole ECGK selon les différents paramètres choisis,et ce pendant 60 s.

— Simuler la Stabilité : permet de simuler la stabilité du protocole ECGK. c'est-à-dire le changement que peut subir les clusters lors de la mobilité des nœuds, et ce pendant 60 s.

4.2.2 Panneau droit

— Nombre de nœuds : permet de choisir le nombre de nœuds (10, 20, 50, 100 et 200) qui seront placés aléatoirement dans la zone de simulation.

— Portée des nœuds : permet de choisir la portée des nœuds (50 m, 100 m et 300 m).

— Vitesse : permet de choisir la vitesse de mobilité des nœuds (5 m/s, 10 m/s et 15 m/s).

— Niveau de confiance : permet de choisir le niveau de confiance des nœuds méfiant(S_{mix}=-0,2 et S_{max}=0), normal($S_{min} = 0$ et $S_{max} = 0,2$) et confiant($S_{min} = 0,2$ et $S_{max} = 0,5$).

— afficher la portée : affiche la portée choisie des nœuds représentée par un cercle où le nœud représente son épi-centre.

— afficher les liens : affiche les différents liens existants entre les nœuds qui sont à portée, dans le cas de simulation du protocole ECGK on affiche les liens de confiances entre les membres des clusters.

4.2.3 Zone de simulation

La zone de simulation est une zone de $600 * 300$ m^2 où les nœuds y sont déployés de manière aléatoire, elle représente aussi la zone de mobilité des nœuds, aussi elle affiche les différents clusters obtenus après simulation du protocole ECGK.

4.3 Simulation

Nous avons utilisé JAVA afin de simuler le protocole de clustering ECGK pour évaluer ses performances. Dans cette section, nous présentons un aperçu de notre modèle de simulation et certains des résultats que nous avons obtenus.

4.3.1 Profil de simulation

Paramètres de simulation	Valeur
Zone de simulation	$600 * 300m^2$
Protocole de clustering	ECGK
Nombre de nœuds	10, 20, 50, 100
Placement des nœuds	Aléatoire
Vitesse	5 m/s, 10 m/s, 15 m/s
Porté	50 m, 100 m, 300 m
Niveau de confiance	Méfiant, Normal, Confiant
Modèle de mouvement	Aléatoire

TABLE 4.1 – Paramètres de simulation

4.3.2 Simulation et résultat

La figure 4.2 illustre un réseau de 50 nœuds déployés aléatoirement dans une surface de 600 $*$ 300 m^2. Chaque nœud du réseau a une portée de signal de 50 m.

Nous illustrons dans la figure 4.3 la couverture de portée du signal de chaque nœud, une arête entre deux nœuds veut dire qu'ils sont à portée l'un de l'autre.

La figure 4.4 montre le résultat de clustering après application du protocole ECGK, les nœuds rouges représentent les clusterheads, les nœuds bleus sont des membres de noyau, les nœuds verts sont des membres de périphérie et les nœuds en gris ne font partie d'aucun cluster. La figure 4.5 montre les différents liens de confiances entre les membres des clusters (bleu : TT et gris : PT)

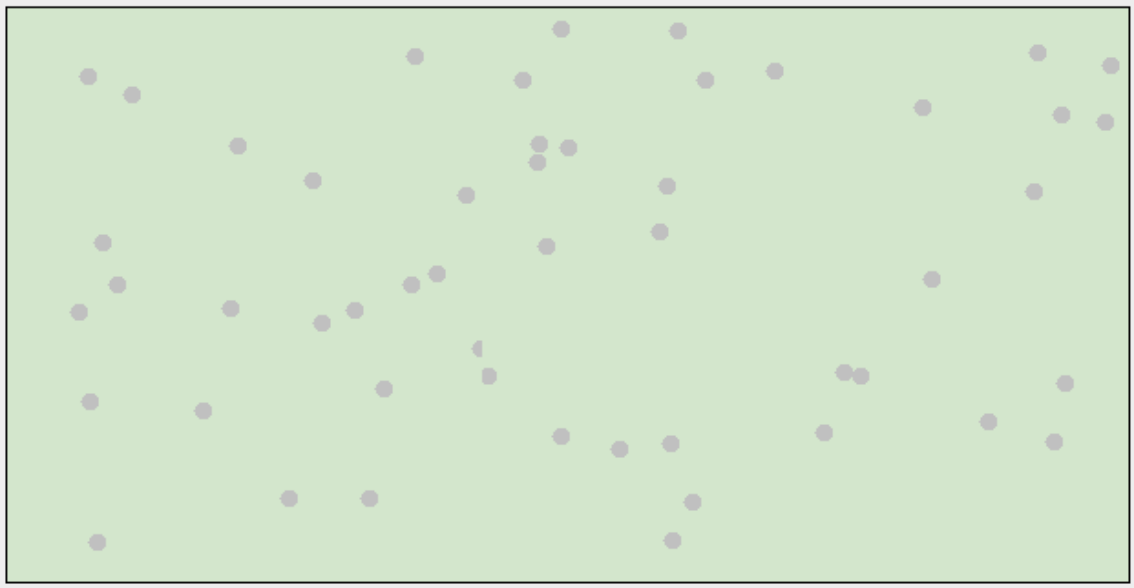

FIGURE 4.2 – Déploiement aléatoire de 50 nœuds.

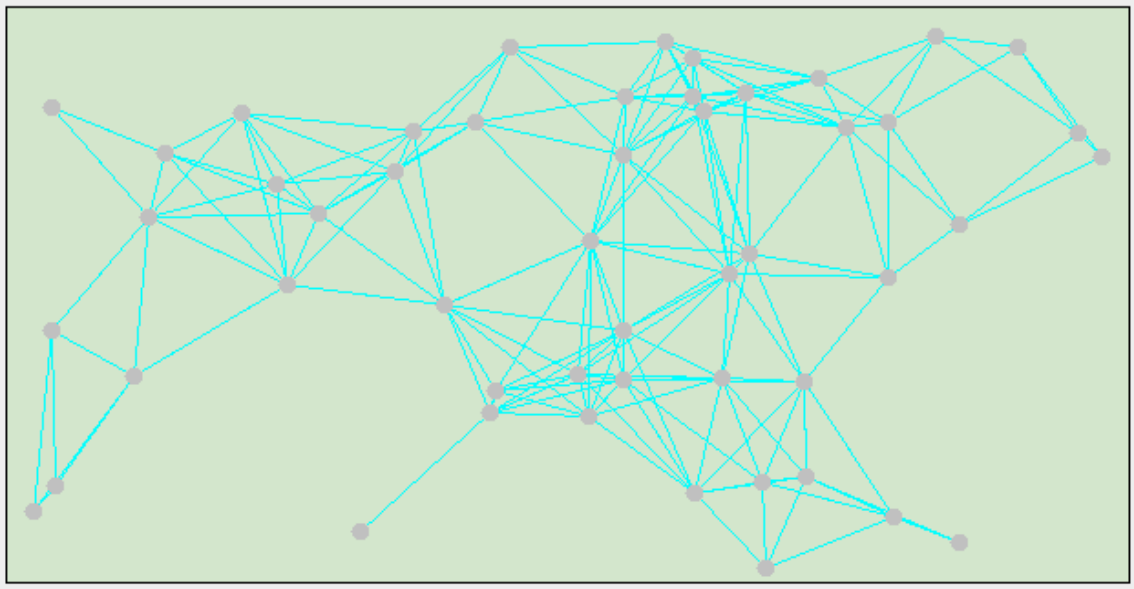

FIGURE 4.3 – Découverte du voisinage.

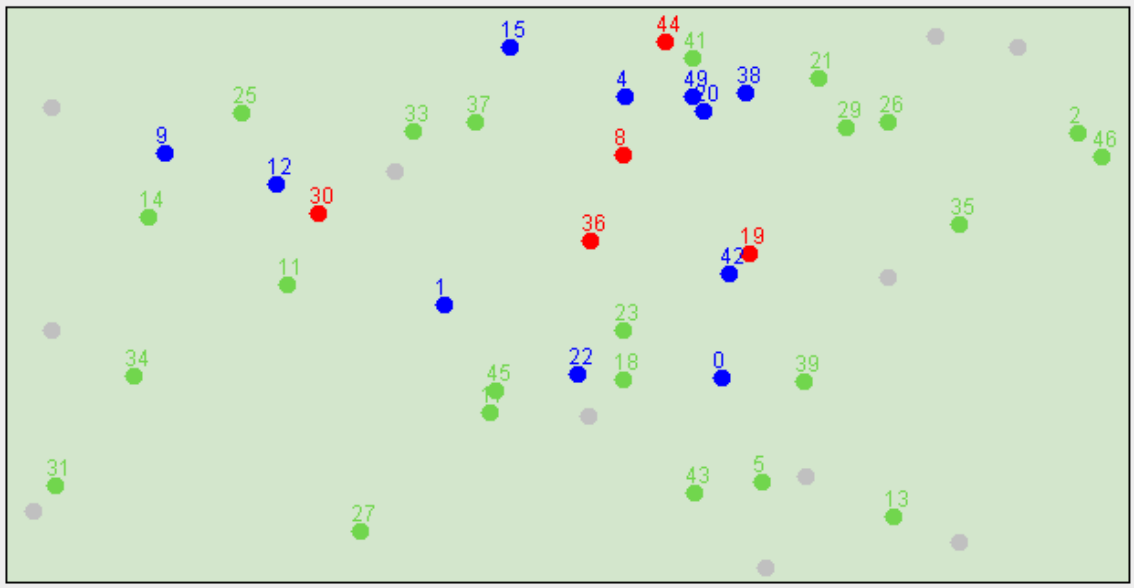

FIGURE 4.4 – Résultat de clustering du protocole ECGK.

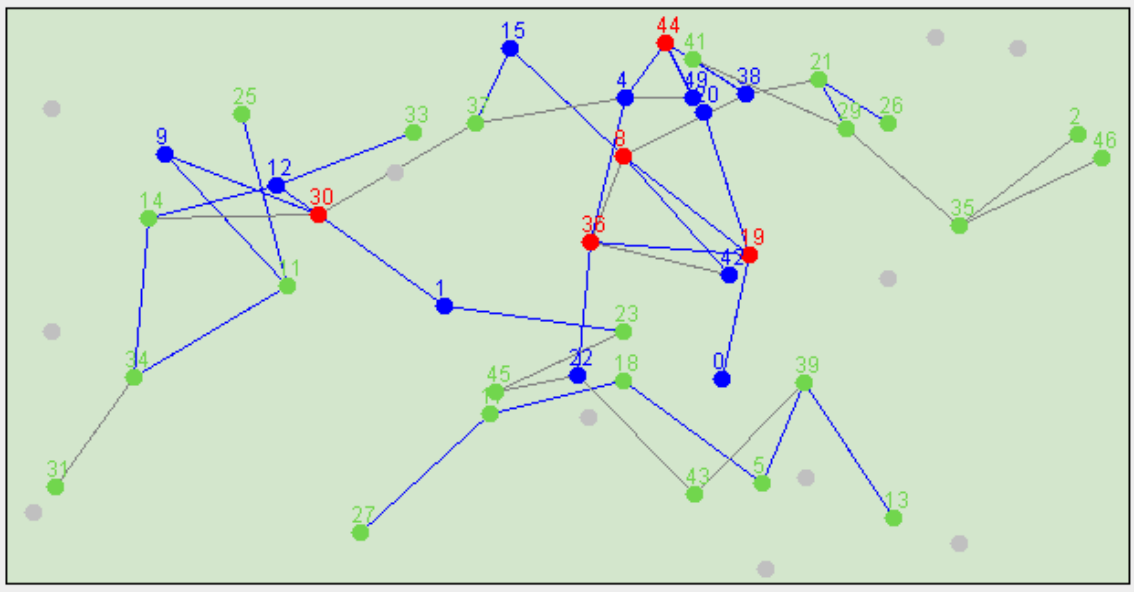

FIGURE 4.5 – Résultat de clustering du protocole ECGK avec liens de confiance.

La figure 4.6 ainsi que la figure 4.7 illustrent l'évolution du taux de clustering ainsi que la stabilité des clusters en fonction du niveau de confiance : méfiant, normal et confiant, pendant 60 s. Elles montrent que plus le niveau de confiance est bas (confiant) plus le taux de clustering est élevé, et moins les clusters sont stables.

Taux de clustering : méfiant : entre 63 et 90, normal : entre 80 et 97, confiant : entre 90 et 100. Ce résultat est dû au fait que lorsque le niveau de confiance est élevé (méfiant), le protocole de

clustering sera plus exigeant vis-à-vis des nœuds lors de leurs inclusions dans un cluster, ce qui implique qu'un nœud a une faible probabilité de faire partie d'un cluster et inversement lorsque le niveau de confiance est bas.
Stabilité des clusters : Lorsque le niveau de confiance est bas, un nœud aura un nombre élevé de valeurs de confiance partagées avec d'autre nœuds, ce qui peut expliquer le fait que les clusters sont moins stables lors de la mobilité des nœuds ; par exemple, un membre de périphérie peut devenir voisin avec un clusterhead avec qui il partage un TT ce qui induira le nœud à devenir membre du noyau. Inversement lorsque le niveau de confiance est bas.
Notons que lors de la simulation on a pris 50 comme nombre de nœuds et 100 m comme porté des nœuds.

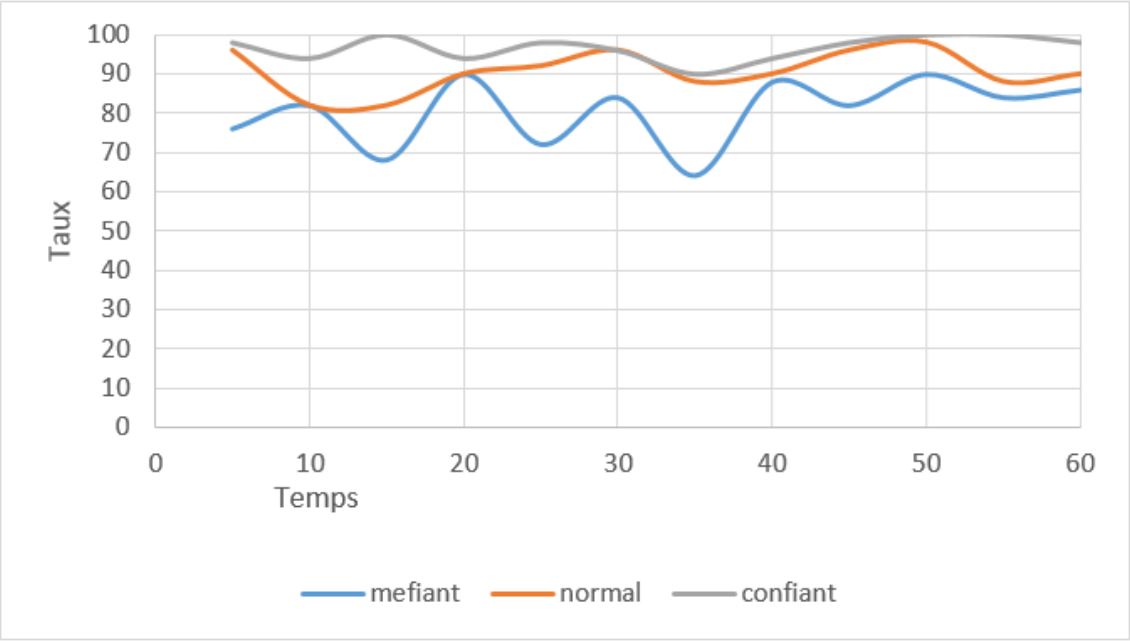

FIGURE 4.6 – Évolution du taux de clustering en fonction du niveau de confiance.

Nous illustrons dans la figure 4.8 l'évolution du taux de clustering en fonction du nombre de nœuds : 10 nœuds, 20 nœuds, 50 nœuds et 100 nœuds pendant 60 s. Nous constatons qu'avec l'augmentation du nombre de nœuds il résulte un taux de clustering élevé. Ceci est dû en premier lieu au fait que plus le nombre de nœuds est important plus un nœud aura un nombre important de voisins, ainsi il aura une forte probabilité d'avoir un lien de confiance avec au moins un nœud faisant partie d'un cluster, et par conséquent le nœud aura plus de chance de faire partie d'un cluster. En deuxième lieu plus le nombre de nœuds est important plus la couverture de voisinage des nœuds sera élevé par conséquent il y aura moins de nœuds isolés.

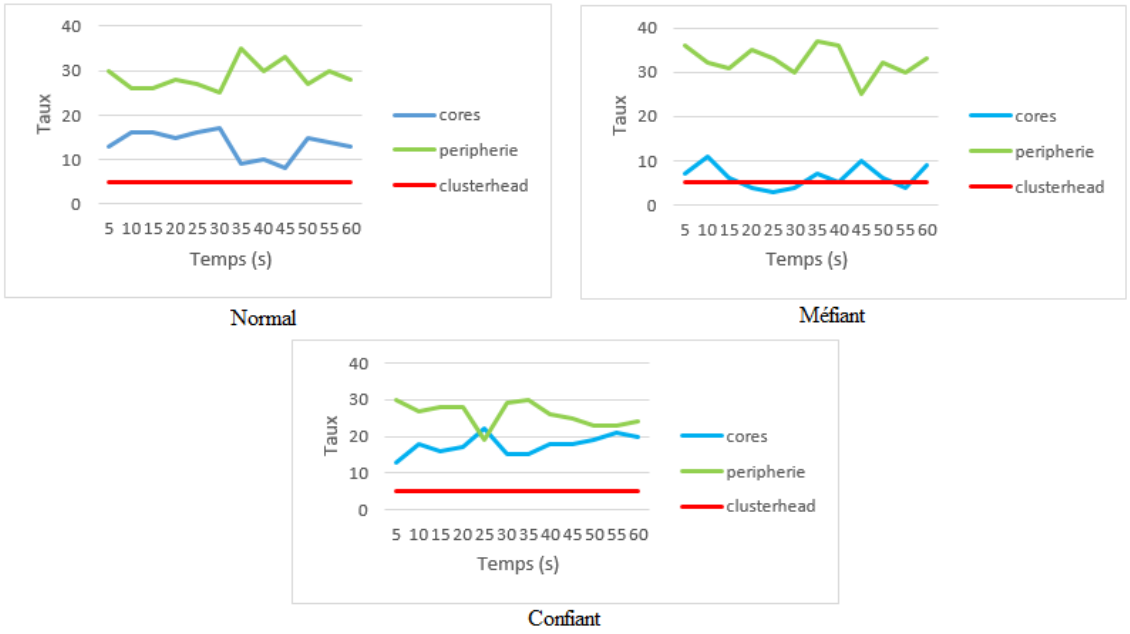

FIGURE 4.7 – Stabilité des clusters en fonction du niveau de confiance.

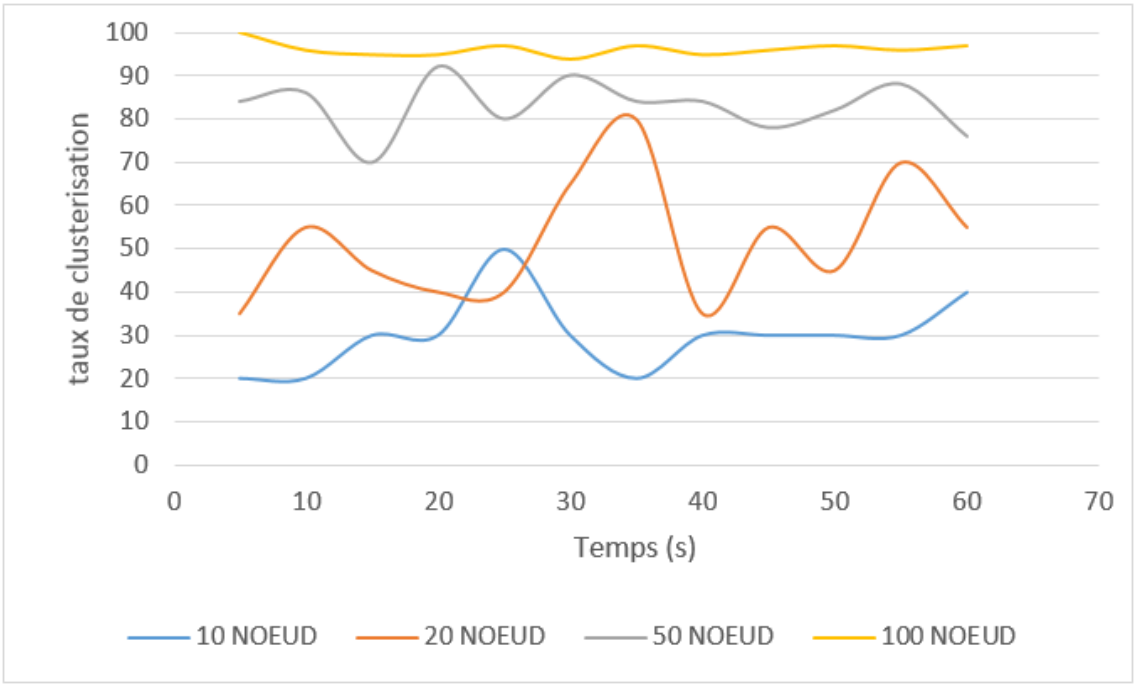

FIGURE 4.8 – Évolution du taux de clustering en fonction du nombre de nœuds.

La figure 4.9 illustre la stabilité des nœuds après clusterisation pendant un temps de 60 s, sur différentes portées 100 m et 300 m.
Nous constatons que lorsque la portée est égale 100 m le nombre des membres de périphérie est plus élevé que le nombre des membres de noyau. Cela est dû à la faible portée des nœuds, en effet une faible portée induit à un nombre moindre de voisins, ce qui implique qu'un nœud a peu de possibilité d'avoir un lien de confiance (TT) avec un clusterhead donc peu de chance d'être membre du noyau. Aussi un nœud faisant partie du noyau d'un cluster a une faible probabilité de le rester à cause de la mobilité des nœuds, Ainsi la topologie des clusters change fréquemment et par conséquents le niveau de stabilité des clusters sera faible.

Lorsque la portée est égale à 300 m, le nombre des membres de noyau est plus grand que le nombre des membres de périphérie, car quand la portée est grande la probabilité qu'un nœud ait un lien de confiance avec un clusterhead est élevé donc la possibilité de faire partie du noyau d'un cluster est aussi élevé. Aussi un nœud faisant partie du noyau d'un cluster a une forte probabilité de le rester, même lors de la mobilité. Ainsi la topologie des clusters change rarement et par conséquent le niveau de stabilité des clusters sera élevé.

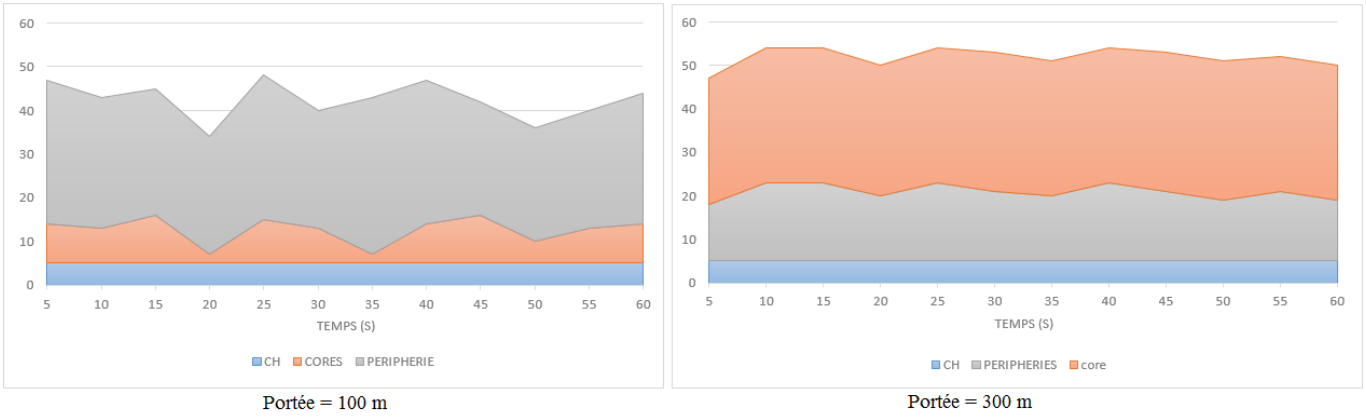

FIGURE 4.9 – Stabilité des nœuds après clusterisation où la portée est 100 m et 300 m .

La figure 4.10 illustre le nombres de messages échangés en fonction de la portée pour les trois niveau de confiance. Nous constatons que le niveau de confiance (valeurs de S_{min} et S_{max} dans l'initialisation du protocole TRP) a une influence sur le nombre de messages échangés lors de l'établissement de clés avec le protocole TRP. En effet, plus le niveau de confiance est élevé moins l'étape d'établissement de clés sera coûteuse en terme de messages échangés, qui peut être expliqué par le fait que lorsque le niveau de confiance est élevé, le taux de clusterisation est bas, donc un nœud aura peu de voisins dans le cluster, et donc une diffusion (étape 2 du TRP) sera moins coûteuse (en nombre de messages).

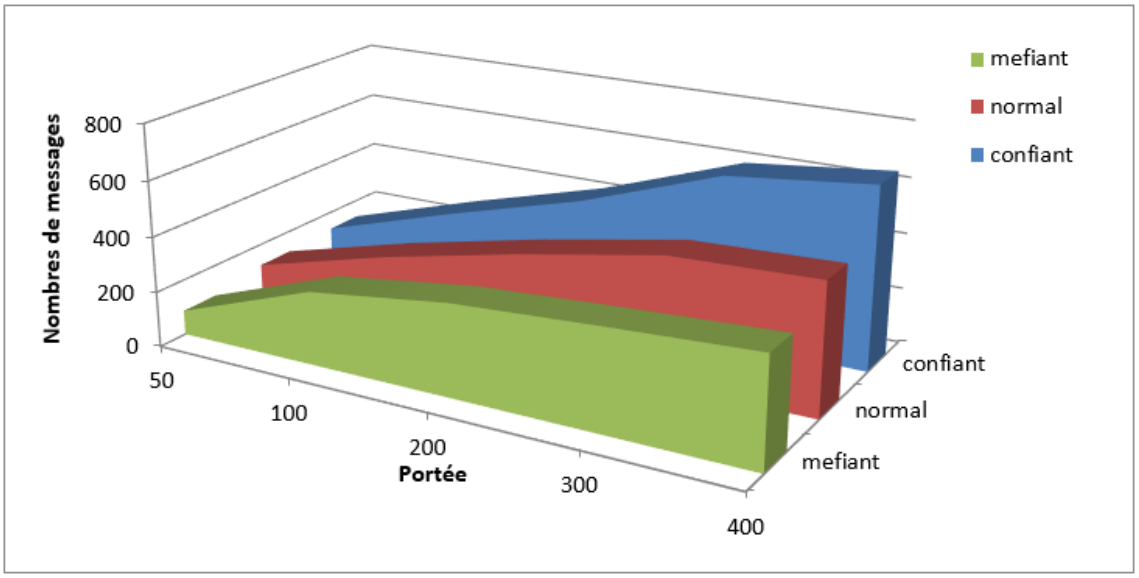

FIGURE 4.10 – Nombres de messages échangés en fonction de la portée pour les trois niveaux de confiance.

La figure 4.11 montre le coût de communication des deux protocoles TRP et μ TRP. Elle montre que le protocole μ TRP qui est une amélioration de TRP est nettement moins coûteux en terme de coût de communication, et ceux avec l'appui des courbes elliptiques. en effet grâce à ces dernières la taille des clés est réduite ce qui a pour effet de minimiser le coût de communication lors de cette étape d'établissement de clés.

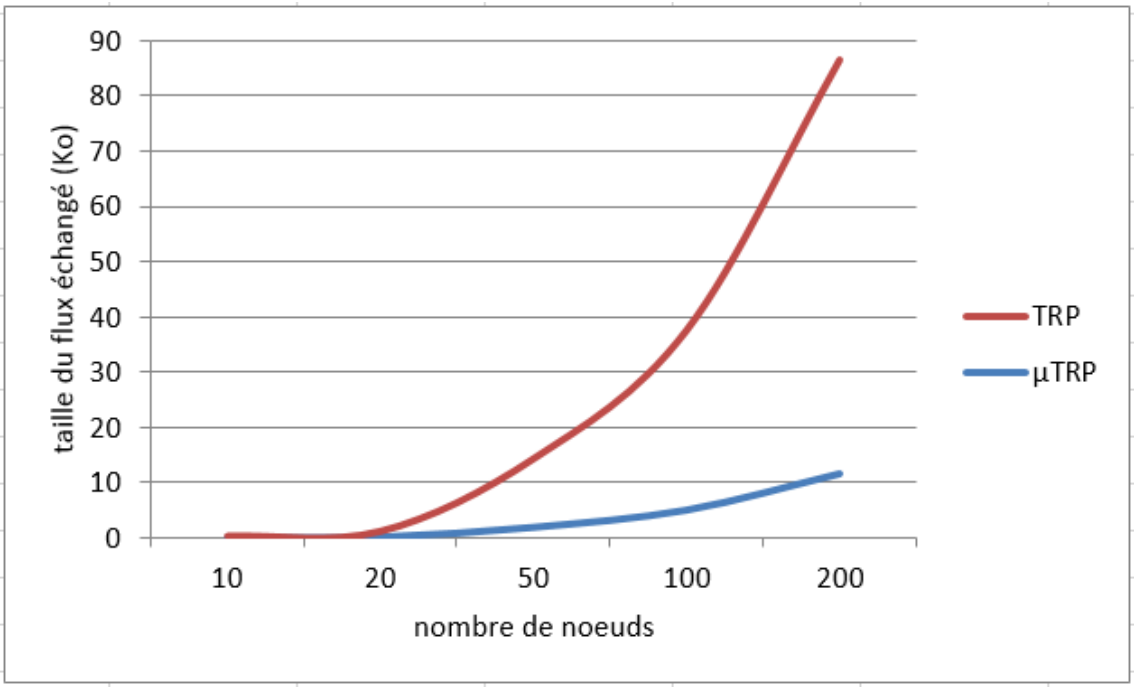

FIGURE 4.11 – coût de communication de μ TRP et TRP.

4.4 Conclusion

Dans ce chapitre, nous avons simulés notre protocole à l'aide du simulateur que nous avons conçus à cet effet. Nous avons présenté les résultats de simulation obtenus schématisés par des graphes, et fait une discussion de ces derniers.

Conclusion Générale

La gestion de clés de groupe dans MANETs se révélé être un challenge, à cause du changement fréquent de la topologie de ces derniers dû au mouvement des nœuds. Pour faire face à ce problème des solutions qui ont été proposées organise le groupe en cluster, l'avantage de la clusterisation est que le renouvellement des clés se fait rapidement, mais en même temps elle apporte de nouveaux défis telles que la synchronisation et le coût de communication.

Le travail consigné dans ce mémoire a été le fruit d'une étude menée dans le contexte des réseaux Ad-Hoc, et ce relativement au problème de sécurité dans ces derniers. Nous avons ainsi étudiés et classés différents protocoles de gestion de clés proposés dans la littérature spécialisée, au travers desquels les buts de sécurité face aux attaques potentielles sont accomplis de manière plus ou moins satisfaisante. Cette étude nous a permis d'adapter une amélioration donc une contribution qui consiste à introduire la notion de cryptographie à courbes elliptiques dans un protocole d'accord de clés, ce qui réduit considérablement la taille de la clé calculée, et de ce fait réduit le nombre de messages échangés afin de partager cette dernière. Théoriquement notre solution montre qu'elle peut fournir le même niveau de sécurité et de rapidité pour le partage de clés avec des coûts minime.

En perspective on souhaiterait pouvoir simuler le coût de calcul des clés de notre protocole, et pouvoir comparer les performances de notre proposition avec d'autres protocoles existants dans la littérature. D'autre aspects des réseaux Ad-Hoc peuvent être pris en considération lors de proposition de protocoles de gestion de clés comme la consommation d'énergie, il serait donc plausible d'avoir aussi comme perspective d'étudier cette aspect là pour le protocole proposé.

Bibliographie

[1] Krishnan Kumar, J Nafeesa Begum, V Sumathy, et al. A novel approach towards cost effective region-based group key agreement protocol for ad hoc networks using elliptic curve cryptography. *Int'l J. of Communications, Network and System Sciences*, 3(04) :369, 2010.

[2] Andrew S Tanenbaum. Computer networks, 4-th edition. *ed : Prentice Hall*, 2003.

[3] Kaouther Drira, Hamida Seba, and Hamamache Kheddouci. Ecgk : An efficient clustering scheme for group key management in manets. *Computer Communications*, 33(9) :1094–1107, 2010.

[4] Ravi K Balachandran, Byrav Ramamurthy, Xukai Zou, and N Variyam Vinodchandran. Crtdh : an efficient key agreement scheme for secure group communications in wireless ad hoc networks. In *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, volume 2, pages 1123–1127. IEEE, 2005.

[5] Daniel Augot, Raghav Bhaskar, Valérie Issarny, and Daniele Sacchetti. An efficient group key agreement protocol for ad hoc networks. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 576–580. IEEE, 2005.

[6] Daniel Augot, Raghav Bhaskar, Valérie Issarny, and Daniele Sacchetti. A three round authenticated group key agreement protocol for ad hoc networks. *Pervasive and Mobile Computing*, 3(1) :36–52, 2007.

[7] Sencun Zhu, Sanjeev Setia, Shouhuai Xu, and Sushil Jajodia. Gkmpan : An efficient group rekeying scheme for secure multicast in ad-hoc networks. *Journal of Computer Security*, 14(4) :301–325, 2006.

[8] Anindo Mukherjee, Anurag Gupta, and Dharma P Agrawal. Distributed key management for dynamic groups in manets. *Pervasive and Mobile Computing*, 4(4) :562–578, 2008.

[9] Mark Manulis. Contributory group key agreement protocols, revisited for mobile ad-hoc groups. In *Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[10] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Group key agreement efficient in communication. *Computers, IEEE Transactions on*, 53(7) :905–921, 2004.

[11] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *Parallel and Distributed Systems, IEEE Transactions on*, 11(8) :769–780, 2000.

[12] Yongdae Kim, Adrian Perrig, and Gene Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security (TISSEC)*, 7(1) :60–96, 2004.

[13] Lijun Liao and Mark Manulis. Tree-based group key agreement framework for mobile ad-hoc networks. *Future Generation Computer Systems*, 23(6) :787–803, 2007.

[14] Maria Striki and John S Baras. Fault-tolerance and efficiency considerations for key distribution protocols in manets. In *Proc. of the 37th Conference on Information Sciences and Systems (CISS), Johns Hopkins University, Baltimore, MD*, 2003.

[15] Maria Striki, John S Baras, and Kyriakos Manousakis. A robust, distributed tgdh-based scheme for secure group communications in manet. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 5, pages 2249–2255. IEEE, 2006.

[16] Xiang-Yang Li, Yu Wang, and Ophir Frieder. Efficient hybrid key agreement protocol for wireless ad hoc networks. In *Computer Communications and Networks, 2002. Proceedings. Eleventh International Conference on*, pages 404–409. IEEE, 2002.

[17] Whitfield Diffie and Martin E Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6) :644–654, 1976.

[18] Yuh-Min Tseng, Chou-Chen Yang, and Der-Ren Liao. A secure group communication protocol for ad hoc wireless networks. *Advances in Wireless Ad Hoc and Sensor Networks, Signals and Communication Technology Series, Springer*, 2007.

[19] Ulaş C Kozat, George Kondylis, Bo Ryu, and Mahesh K Marina. Virtual dynamic backbone for mobile ad hoc networks. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 1, pages 250–255. IEEE, 2001.

[20] Chaiporn Jaikaeo and Chien-Chung Shen. Adaptive backbone-based multicast for ad hoc networks. In *Communications, 2002. ICC 2002. IEEE International Conference on*, volume 5, pages 3149–3155. IEEE, 2002.

[21] Jin-Hee Cho, Ray Chen, and Ding-Chau Wang. Performance optimization of region-based group key management in mobile ad hoc networks. *Performance Evaluation*, 65(5) :319–344, 2008.

[22] Jason H Li, Bobby Bhattacharjee, Miao Yu, and Renato Levy. A scalable key management and clustering scheme for wireless ad hoc and sensor networks. *Future Generation Computer Systems*, 24(8) :860–869, 2008.

[23] Jason H Li, Renato Levy, Anna Teittinen, et al. A mobility-resistant efficient clustering approach for ad hoc and sensor networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 10(2) :1–12, 2006.

[24] Xia Li, Jill Slay, and Shaokai Yu. Evaluating trust in mobile ad hoc networks. In *Proc. of the International Conference on Computational Intelligence and Security*. Citeseer, 2005.

[25] Zhaoyu Liu, Anthony W Joy, and Robert A Thompson. A dynamic trust model for mobile ad hoc networks. In *Distributed Computing Systems, 2004. FTDCS 2004. Proceedings. 10th IEEE International Workshop on Future Trends of*, pages 80–85. IEEE, 2004.

[26] Xukai Zou, Byrav Ramamurthy, and Spyros S Magliveras. *Secure group communications over data networks*. Springer Science & Business Media, 2007.

[27] Dennis J Baker and Anthony Ephremides. The architectural organization of a mobile radio network via a distributed algorithm. *Communications, IEEE Transactions on*, 29(11) :1694–1701, 1981.

[28] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[29] Lawrence C Washington. *Elliptic curves : number theory and cryptography*. CRC press, 2008.

[30] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987.

[31] Brown L Daniel R. Elliptic curve cryptography, May 2009. Certicom Research.

[32] Alfred J Menezes and Scott A Vanstone. Elliptic curve cryptosystems and their implementation. *Journal of Cryptology*, 6(4) :209–224, 1993.

[33] M. Steiner, G. Tsudik, and M. Wainder. Diffie-hellman key distribution extended to group communication. *ACM Symposium on Computer and Communication Security*, Mars 1996.

[34] Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. *in CCS'00 : Proceedings of the 7th ACM conference on Computer and communications security. New York, NY, USA : ACM Press*, pages 235–244, 2000.

[35] Y. Amir, Y.Kim, C. Nita-Rotaru, and G. Tsudik. On the performance of group key-agreement protocols. *in Proc. of the 22nd IEEE International Conference on Distributed Computing Systems, Viena, Austria*, June 2002.

[36] K. S. Hagzan and H.-P. Bischof. The performance of group diffie-hellman paradigms. *in The 2004 International Conference on Wireless Networks (ICWN'04), Las Vegas, Nevada, USA*, June 2004.

[37] Lein Harn and Changlu Lin. Efficient group diffie–hellman key agreement protocols. *Computers & Electrical Engineering*, 40(6) :1972–1980, 2014.

[38] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic hardware and embedded systems-CHES 2004*, pages 119–132. Springer, 2004.

[39] DJ. Malan, M. Welsh, and MD. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. *In :First annual IEEE communications society conference on sensor and ad hoc communications and networks, IEEE SECON*, 2004.

[40] Yong Wang, Byrav Ramamurthy, and Xukai Zou. The performance of elliptic curve based group diffie-hellman protocols for secure group communication over ad hoc networks. In *Communications, 2006. ICC'06. IEEE International Conference on*, volume 5, pages 2243–2248. IEEE, 2006.

[41] Vankamamidi S. Naresh and Nistala V.E.S. Murthy. Elliptic curve based dynamic contributory group key agreement protocol for secure group communication over ad-hoc networks. *International Journal of Network Security*, 2014.

Résumé

La gestion de clés est un problème fondamental à résoudre pour la sécurité des communications de groupe dans les réseaux. C'est un domaine très étudié dans la littérature. La caractéristique particulière des réseaux mobiles Ad-Hoc rend l'adaptation des solutions proposées un vrai défi. Des solutions récentes pour ces réseaux s'orientent vers un accord de clés hiérarchique. Par ailleurs la technologie de la cryptographie à courbes elliptiques (ECC), est devenue le choix de chiffrement pour les réseaux Ad-Hoc. Le chiffrement par courbes elliptiques utilise des clés relativement petites et est mathématiquement très efficace, qui rend idéal pour les petits dispositifs de communication utilisés aujourd'hui.

Ce travail consiste en un état de l'art critique sur les protocoles de gestion de clés dans les réseaux mobiles Ad-Hoc, particulièrement les solutions hiérarchiques d'accord de clés de groupe. L'étude s'oriente ensuite à l'étude de la technologie ECC et son apport pour les solutions d'accord de clés dans les réseaux mobiles Ad-Hoc. Le travail s'achève ensuite par une proposition d'une nouvelle solution et l'évaluation des performances de cette dernières.

Mots clés :

Gestion de clés de groupe, Accord de clés, Cryptographie à courbes elliptiques, réseaux mobiles Ad-Hoc, ECGK, TRP.

Abstract

The group key management is a fundamental problem to solve for securing group communications in the networks. This area is very studied in literature. The particular characteristic of mobile Ad Hoc networks returns the adaptation of the suggested solutions a true challenge. Recent solutions for these networks are directed towards a hierarchical keys agreement. In addition elliptic curves cryptography (ECC) became the choice of encryption for Ad Hoc networks. Elliptic curve uses very small keys is mathematically very effective, which makes it ideal for communication with small devices used today.

This work consists in making critical states of the arts of group key management in the mobile Ad Hoc networks, particularly the hierarchical solutions of Group Key Agreements. The study will be directed then to ECC technology and its contribution for the solutions of group key agreements in the mobile Ad Hoc networks. The work will be completed by proposing for a new solution and evaluating its performance.

Keywords :

Group Key Management, Key Agreement, Elliptic Curves Cryptography (ECC), Mobile Ad-Hoc NETworks (MANETs), ECGK, TRP.